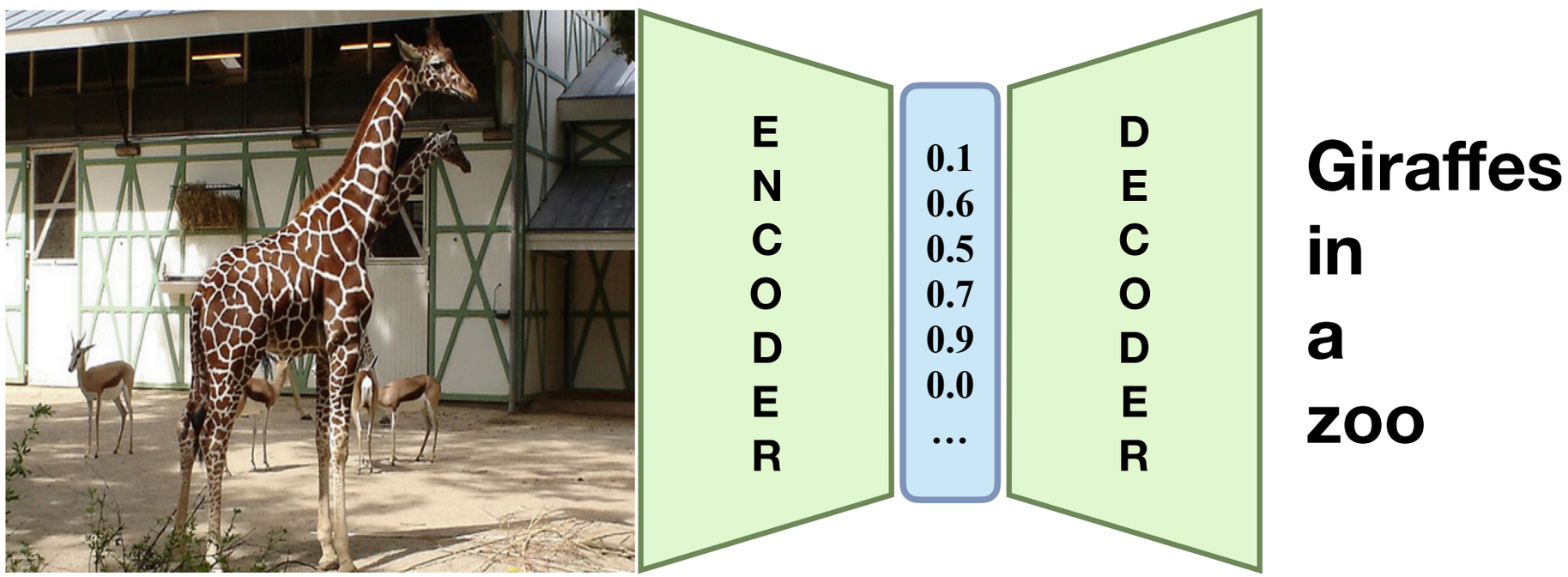
The transformer model and transfer learning

Recap

Bidirectional and **deep** RNNs are extensions of the basic (forward, single-layer) RNN architecture allowing access to “future” inputs, higher capacity, and learning different levels of abstraction

Encoder-decoder architectures are a general framework for training end-to-end models for a broad range of tasks

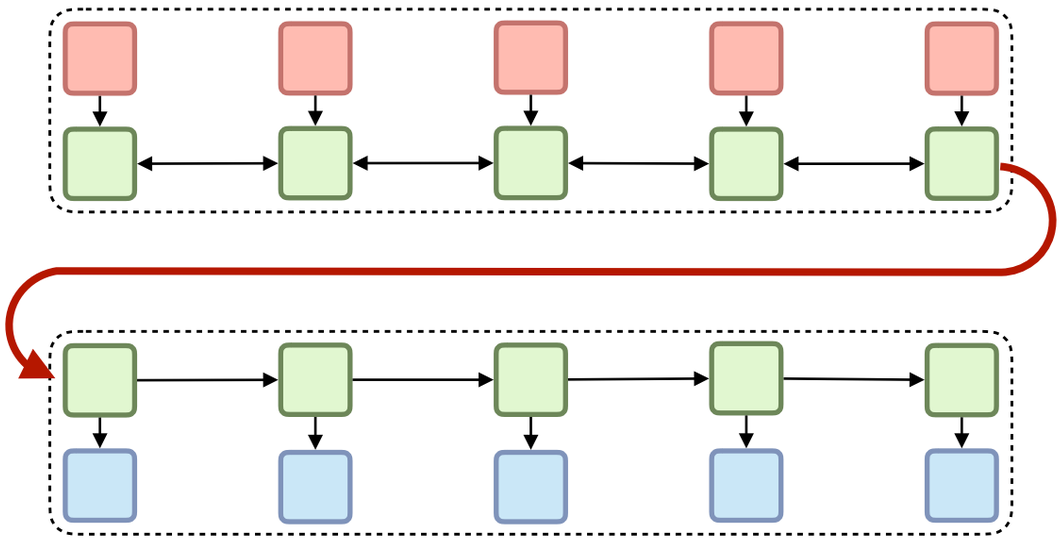


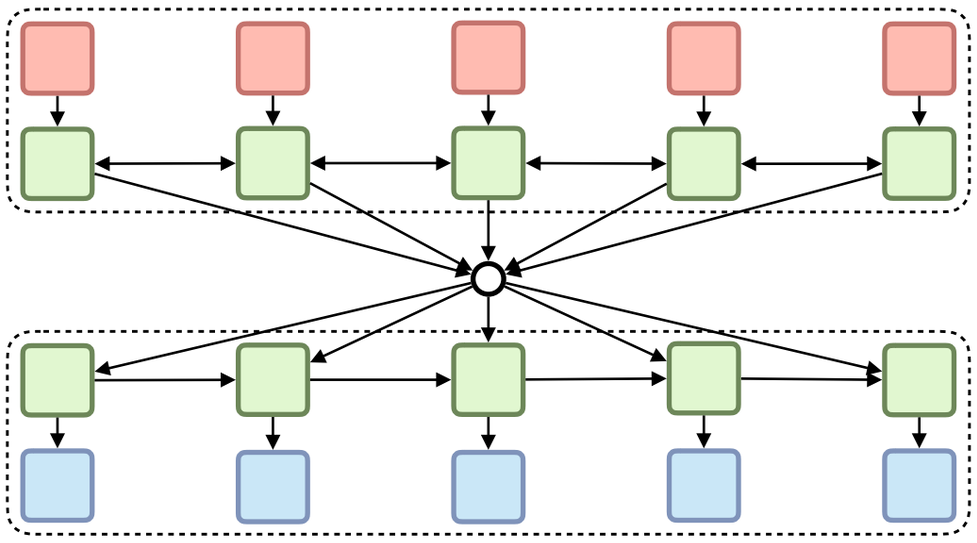
Recap

RNNs can be used to implement **sequence-to-sequence** models for e.g. machine translation, summarization, and dialogue tasks

Fixed-length encoded representation is a **bottleneck** for basic seq2seq models

Attention relieves this bottleneck, allowing encoder state to serve as "random-access memory" for the decoder

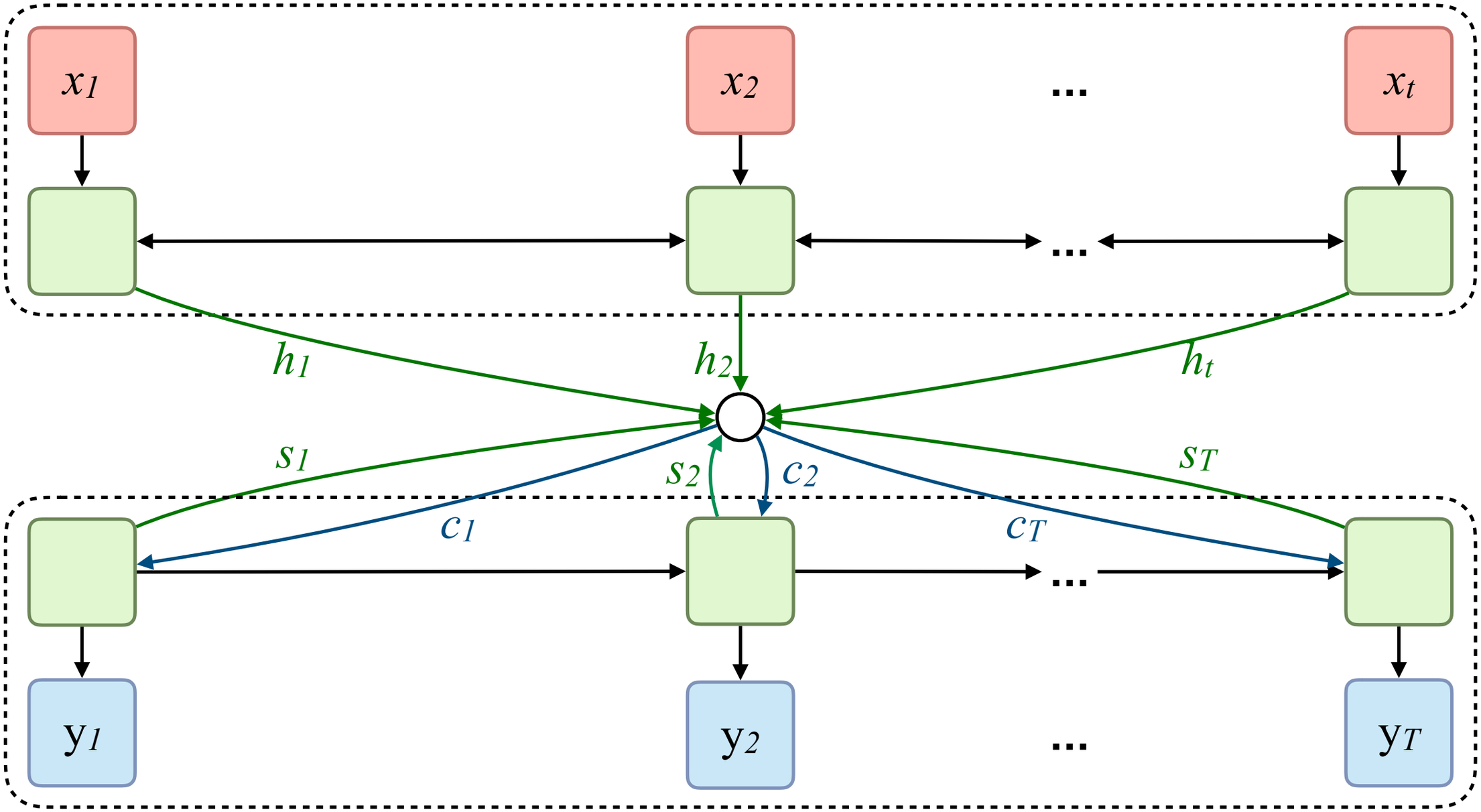




Self-attention and the transformer architecture

Generalizing attention

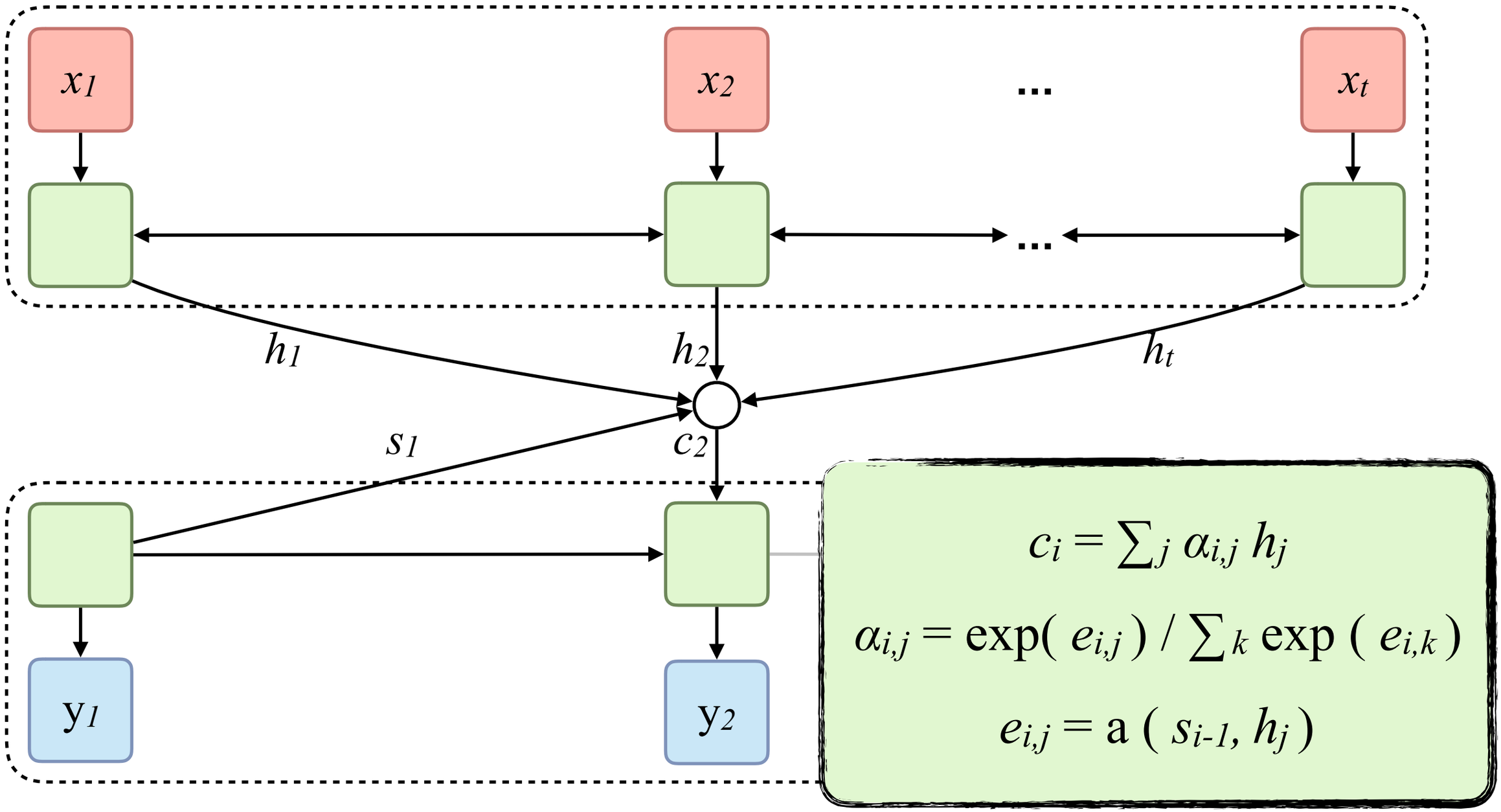
Contexts c_i calculated as function of encoder and decoder states h_j and s_i



Bahdanau *et al*. (2014) *Neural machine translation by jointly learning to align and translate*

Generalizing attention

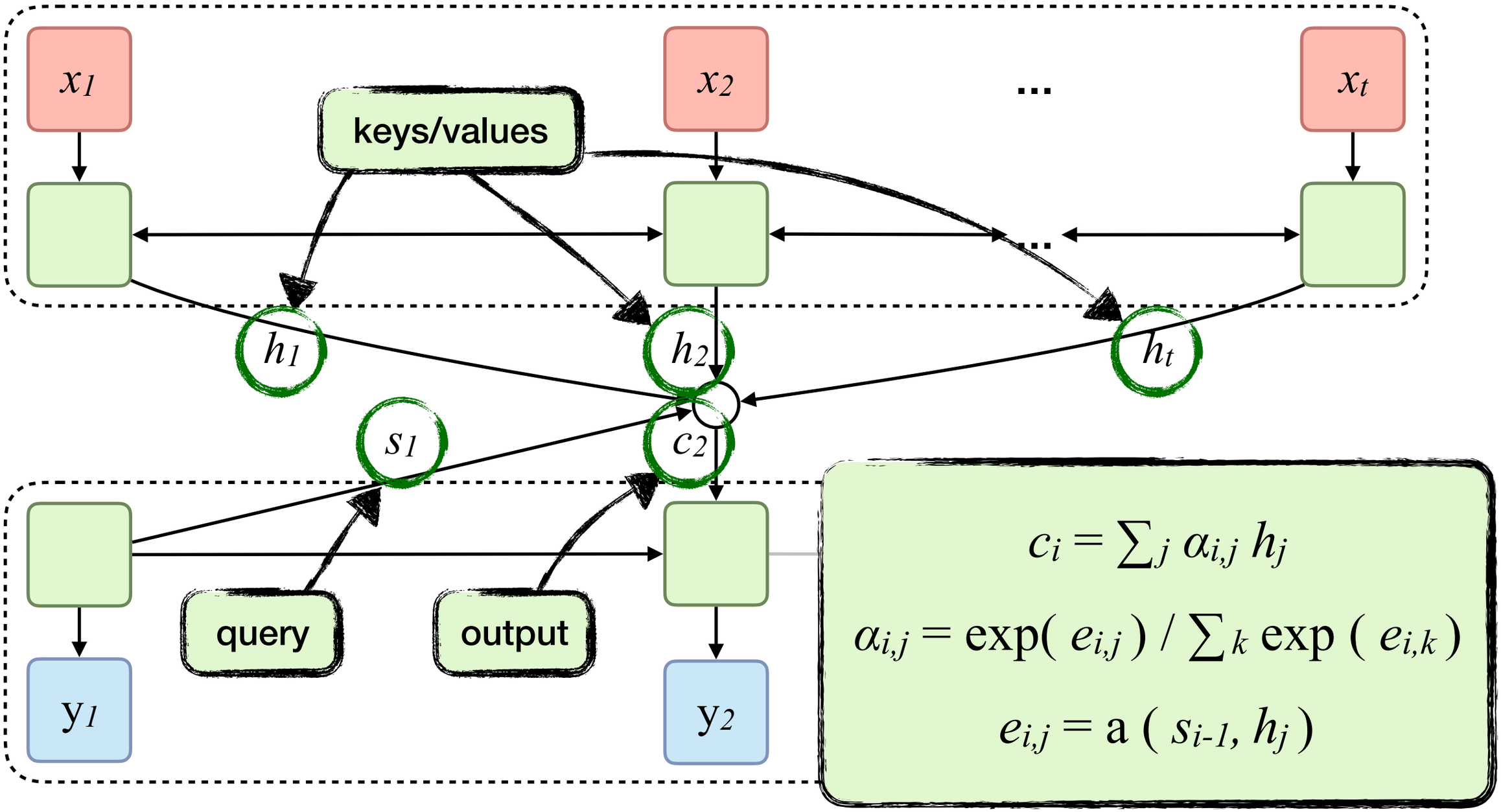
Context c_i calculated as weighted sum of encoder states h_j , weights $\alpha_{i,j}$ calculated by comparing decoder state s_{i-1} with all encoder states h_j



Bahdanau *et al*. (2014) *Neural machine translation by jointly learning to align and translate*

Generalizing attention

Output c_i calculated as weighted sum of **values** h_j , weights $\alpha_{i,j}$ calculated by comparing **query** s_{i-1} with all **keys** h_j



Bahdanau *et al*. (2014) *Neural machine translation by jointly learning to align and translate*

Generalizing attention

Attention outputs calculated based on queries, keys, and values

Rough intuition:

- **Query** represents what information is needed
- **Keys** represent information stored in memory
- **Values** are the contents of the memory

Query compared with each key using alignment/attention **score function** determining their compatibility to calculate a weight for the corresponding value; output is weighted sum of values

Information retrieval analogy: **query** is search terms, **key** the part of the document that is matched against, and **value** the returned document

Generalizing attention

Alignment / attention score functions

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

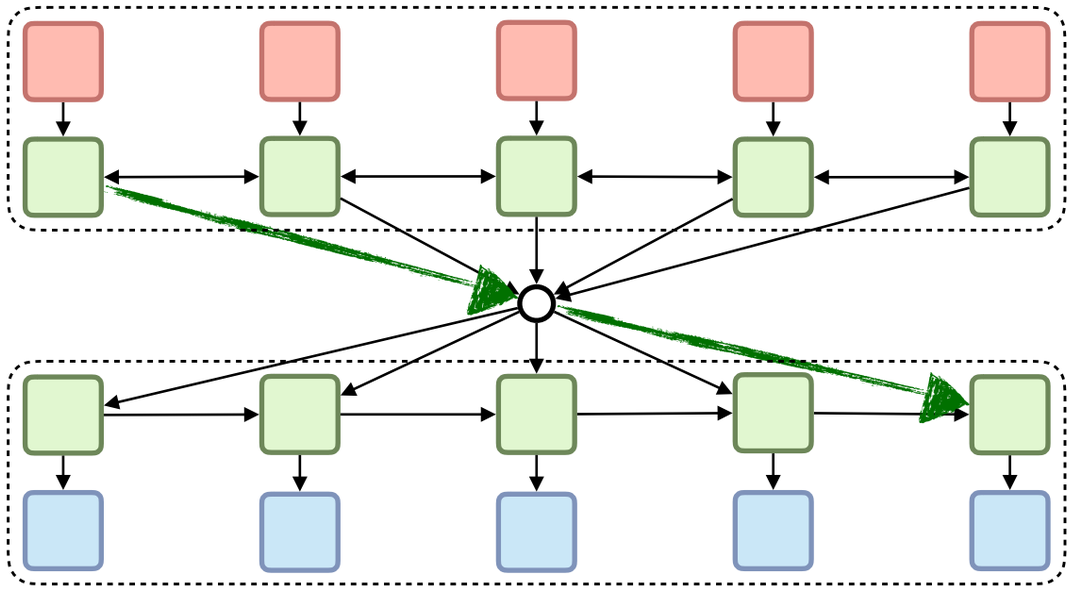
Table from Weng (2018) Attention? Attention!

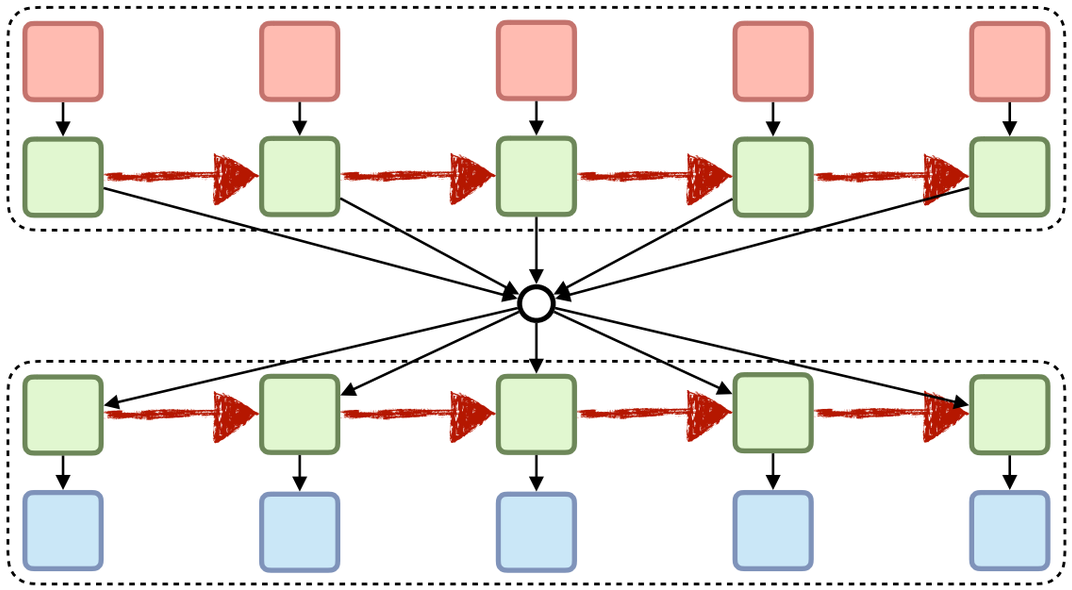
Self-attention

In models discussed so far, **decoder** attends to **encoder** states

→ decoder RNN has constant distance to any encoder RNN state

… but distances between states of the **same RNNs** remain linear





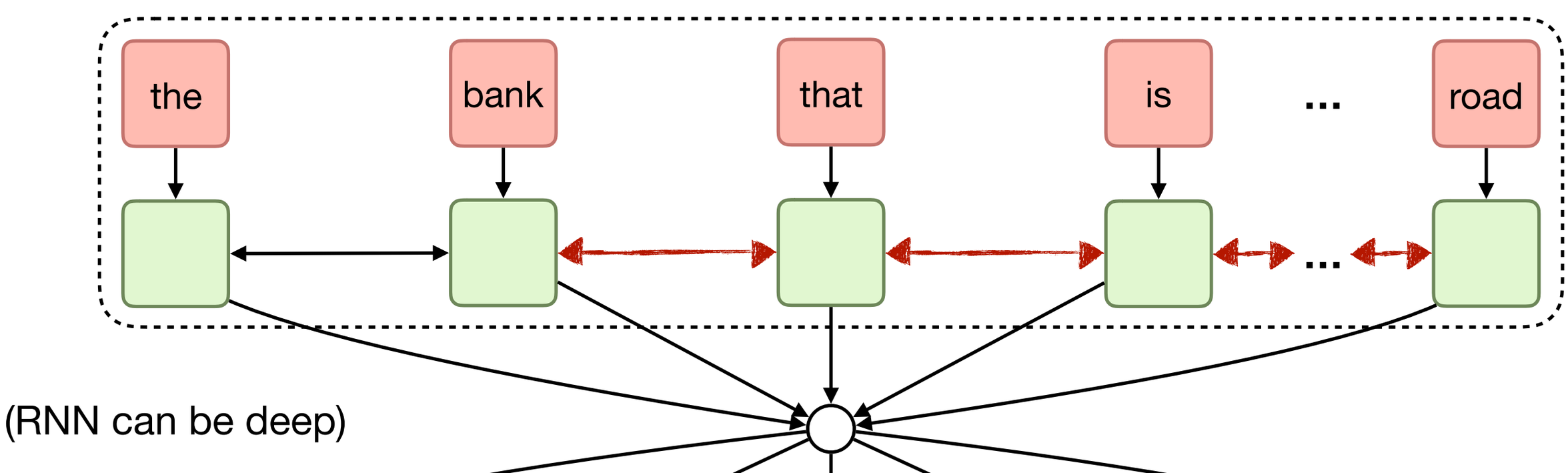
Self-attention

Consider selecting the right sense of *bank* (river vs. financial) in

*The **bank** that is on the other side of that wide **road***

vs.

*The **bank** that is on the other side of that wide **river***



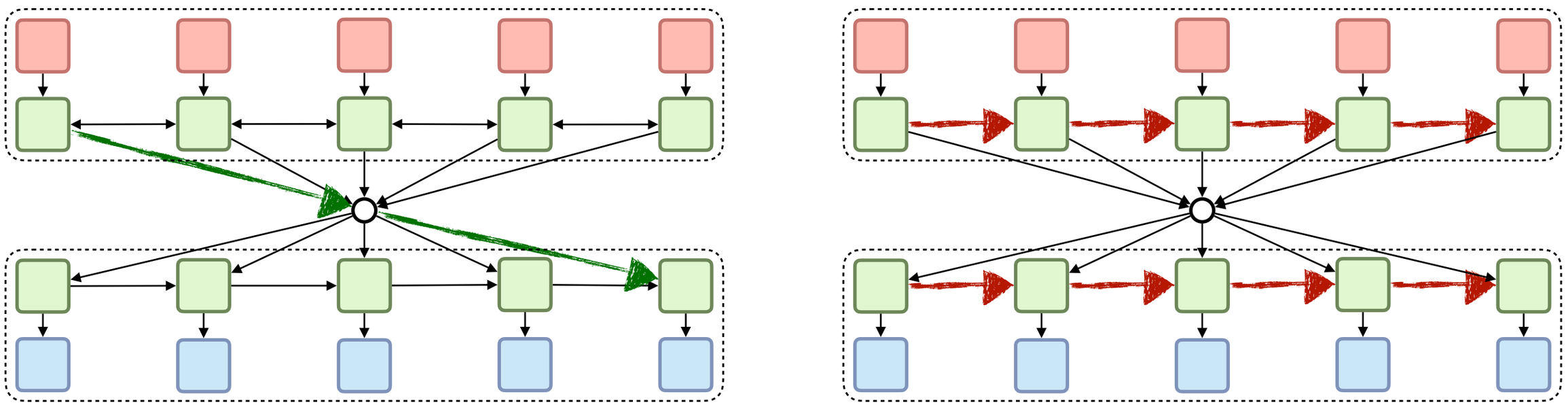
Self-attention

In models discussed so far, **decoder** attends to **encoder** states

→ decoder RNN has constant distance to any encoder RNN state

… but distances between states of the **same RNNs** remain linear

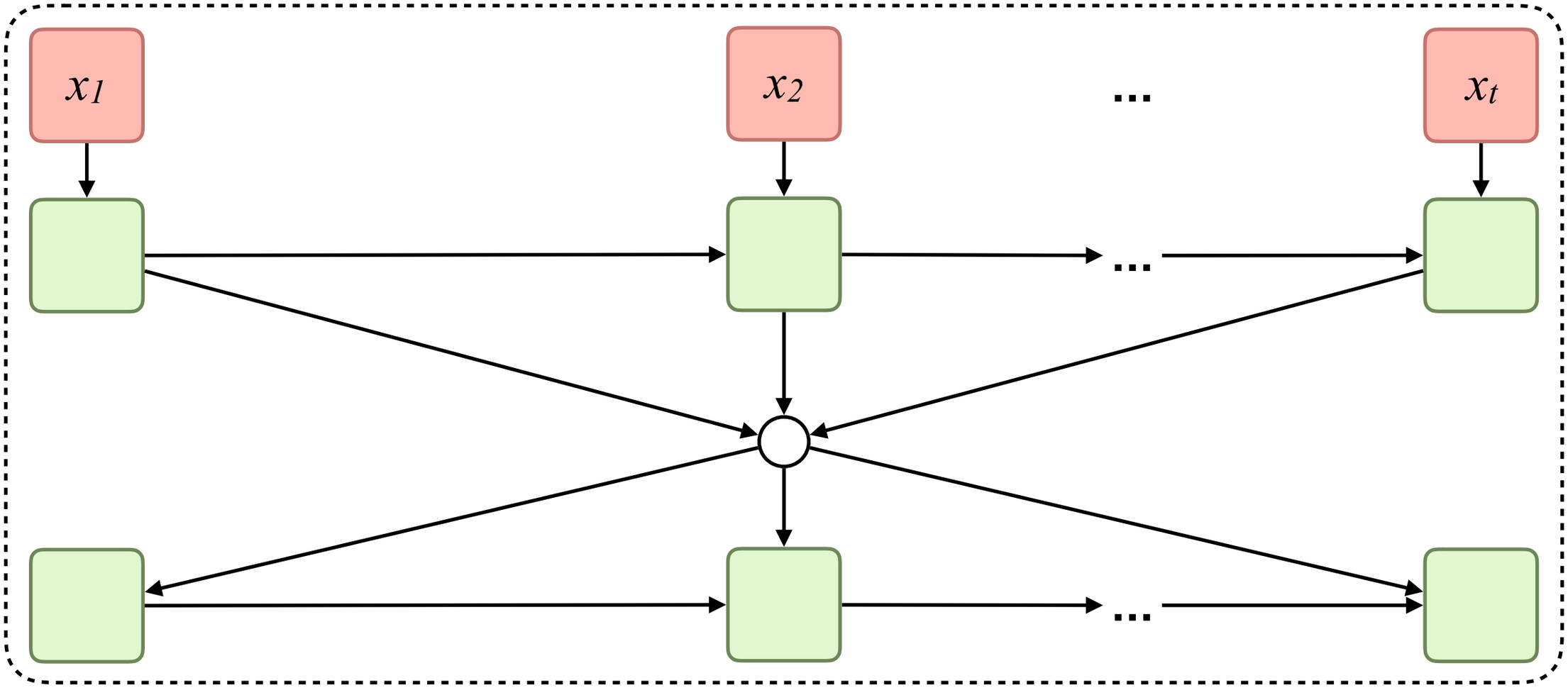
→ apply attention also *within* the RNNs!



Cheng *et al*. (2016) *Long Short-Term Memory-Networks for Machine Reading*

Self-attention

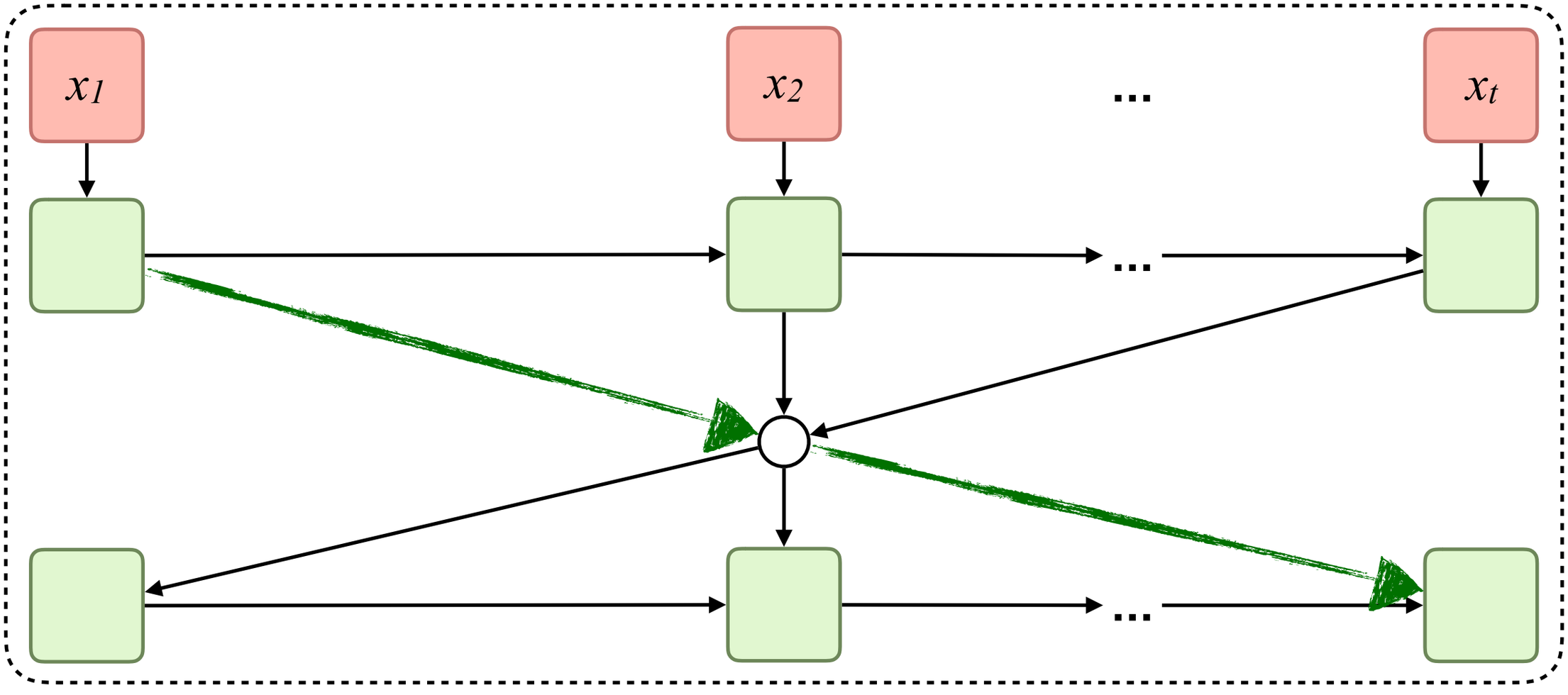
In **self-attention** (or intra-attention), a model attends to its own inputs



Self-attention

In **self-attention** (or intra-attention), a model attends to its own inputs

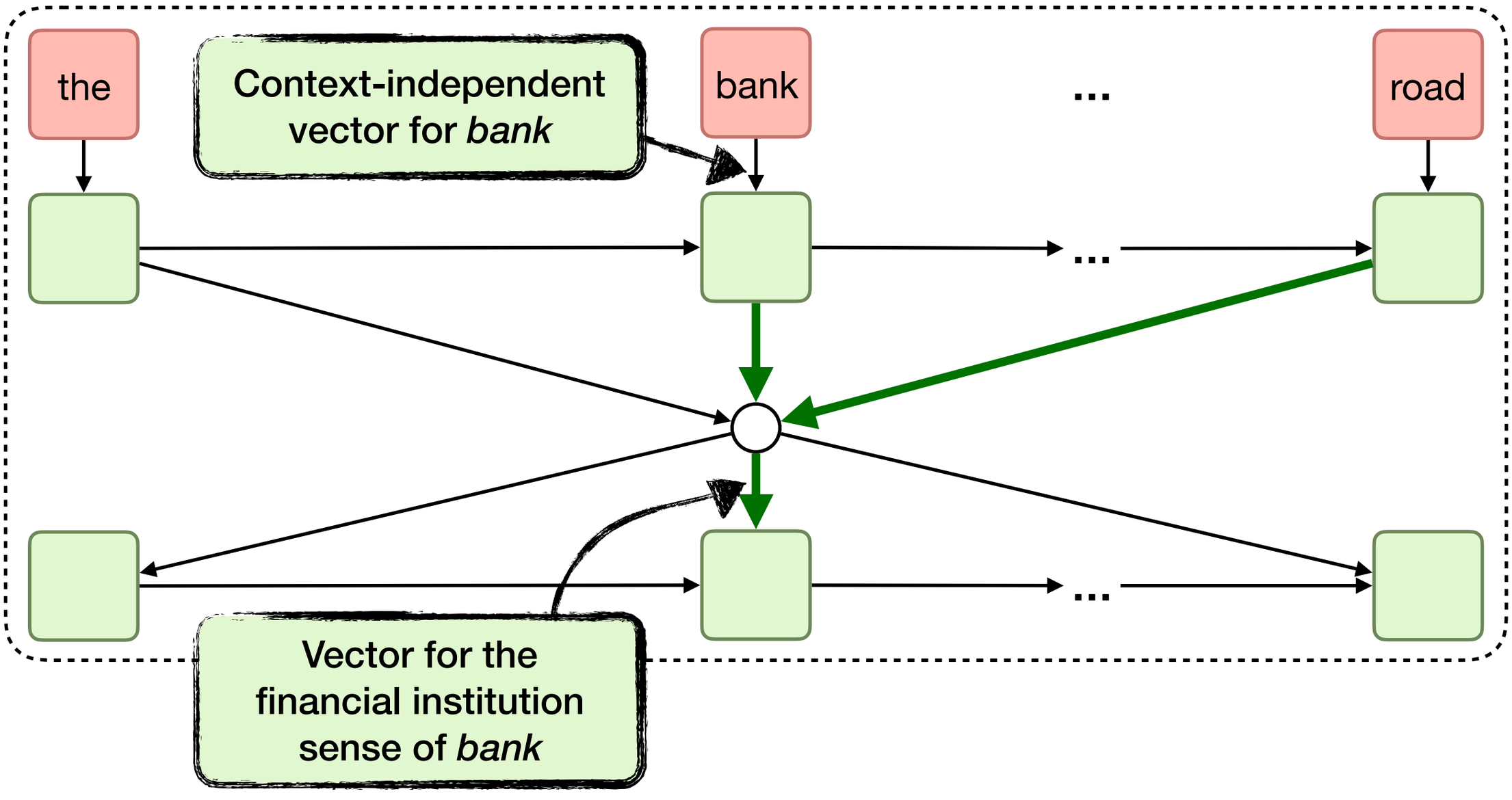
→ constant distance between all pairs



Self-attention

In **self-attention** (or intra-attention), a model attends to its own inputs

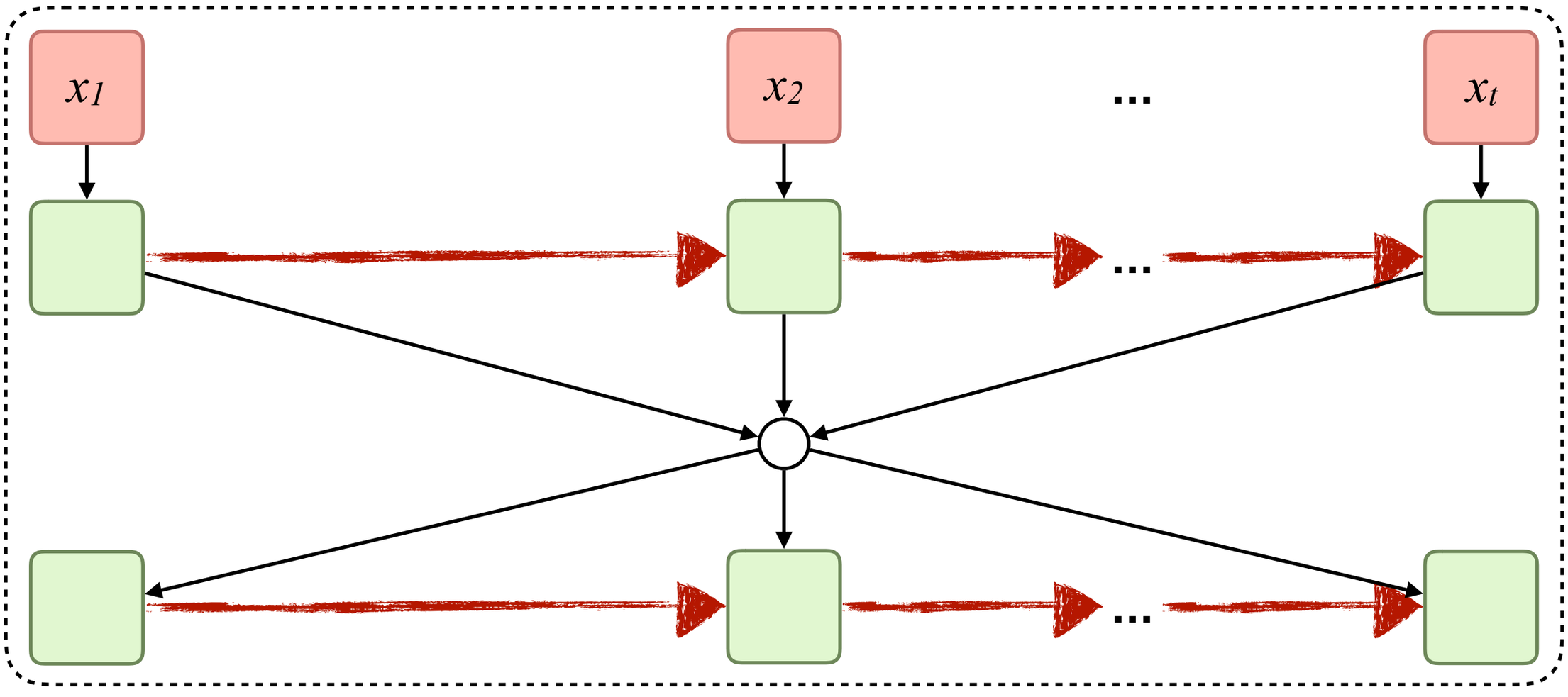
→ allows model to learn **contextual** embeddings



Self-attention

In **self-attention** (or intra-attention), a model attends to its own inputs

(in this model, do we still need the recurrent connections?)



Transformer

Model introduced by Vasvani et al. (2017) *Attention Is All You Need*

Based on dot-product **self-attention** *without recurrence* (RNN)
(sequence information preserved by added **positional encodings**)

Emphasizes computational efficiency; aims to minimize sequential computation, maximize **parallelizability**

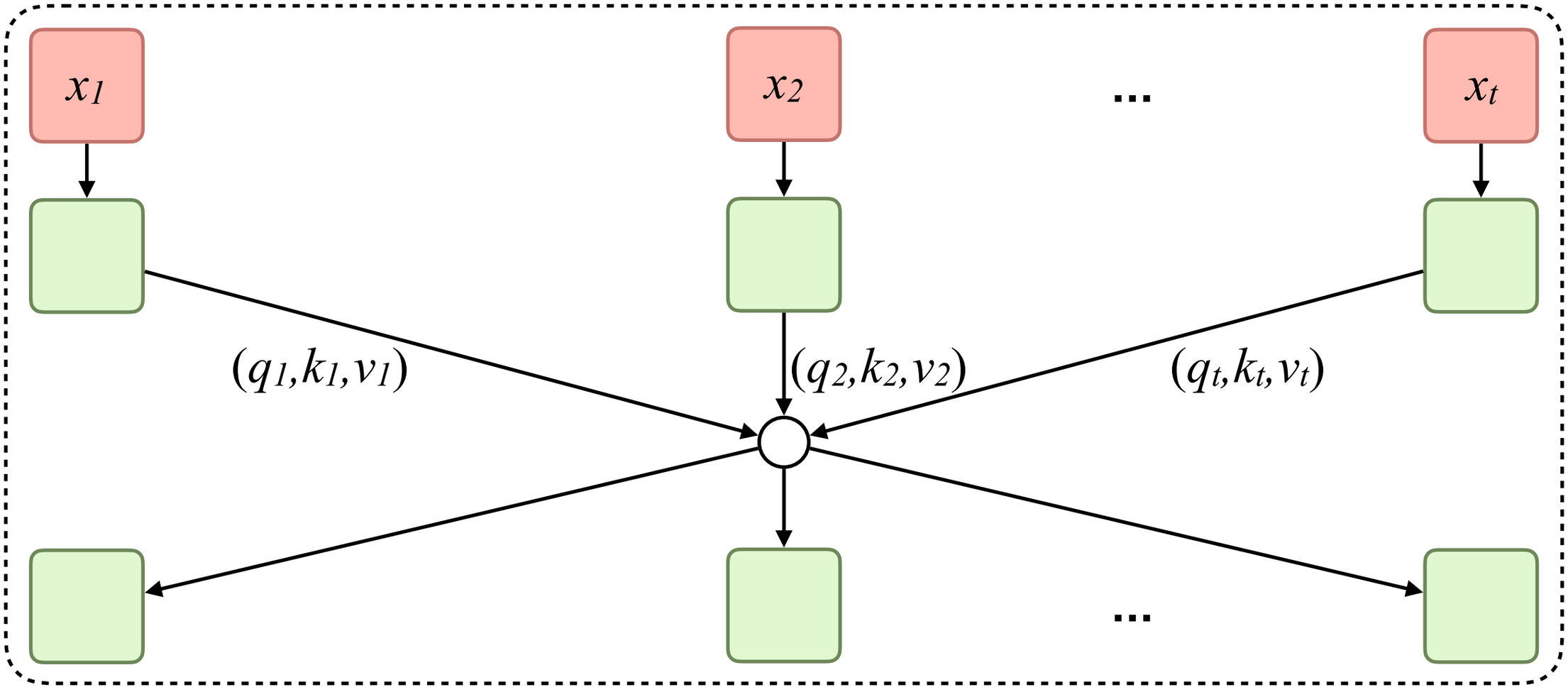
Substantially improved SOTA performance in machine translation when introduced, applied to very broad range of tasks in NLP and other domains since

Vaswani *et al*. (2017) *Attention Is All You Need*

Transformer

Queries q_i, keys k_i and values v_i computed independently for each state

No recurrence → all q_i, k_i and v_i available simultaneously



Vaswani *et al*. (2017) *Attention Is All You Need*

Transformer

Transformer dot-product attention calculation:

Keys, queries and values q_i, k_i and v_i stacked into matrices Q, K and V

Scaled dot-product attention outputs calculated for all positions as

$$\text{Attention}(Q, K, V) = \text{softmax}(Q K^{\mathrm{T}} / s) V$$

dot products

Where the scaling factor s is the square root of the number of dimensions in the key/query vectors.

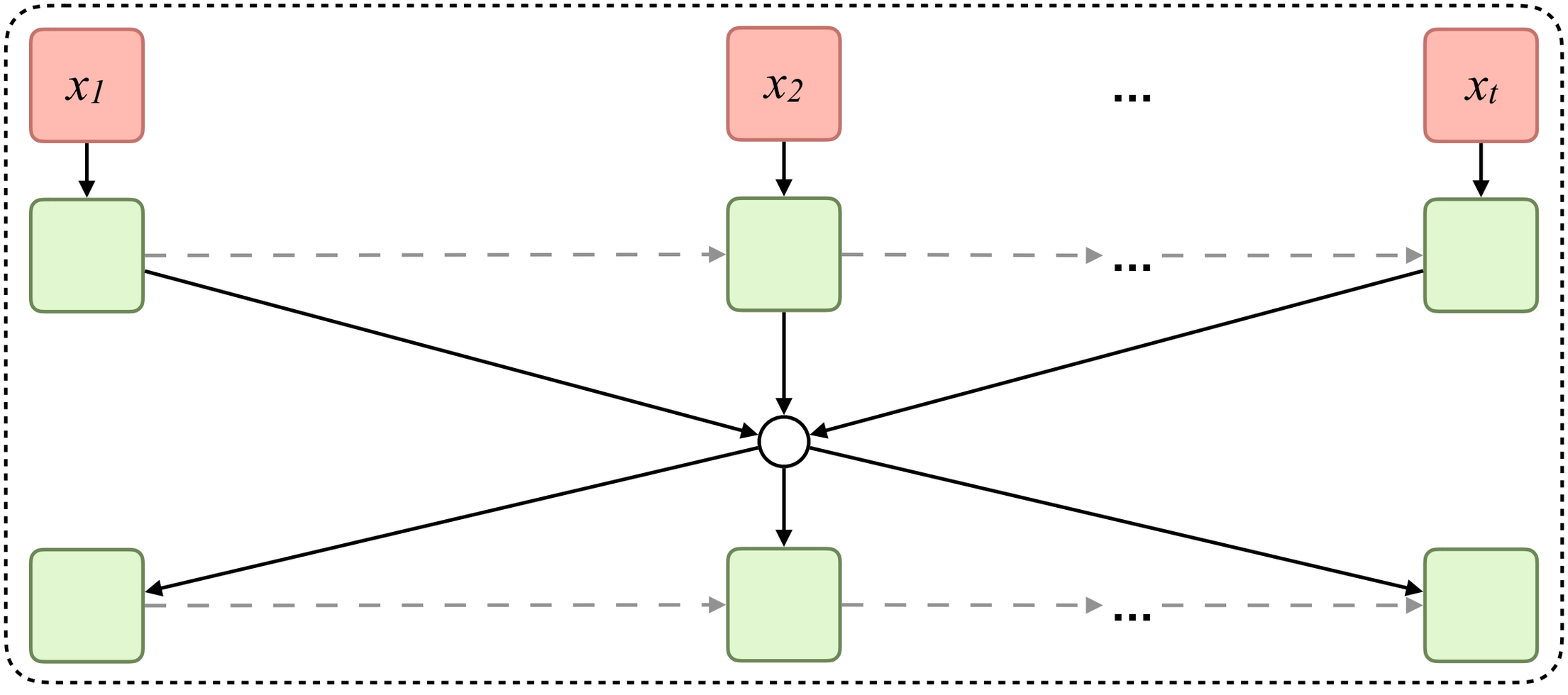
Tweak to avoid small gradient values, not important for understanding the model

Vaswani *et al*. (2017) *Attention Is All You Need*

Transformer

Without recurrent connections, the model loses sequence information

→ we're back to "man bites dog" = "dog bites man"



Vaswani *et al*. (2017) *Attention Is All You Need*

Transformer

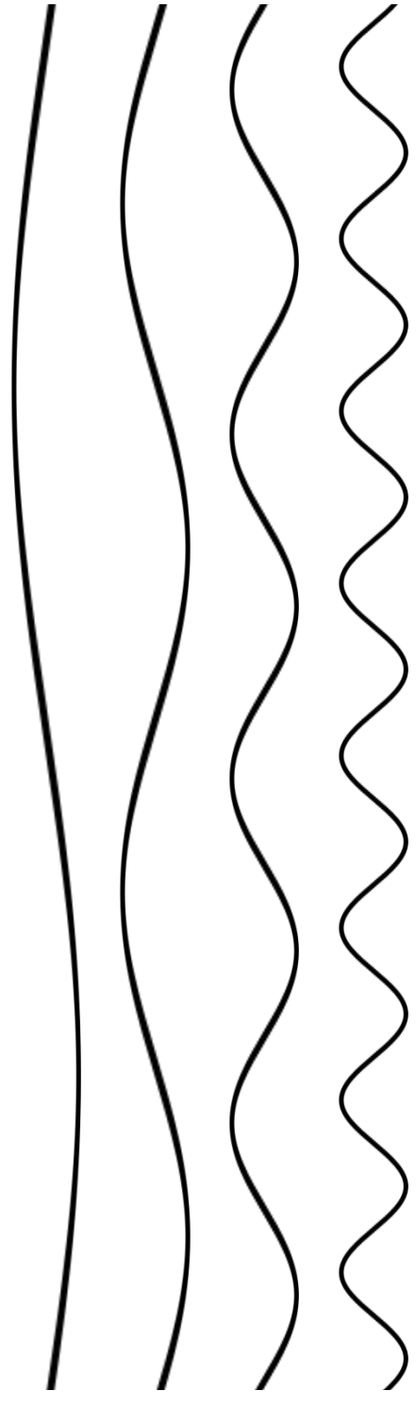
Positional encodings are added to input vectors to give the model information on position in sequence

Intuition: consider the binary encoding of integer position values (0, 1, 2, …)

The lowest bit changed with a high frequency, the next-lowest with half that, etc.

The value is a combination of these

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1



Vaswani *et al*. (2017) *Attention Is All You Need*

Transformer

Positional encodings are n-dimensional vectors where the value for a position *pos* and dimension *i* is calculated as

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

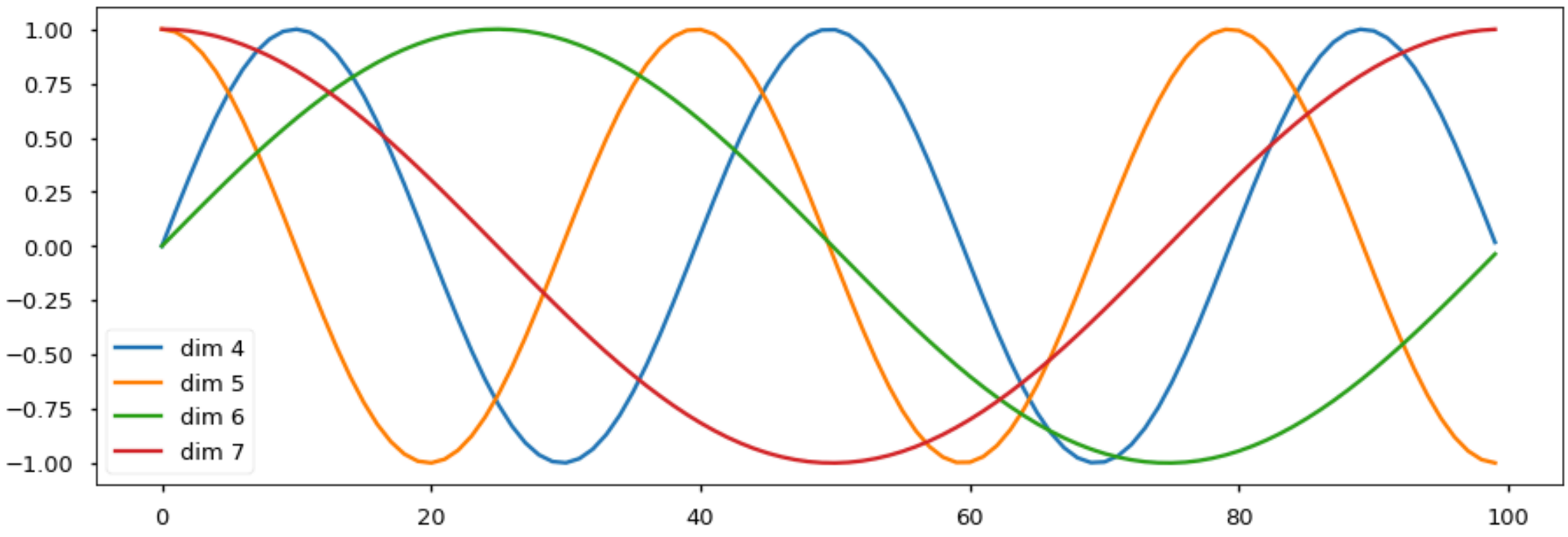


Figure from https://nlp.seas.harvard.edu/2018/04/03/attention.html

Transformer

Illustration of position 0-19 (x) encoding values for dimensions 0-100 (y)

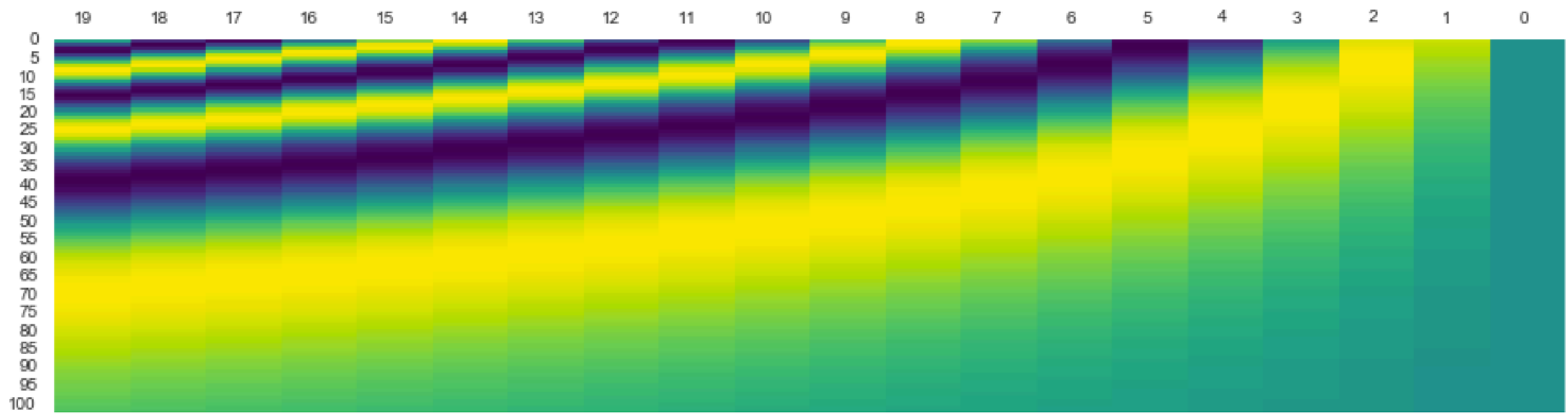


Figure from http://jalammar.github.io/illustrated-transformer/

Transformer

A high-level view of transformer architecture

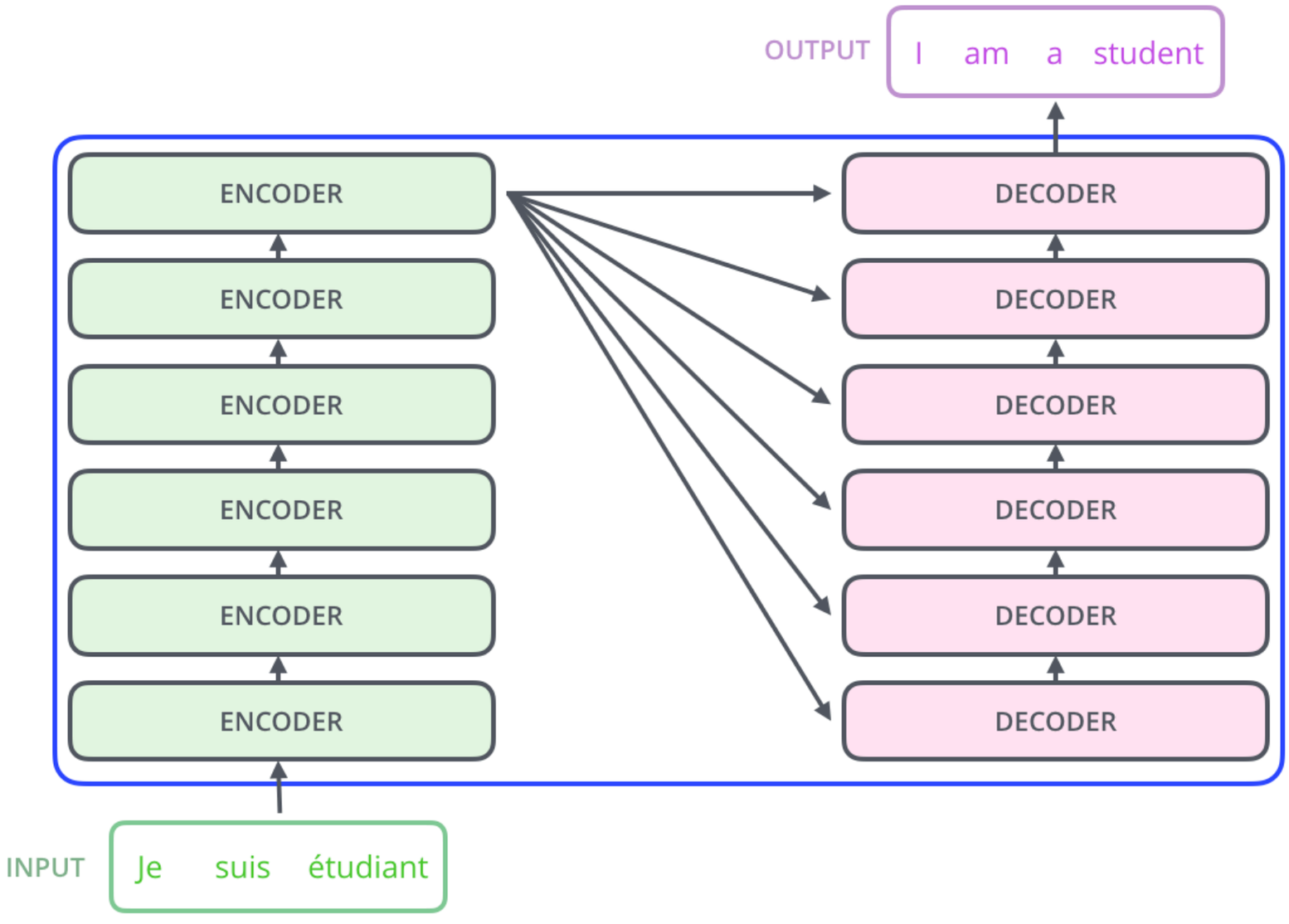


Figure from http://jalammar.github.io/illustrated-transformer/

Transformer

Encoder-decoder architecture

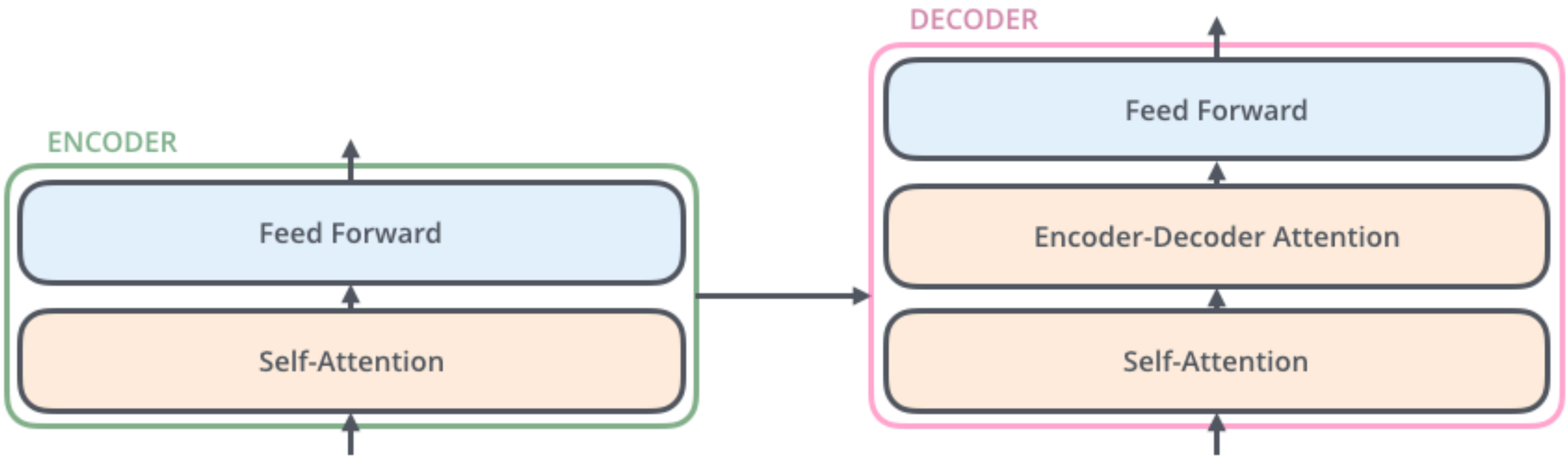


Figure from http://jalammar.github.io/illustrated-transformer/

Transformer

Transformer architecture detail

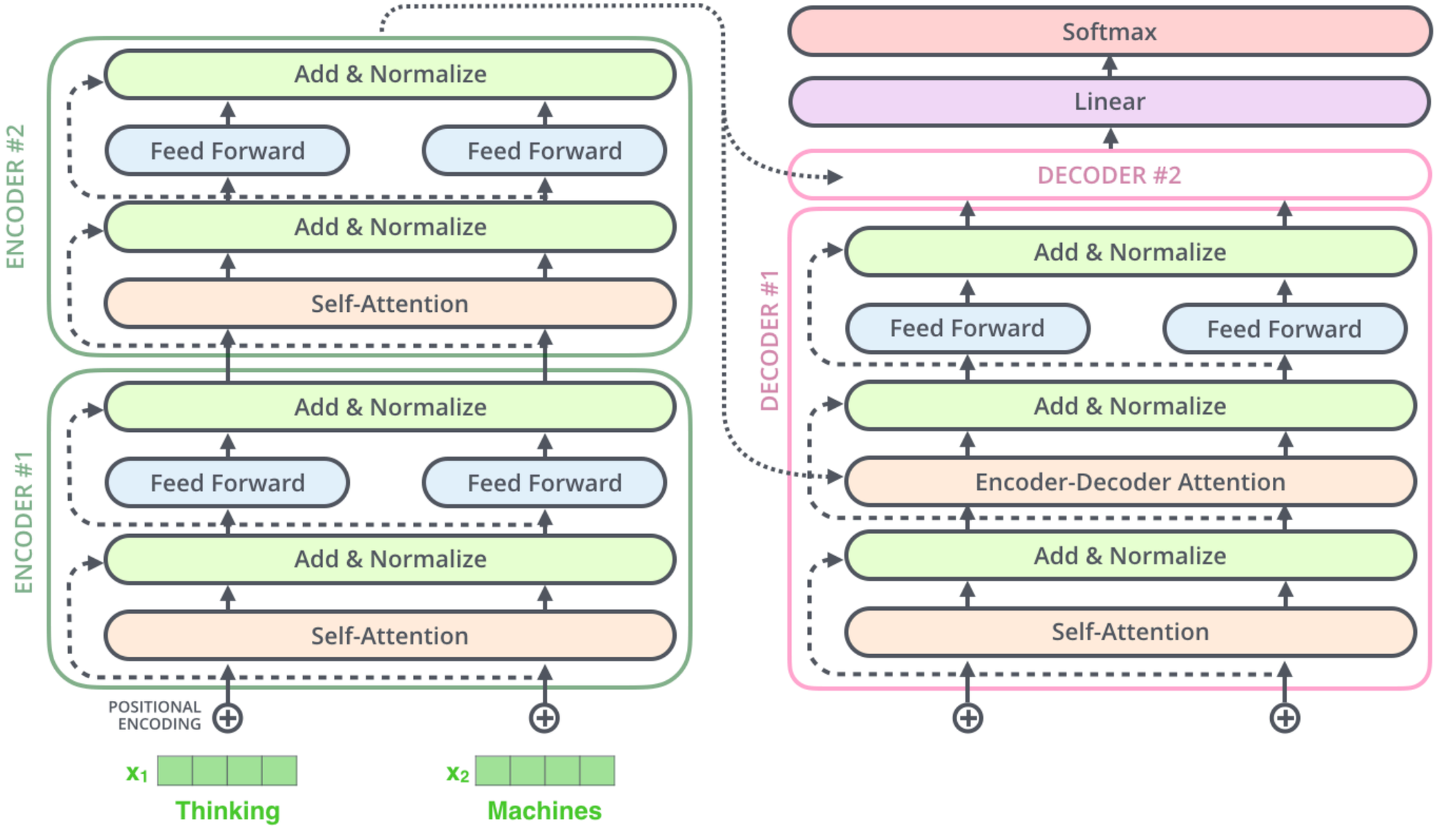


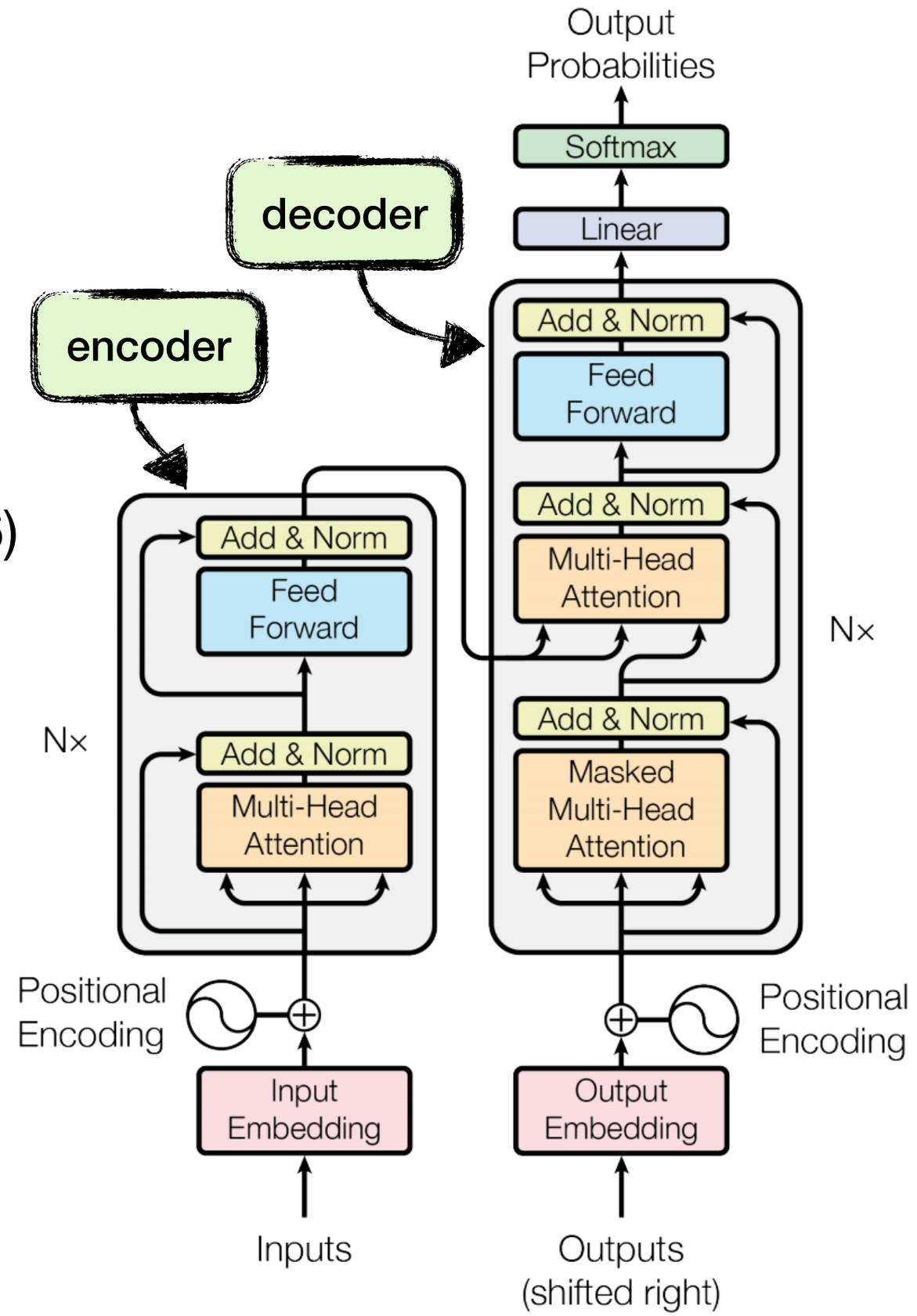
Figure from http://jalammar.github.io/illustrated-transformer/

Figure from Vaswani *et al*. (2017) *Attention Is All You Need*

Transformer

Architecture in more detail:

- Encoders and decoders **stacked** (x6)
- Simple **feed-forward NN** applied after attention at each position
- **Residual connections** [1] and **layer normalization** [2] applied
- **Multi-head attention**: Q, K and V projected with different weight matrices, attentions concatenated



[1] He *et al*. (2015) *Deep Residual Learning for Image Recognition*
[2] Ba *et al*. (2016) *Layer Normalization*

Transformer

Encoding and decoding (animation)

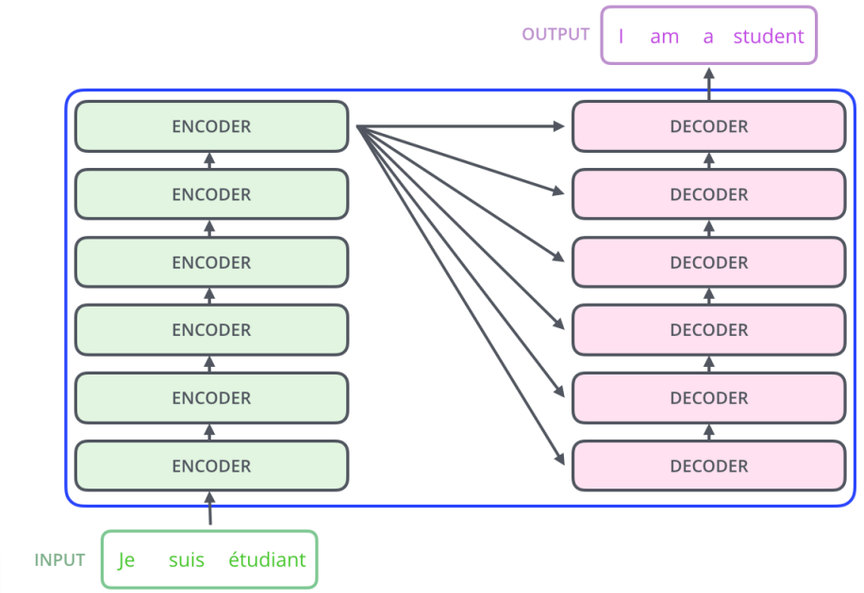


Illustration from https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Computational efficiency

RNNs with attention are quadratic (recall $e_{i,j} = \mathrm{a}(s_{i-1}, h_j)$) and hard to parallelize due to the sequential nature of the computation

Dot-product self-attention for each layer can be computed in parallel using matrix multiplications (Attention$(Q, K, V) =$ softmax$(QK^{\mathrm{T}}/s)V$)

	FLOPs	
Self-Attention	O(length^2 · dim)	= $4 \cdot 10^9$
RNN (LSTM)	O(length · dim^2)	= $16 \cdot 10^9$
Convolution	O(length · dim^2 · kernel_width)	= $6 \cdot 10^9$

length=1000 dim=1000 kernel_width=3

Figure from Vaswani and Huang (2019) Self-Attention For Generative Models

Summary

Attention can be generalized e.g. with the **query**, **key**, and **value** approach: query-key compatibility function determines value weights

RNN models where a decoder attends over encoder state still have **long-distance dependencies** within the encoder and decoder

Self-attention models attend to their own inputs, making connections **constant-length** and facilitating learning **contextual embeddings**

The **transformer** model uses dot-product **self-attention** without recurrence, using **positional encodings** for sequence information; the model is optimized for efficiency and parallelization

Transfer learning and neural language models

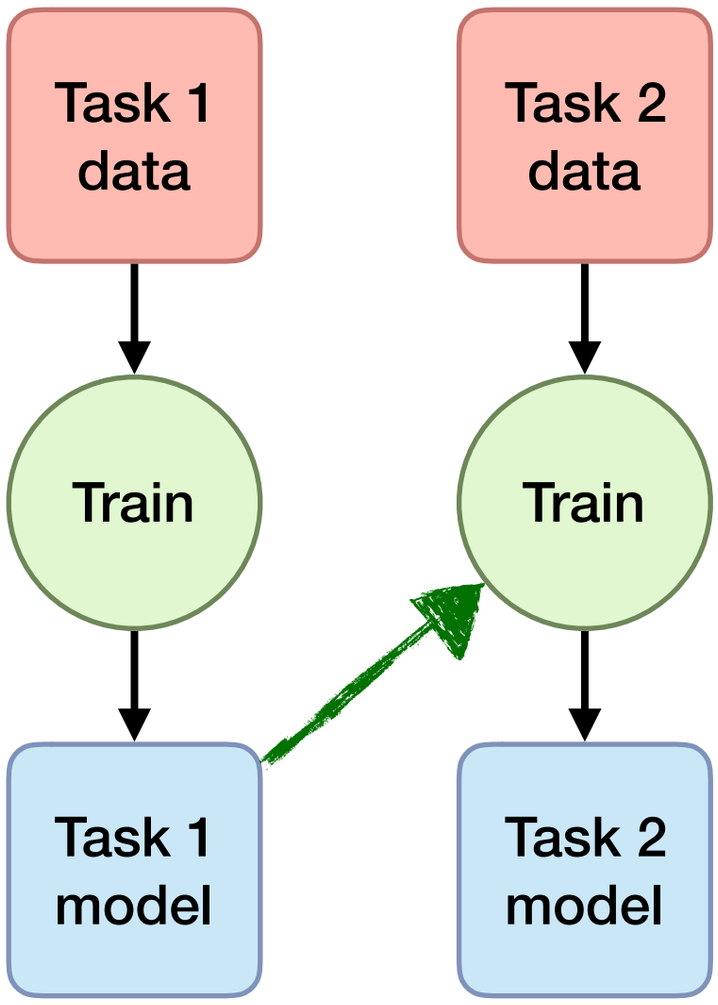
Transfer learning

Models commonly trained “from scratch”: at the start, the model knows nothing

(e.g. weights initialized randomly)

For related tasks, knowing one can support learning another

In **transfer learning**, knowledge from one task is used when learning another



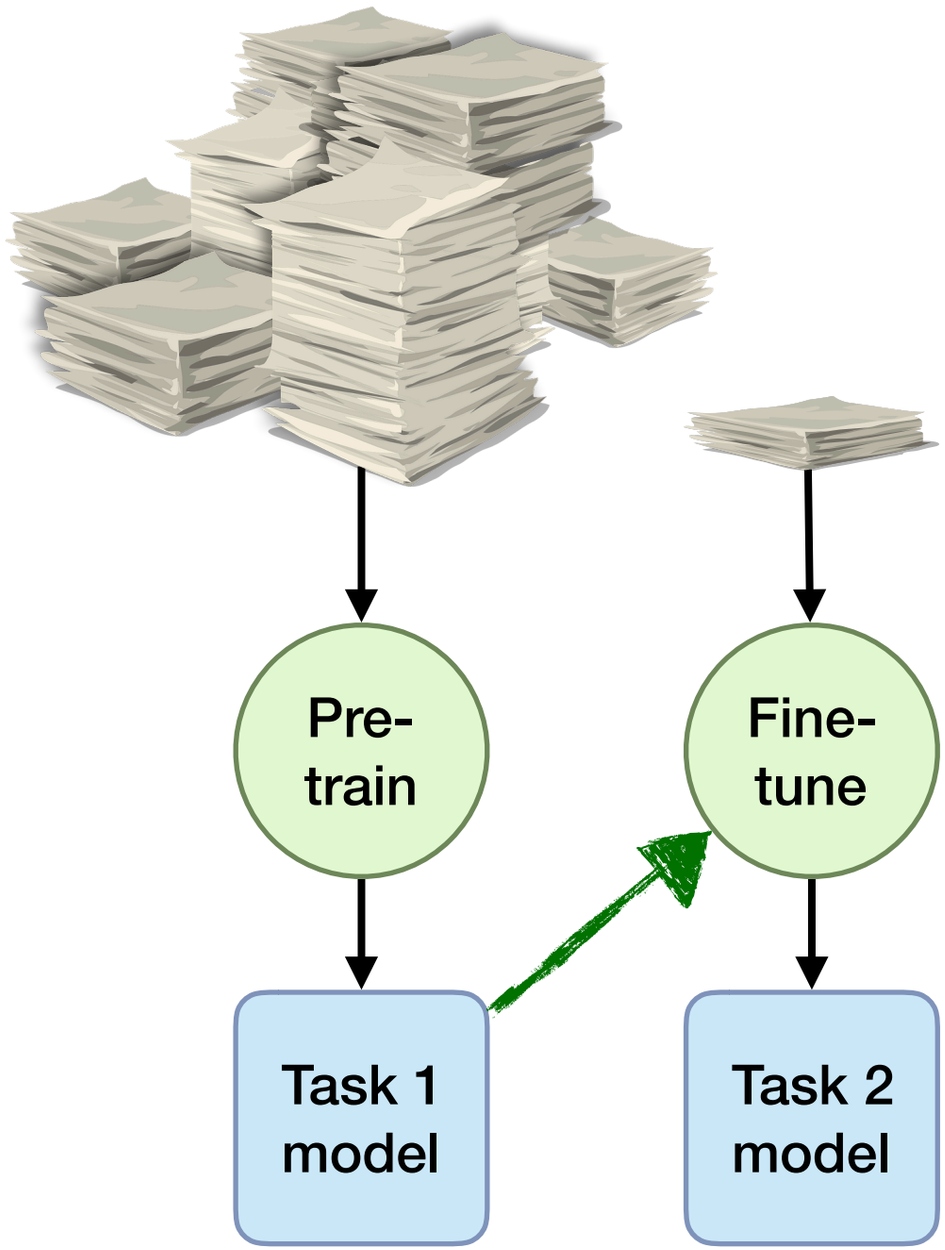
Transfer learning

For NLP, transfer from *unsupervised* tasks is particularly appealing:

Billions of words of **unannotated text** readily available on the internet

Annotated corpora are comparatively small and expensive to create

Deep learning methods are data-hungry



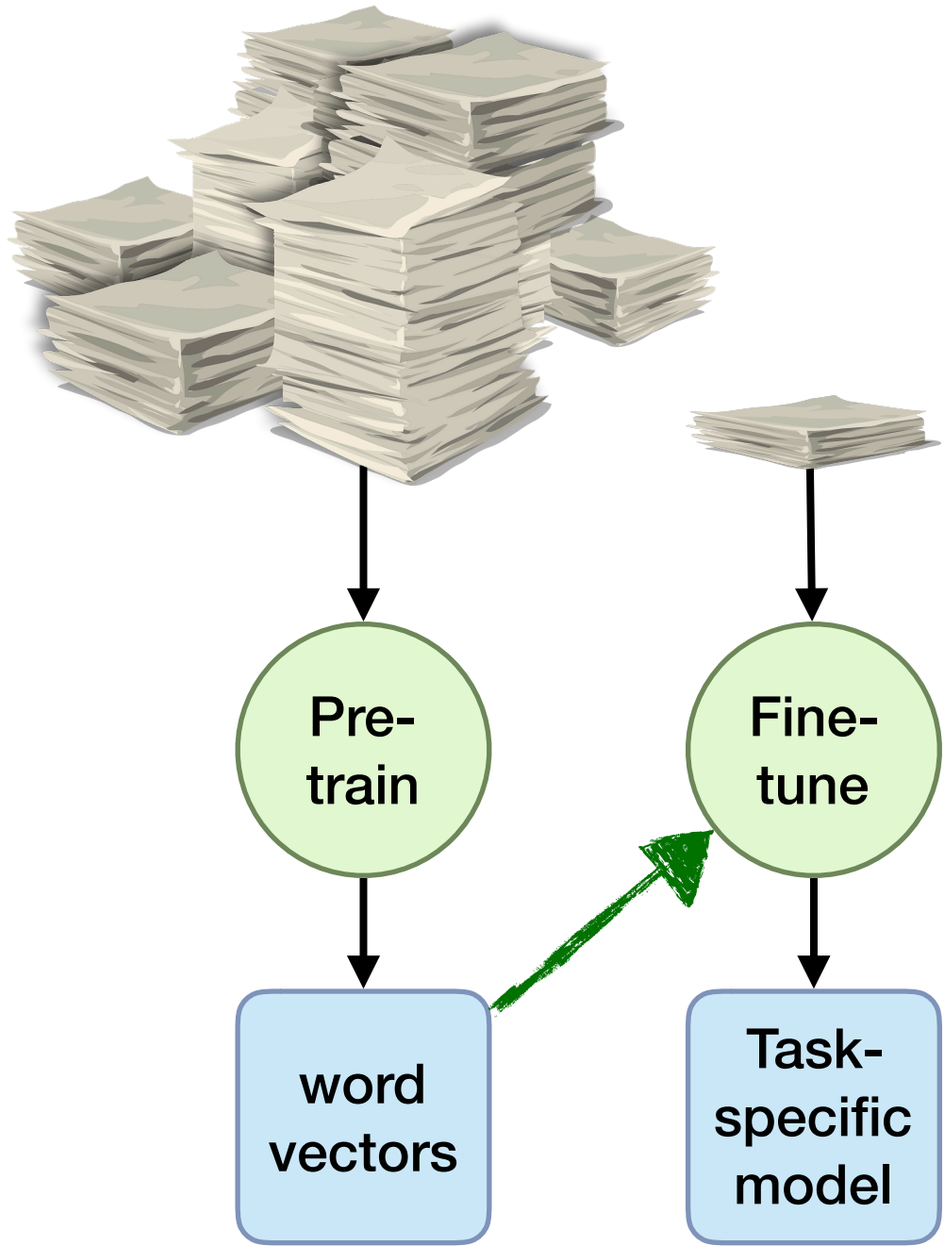
Transfer learning

Weight initialization is a straightforward way to implement transfer learning for NN models

A familiar example:

Task 1: using unannotated texts, train word2vec model, creating word vectors

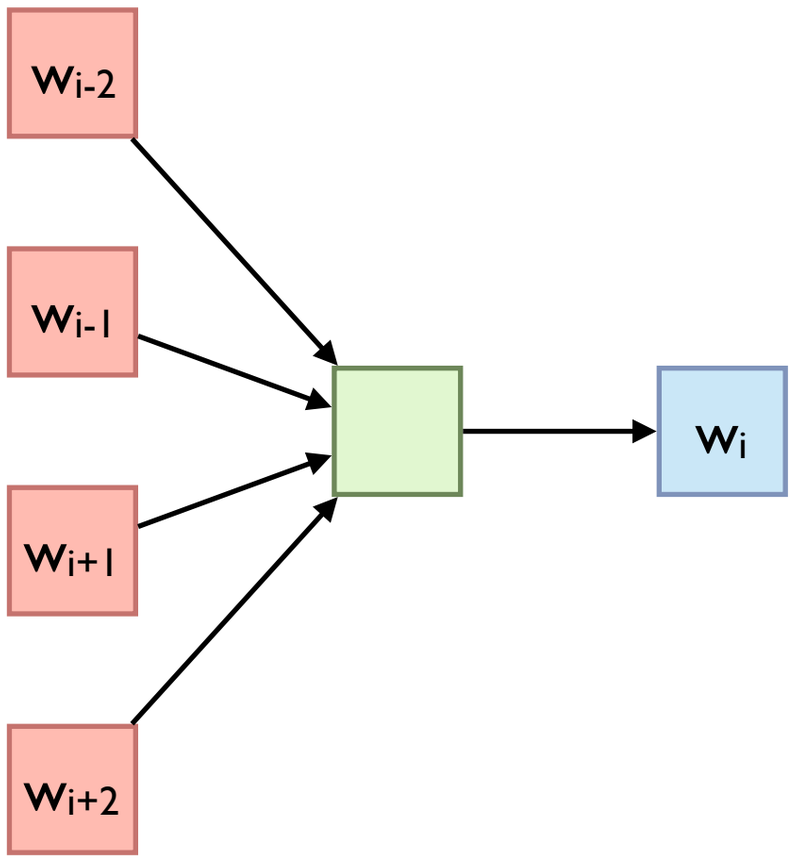
Task 2: initialize NN embedding weights from Task 1, train on task-specific data (e.g. NER)



Reminder: word2vec

Predict word given context

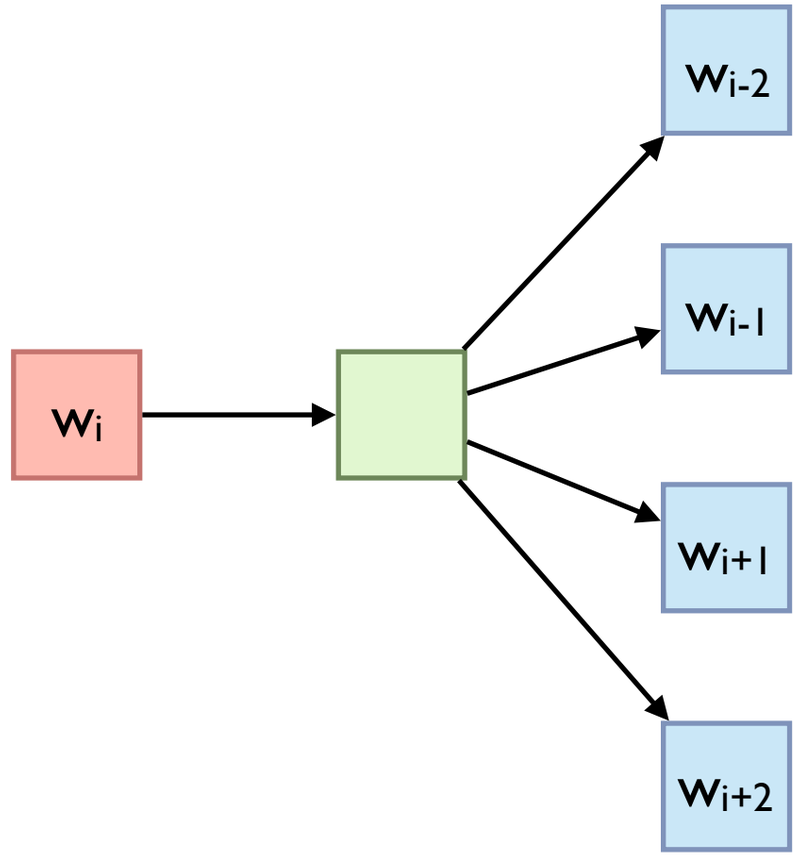
$P(w_i \mid w_{i-2}, w_{i-1}, \ldots)$



continuous bag-of-words

Predict context given word

$P(w_{i-2}, w_{i-1}, \ldots \mid w_i)$



skip-gram

Mikolov *et al*. (2013) *Efficient Estimation of Word Representations in Vector Space*

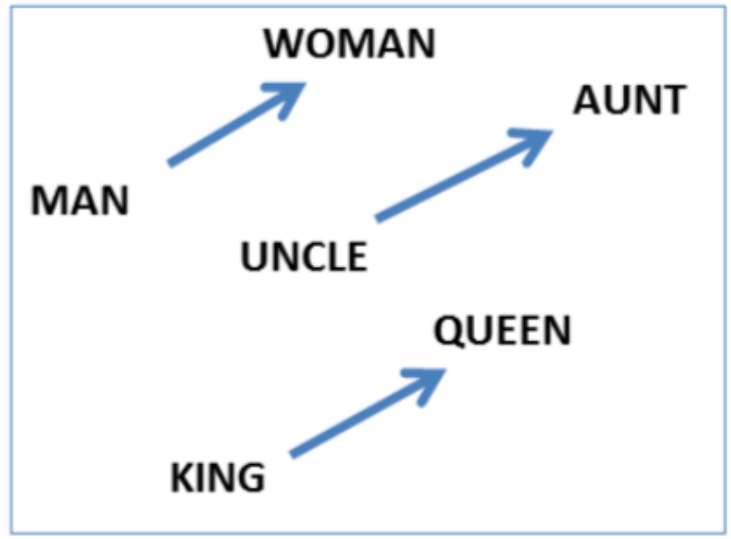
Reminder: word2vec

Word vectors created using “shallow” methods such as word2vec can capture many aspects of word meaning:

- Similarity: vectors of words with similar meanings have high cosine similarity

- Relations: differences between word vectors capture relationships between words

Initializing embedding weights from word vectors benefits many downstream tasks



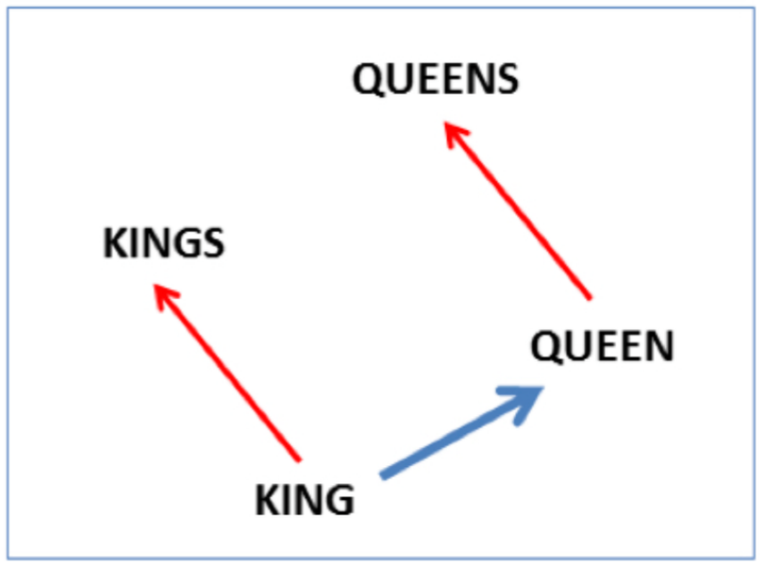


Figure from Mikolov *et al*. (2013) *Linguistic Regularities in Continuous Space Word Representations*

In code (keras + gensim)

```
…
emb = Embedding(…)(input_)
out = Dense(…)(emb)
```

```
weights = np.random.uniform(…)
…
wv = KeyedVectors.load_word2vec_format(“en.vec”)
for word, idx in vocabulary.items():
  weights[idx] = wv.get_vector(word)
…
emb = Embedding(…, weights=[weights])(input_)
out = Dense(…)(emb)
```

Word vectors: limitations

The implicit assumption that each word maps to exactly one vector representing its meaning is limiting in a number of cases, including

- **Polysemous words**: “money in the *bank*” vs. “river *bank*”
- **Multi-word expressions**: “*kick the bucket*”
- **Change in meaning**: “*gay*” means “cheerful” in old texts (~1900)
- **Multilingual models**: English “*on*” differs from Finnish “*on*”

→ What’s really needed are representations of **meaning in context**

Language modeling

Recall standard language modeling objective:

$$P(w_i \mid w_{i\text{-}1}, w_{i\text{-}2}, \ldots)$$

Bigram model (Markov assumption)

$$P(w_i \mid w_{i\text{-}1}, w_{i\text{-}2}, \ldots) \approx P(w_i \mid w_{i\text{-}1})$$

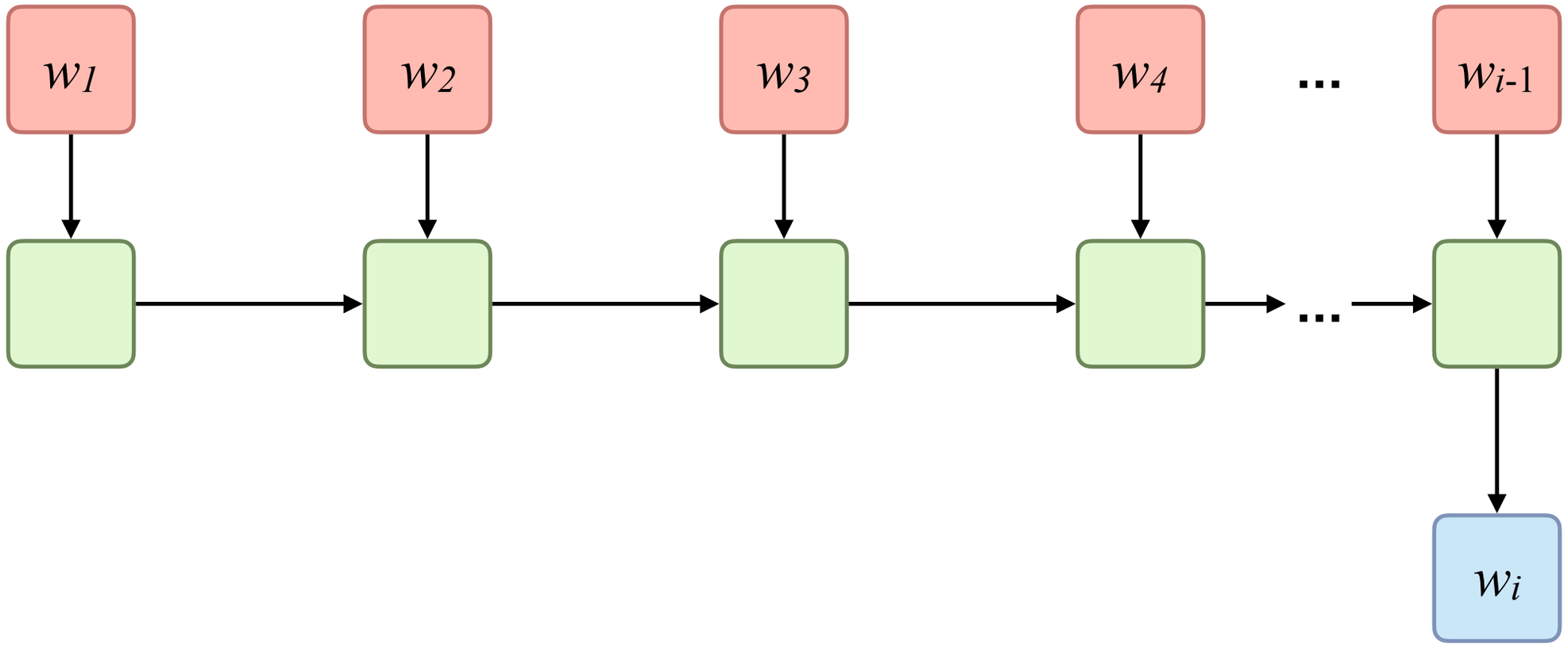
Maximum-likelihood estimate (C is count)

$$P(w_i \mid w_{i\text{-}1}) = C(w_{i\text{-}1}\, w_i) \,/\, C(w_{i\text{-}1})$$

A bigram model can tell you e.g. that P(“dog” | “the”) > P(“the” | “the”) as “the dog” is (hopefully) more common in text than “the the”

Language modeling

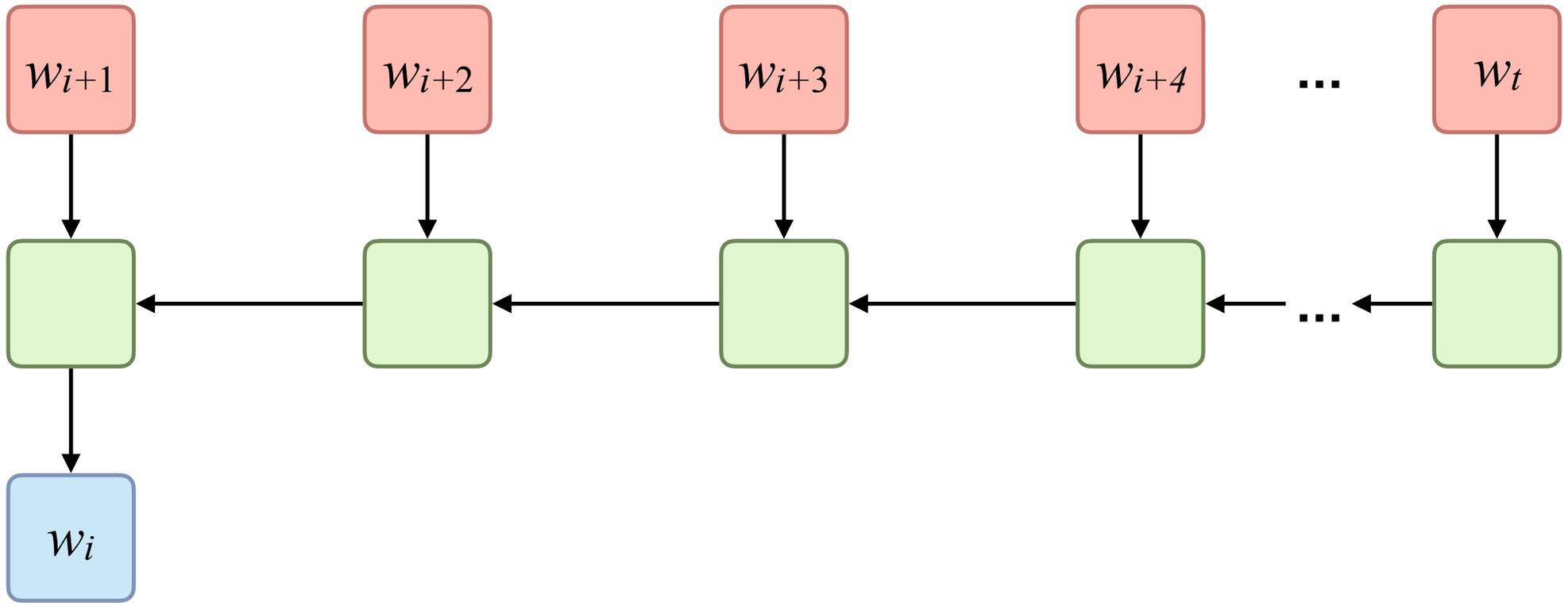
We can straightforwardly address the objective P(w_i | w_{i-1}, w_{i-2}, …) using neural networks, e.g. simple RNN model



(This is just a word-level version of our character generation model)

Language modeling

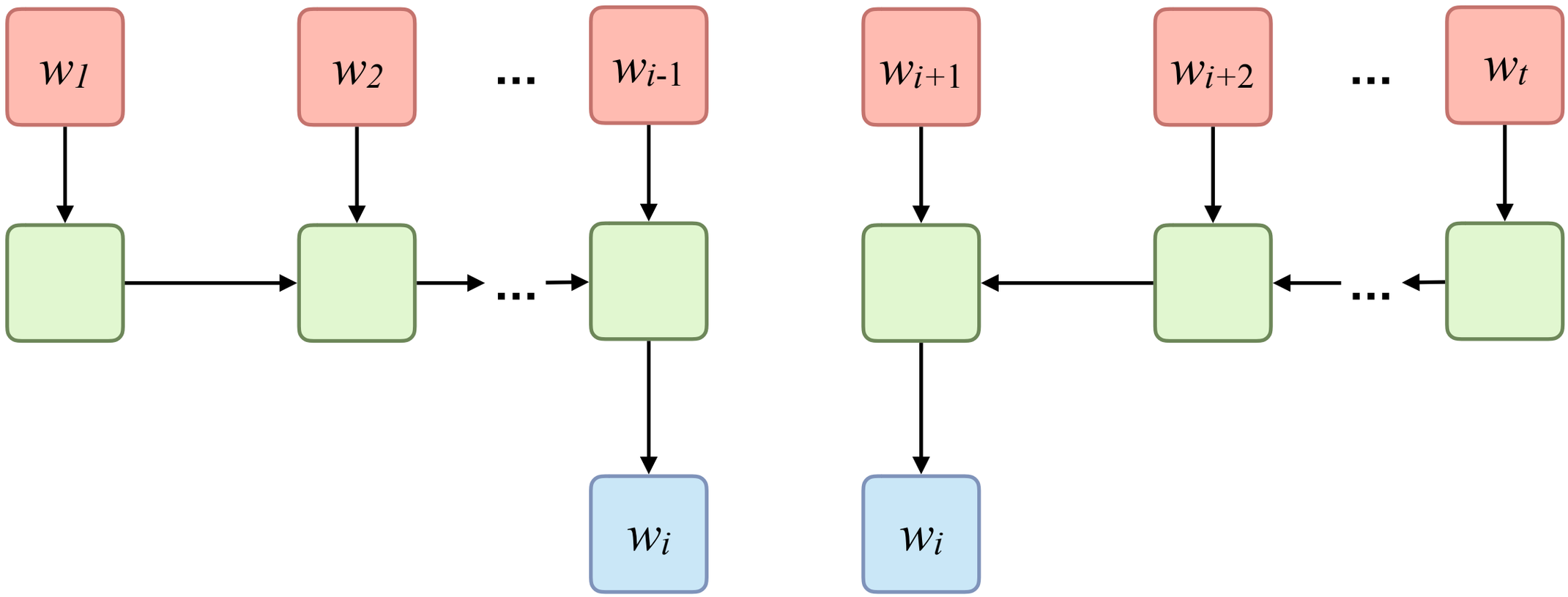
We can similarly model P(w_i | w_{i+1}, w_{i+2}, …) with a reverse RNN



(Also note no Markov assumption!)

Language modeling

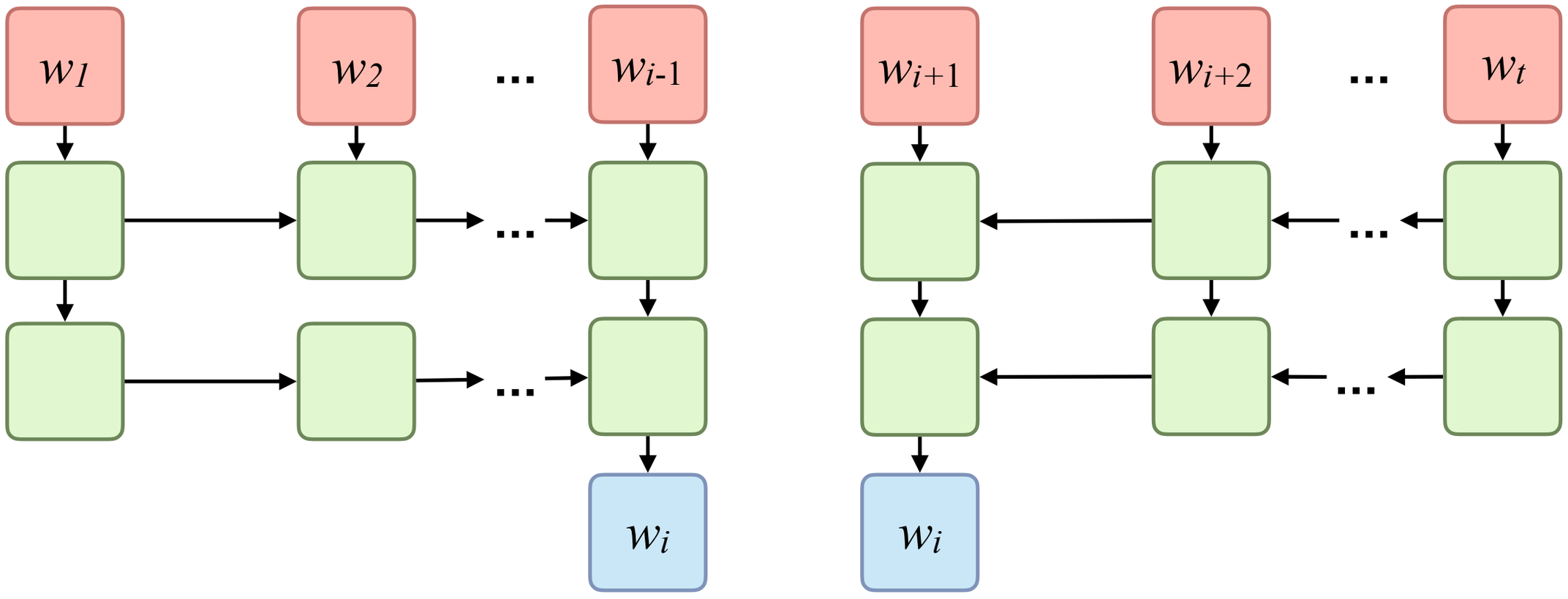
Combining the two, we can address P(w_i | $w_1, w_2, \ldots, w_{i-1}, w_{i+1}, w_{i+2}, \ldots$)



(Just need to somehow combine values provided by the two RNNs)

Language modeling

RNN models can be stacked to give deep RNNs …



and the RNN cells can have any design (vanilla, LSTM, GRU)

ELMo

Embeddings from Language Models

Deep language model using forward and backward LSTMs

Forward and backward RNN states concatenated to create contextual word representation

Pre-trained on 1B word unannotated corpus of English

In combination with supervised NLP tasks, advanced SOTA in tasks including question answering, NER and sentiment analysis

Peters *et al*. (2018) *Deep contextualized word representations*

ELMo

Contextualized representations can differentiate between word senses

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

Peters *et al*. (2018) *Deep contextualized word representations*

GPT

Generative Pre-training Transformer

Deep *forward* language model using transformer decoding

→ no need for “future” inputs, emphasis on generation

Pre-trained on ~1B word BooksCorpus (English)

Fine-tuning for supervised NLP tasks, advanced SOTA

Larger follow-up model GPT-2 was model “too dangerous to release”

Radford *et al*. (2018) *Improving Language Understanding by Generative Pre-Training*

GPT-2 text generation

PROMPT (HUMAN-WRITTEN)

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

MODEL COMPLETION (MACHINE-WRITTEN)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

…

Example from https://openai.com/blog/better-language-models/

BERT

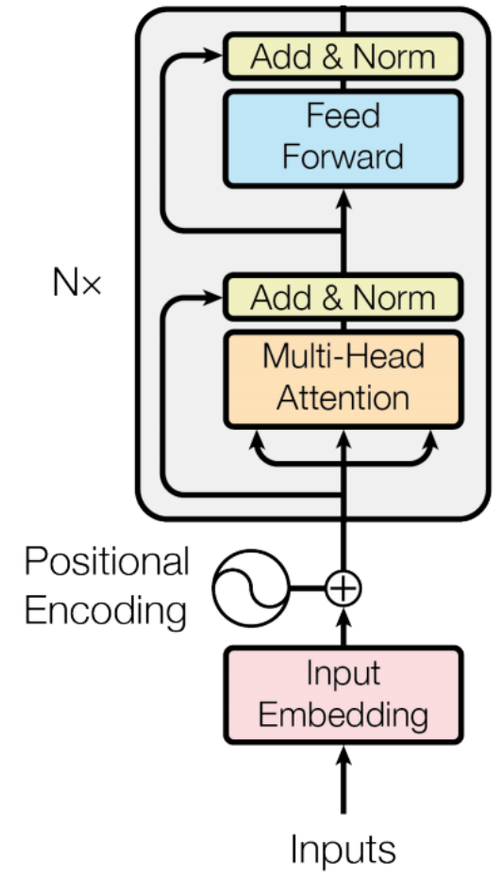
Bidirectional Encoder Representations from Transformers

Deep *bidirectional* language model using transformer encoder

Trained using **masked language modeling** and **next sentence prediction** objectives

English model pretrained on 3B words (Wikipedia + BooksCorpus)

Fine-tuning for various language understanding tasks showed results *surpassing human performance*



Devlin *et al*. (2019) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

Summary

In **transfer learning**, knowledge from one task is used when learning another; word embedding weight initialization is a familiar example

Neural language models trained on large unannotated corpora can be used to create **contextualized representations** of meaning

Transfer learning using **deep language models** such as ELMo, GPT and BERT have substantially advanced the state of the art in many NLP tasks, exceeding some estimates of human performance

Related work

Melamud *et al*. (2016) *context2vec: Learning Generic Context Embedding with Bidirectional LSTM*

McCann *et al*. (2018) Learned in Translation: Contextualized Word Vectors [CoVe]

Howard and Ruder (2018) *Universal Language Model Fine-tuning for Text Classification* [ULMFiT]

Radford *et al*. (2019) *Language Models are Unsupervised Multitask Learners* [GPT-2]

Dai *et al.* (2019) Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Liu *et al*. (2019) RoBERTa: *A Robustly Optimized BERT Pretraining Approach*

Lan *et al*. (2019) ALBERT: A Lite BERT for Self-supervised Learning of Language Representations

Clark *et al*. (2020) *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*

Resources

Google AI Blog: Transformer: A Novel Neural Network Architecture for Language Understanding
https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Alexander Rush: The Annotated Transformer
https://nlp.seas.harvard.edu/2018/04/03/attention.html

Jay Alammar: The Illustrated Transformer
http://jalammar.github.io/illustrated-transformer/

Google AI: Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing
https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

OpenAI: Better Language Models and Their Implications
https://openai.com/blog/better-language-models