Today's Topics

- Neural word embeddings
- Motivation: machine neural translation for long sentences
- Decoder: attention
- Transformer overview
- Self-attention

Slides Thanks to Dana Gurari

Converting Text to Vectors

1. Tokenize training data; convert data into sequence of tokens (e.g., data ->“This is tokening”)
2. Learn vocabulary
3. Encode data as vectors

Two common approaches:

Character Level

[T] [h] [i] [s] [i] [s] [t] [o] [k] [e] [n] [i] [z] [i] [n] [g] [.]

Word Level

[This] [is] [tokenizing] [.]

https://nlpiation.medium.com/how-to-use-huggingfaces-transformers-pre-trained-tokenizers-e029e8d6d1fa

Converting Text to Vectors

1. Tokenize training data
2. Learn vocabulary by identifying all unique tokens in the training data
3. Encode data as vectors

Two common approaches:

Character Level

Token	a	b	c	***	0	1	***	!	@	***
Index	1	2	3	***	27	28	***	119	120	***

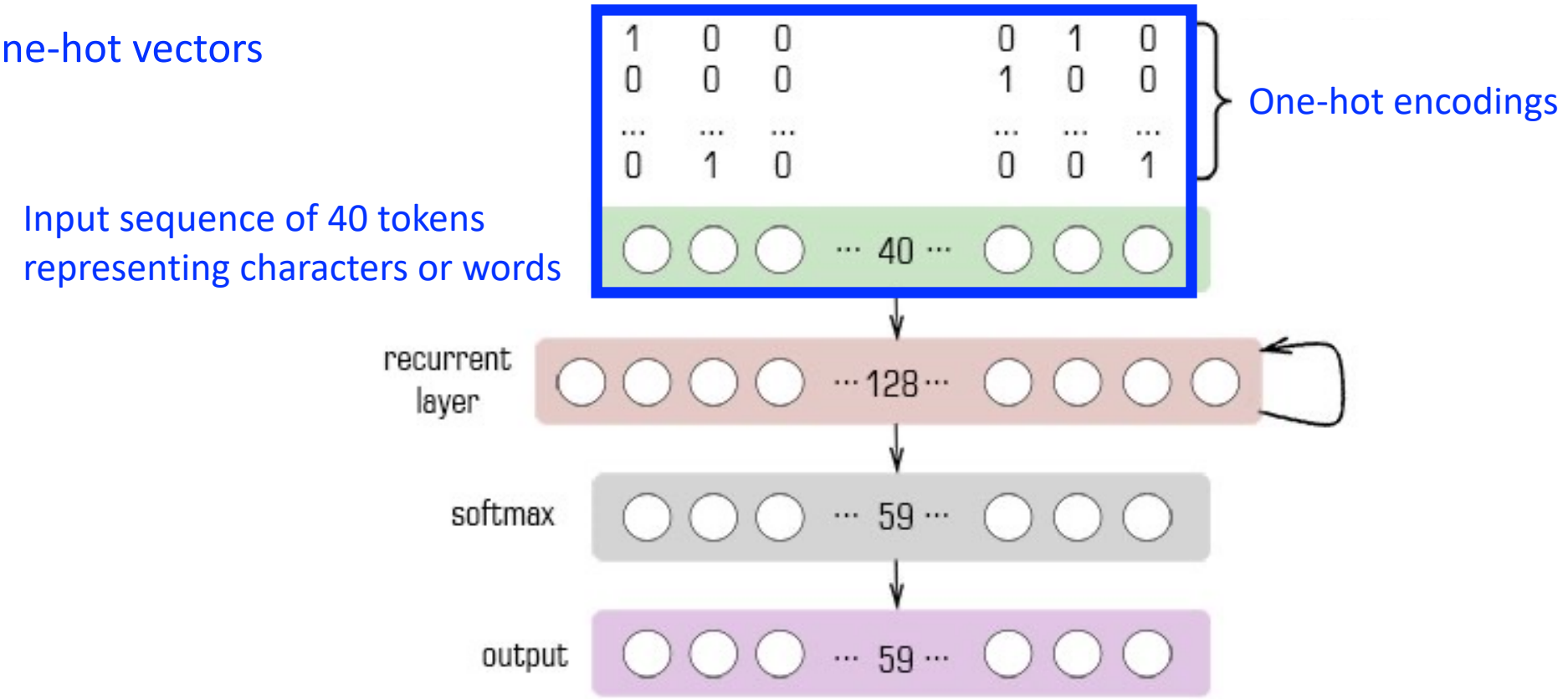
Word Level

Token	a	an	at	***	bat	ball	***	zipper	zoo	***
Index	1	2	3	***	527	528	***	9,842	9,843	***

https://nlpiation.medium.com/how-to-use-huggingfaces-transformers-pre-trained-tokenizers-e029e8d6d1fa

Converting Text to Vectors

1. Tokenize training data
2. Learn vocabulary by identifying all unique tokens in the training data
3. Encode data as one-hot vectors



https://github.com/DipLernin/Text_Generation

Converting Text to Vectors

What are the pros and cons for using word tokens instead of character tokens?

Character Level

Token	a	b	c	***	0	1	***	!	@	***
Index	1	2	3	***	27	28	***	119	120	***

Word Level

Token	a	an	at	***	bat	ball	***	zipper	zoo	***
Index	1	2	3	***	527	528	***	9,842	9,843	***

- Pros: length of input/output sequences is shorter, simplifies learning semantics
- Cons: “UNK” word token needed for out of vocabulary words; vocabulary can be large

https://nlpiation.medium.com/how-to-use-huggingfaces-transformers-pre-trained-tokenizers-e029e8d6d1fa

Converting Text to Vectors

Character Level

Token	a	b	c	***	0	1	***	!	@	***
Index	1	2	3	***	27	28	***	119	120	***

Word Level

Token	a	an	at	***	bat	ball	***	zipper	zoo	***
Index	1	2	3	***	527	528	***	9,842	9,843	***

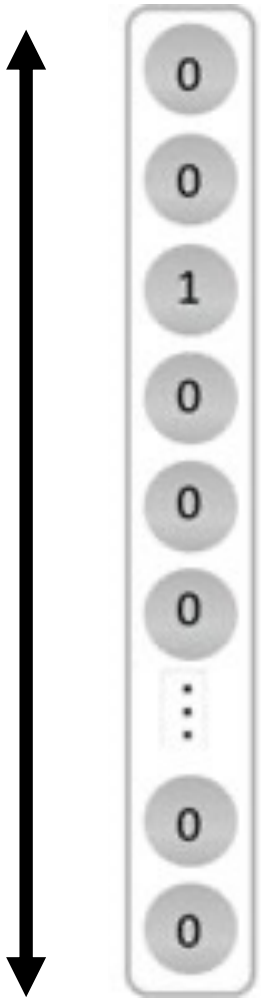
Word level representations are more commonly used

https://nlpiation.medium.com/how-to-use-huggingfaces-transformers-pre-trained-tokenizers-e029e8d6d1fa

Problems with One-Hot Encoding Words?

Dimensionality = vocabulary size

e.g., English has ~170,000 words
with ~10,000 commonly used words

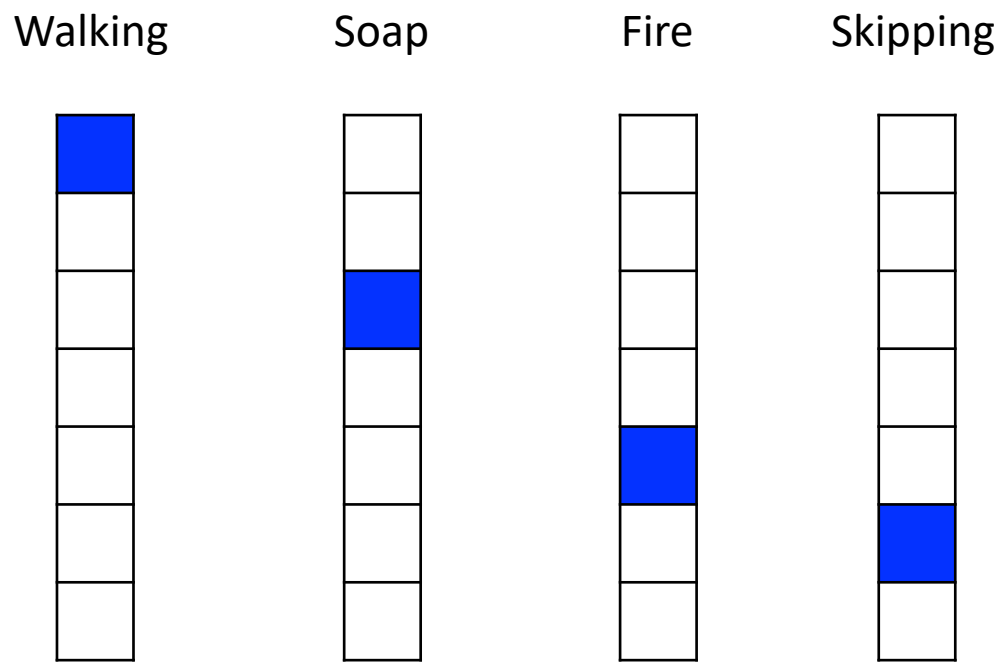


- Huge memory burden
- Computationally expensive

Kamath, Liu, and Whitaker. Deep Learning for NLP and Speech Recognition. 2019.

Limitation of One-Hot Encoding Words

- No notion of which words are similar, yet such understanding can improve generalization
  - e.g., “walking”, “running”, and “skipping” are all suitable for “He was ____ to school.”

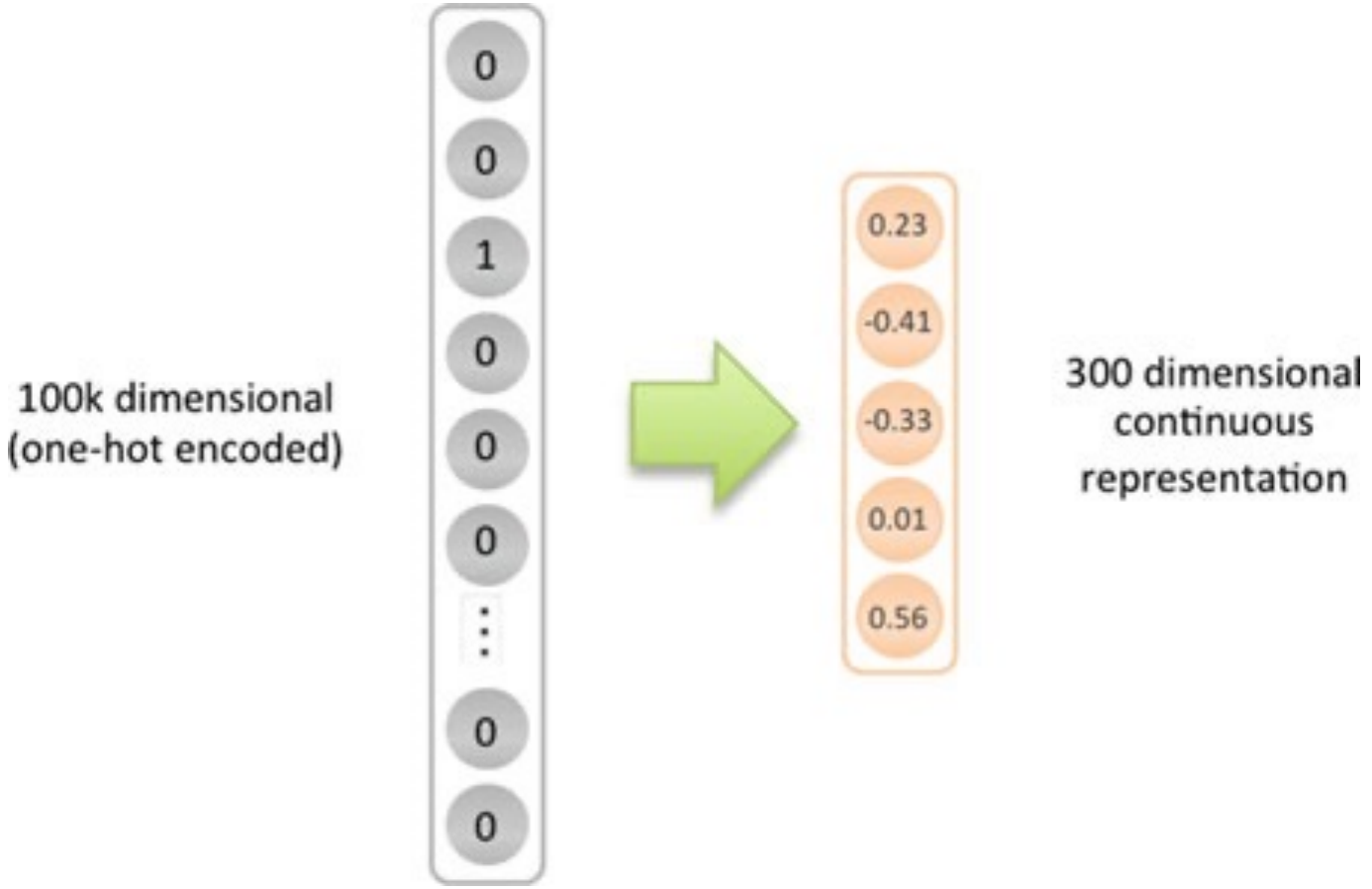


The distance between all words is equal!

Today's Topics

- Introduction to natural language processing
- Text representation
- Neural word embeddings
- Programming tutorial

Idea: Represent Each Word Compactly in a Space Where Vector Distance Indicates Word Similarity



Kamath, Liu, and Whitaker. Deep Learning for NLP and Speech Recognition. 2019.

Inspiration: Distributional Semantics

"The distributional hypothesis says that the meaning of a word is derived from the context in which it is used, and words with similar meaning are used in similar contexts."

- Origins: Harris in 1954 and Firth in 1957

Kamath, Liu, and Whitaker. Deep Learning for NLP and Speech Recognition. 2019.

Inspiration: Distributional Semantics

"The distributional hypothesis says that the meaning of a word is derived from the context in which it is used, and words with similar meaning are used in similar contexts."

Kamath, Liu, and Whitaker. Deep Learning for NLP and Speech Recognition. 2019.

Inspiration: Distributional Semantics

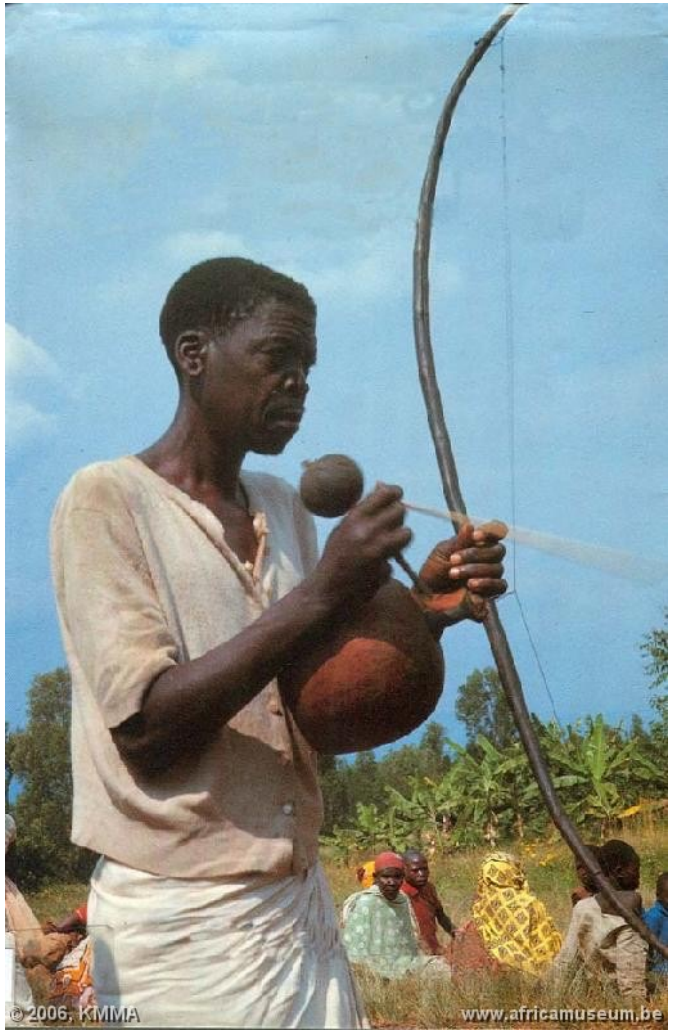
- What is the meaning of berimbau based on context?

  Background music from a berimbau offers a beautiful escape.

  Many people danced around the berimbau player.

  I practiced for many years to learn how to play the berimbau.

- Idea: context makes it easier to understand a word's meaning



[Adapted from slides by Lena Voita]

https://capoeirasongbook.wordpress.com/instruments/berimbau/

Inspiration: Distributional Semantics

"The distributional hypothesis says that the meaning of a word is derived from the context in which it is used, and words with similar meaning are used in similar contexts."

Kamath, Liu, and Whitaker. Deep Learning for NLP and Speech Recognition. 2019.

Inspiration: Distributional Semantics

- What other words could fit into these context?

1. Background music from a _____ offers a beautiful escape.
2. Many people danced around the _____ player.
3. I practiced for many years to learn how to play the _____.

Hypothesis is that words with similar row values have similar meanings

	1.	2.	3.
Berimbau	1	1	1
Soap	0	0	0
Fire	0	0	0
Guitar	1	1	1

Contexts

1 if a word can appear in the context
0 otherwise

[Adapted from slides by Lena Voita]

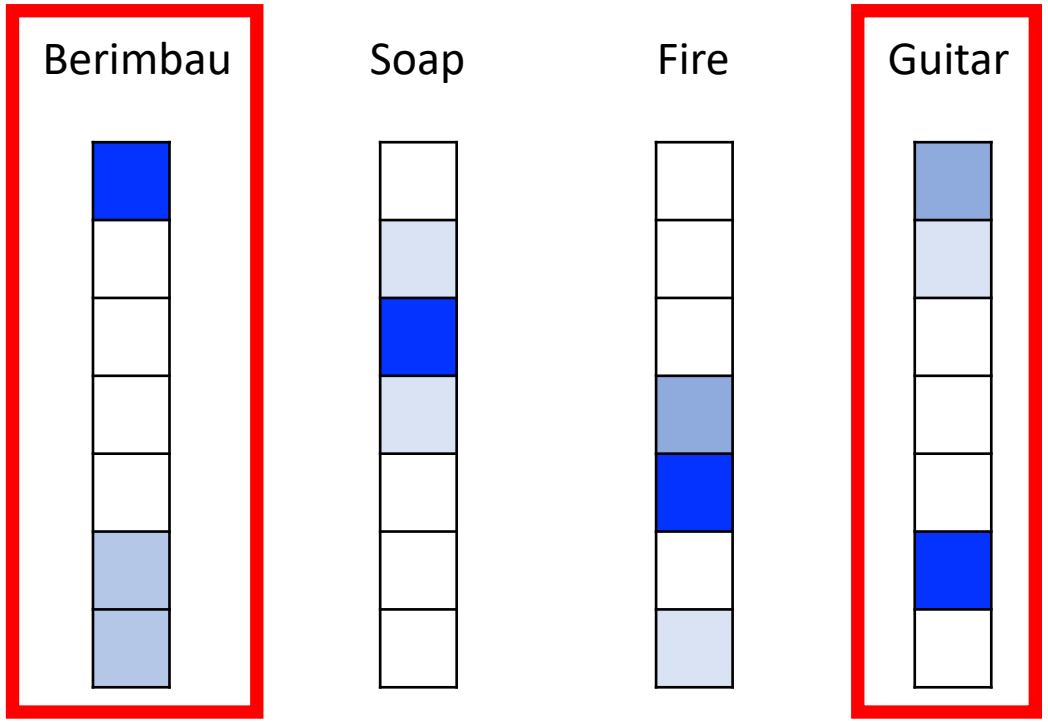
Inspiration: Distributional Semantics

"The distributional hypothesis says that the meaning of a word is derived from the context in which it is used, and words with similar meaning are used in similar contexts."

Kamath, Liu, and Whitaker. Deep Learning for NLP and Speech Recognition. 2019.

Approach

- Learn a dense (lower-dimensional) vector for each word by characterizing its **context**, which inherently will reflect similarity/differences to other words



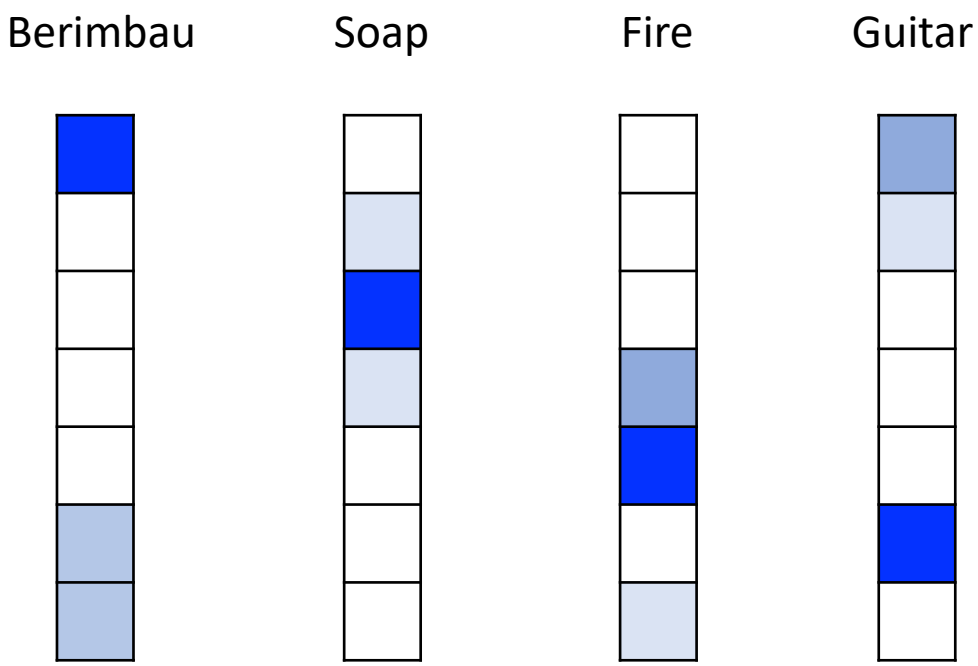
Berimbau and guitar are the closest word pair

The distance between each pair of words differs!

Note: many ways to measure distance (e.g., cosine distance)

Approach

- Learn a dense (lower-dimensional) vector for each word by characterizing its **context**, which inherently will reflect similarity/differences to other words

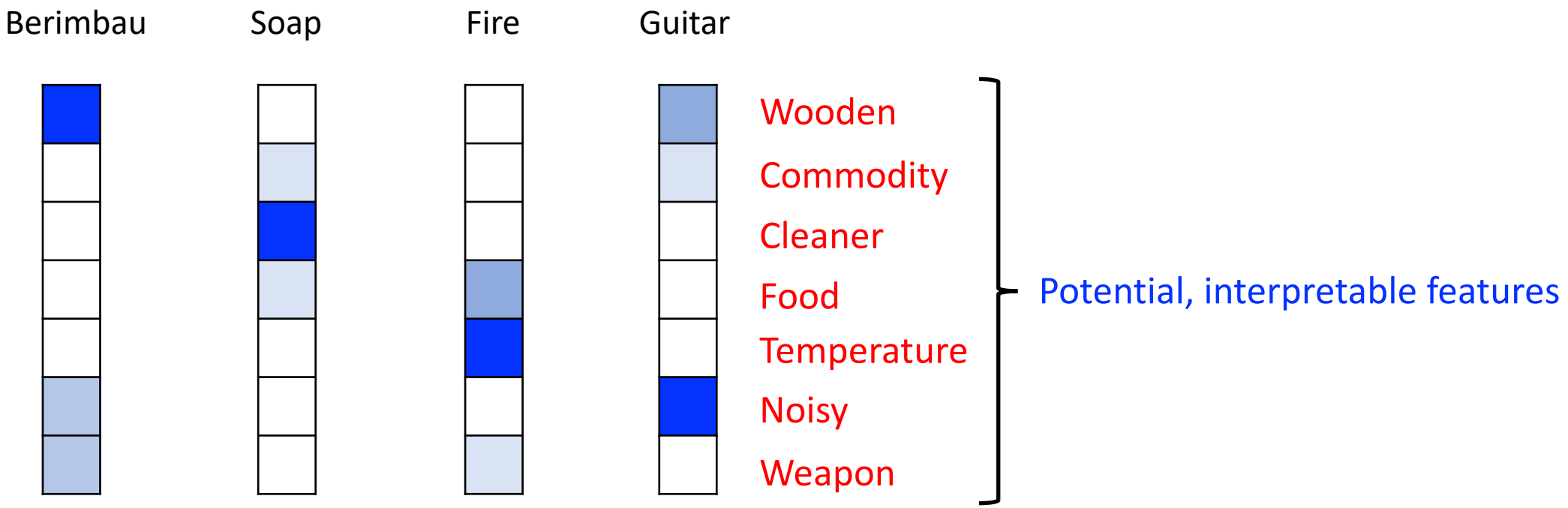


We embed words in a shared space so they can be compared with a few features

What features would discriminate these words?

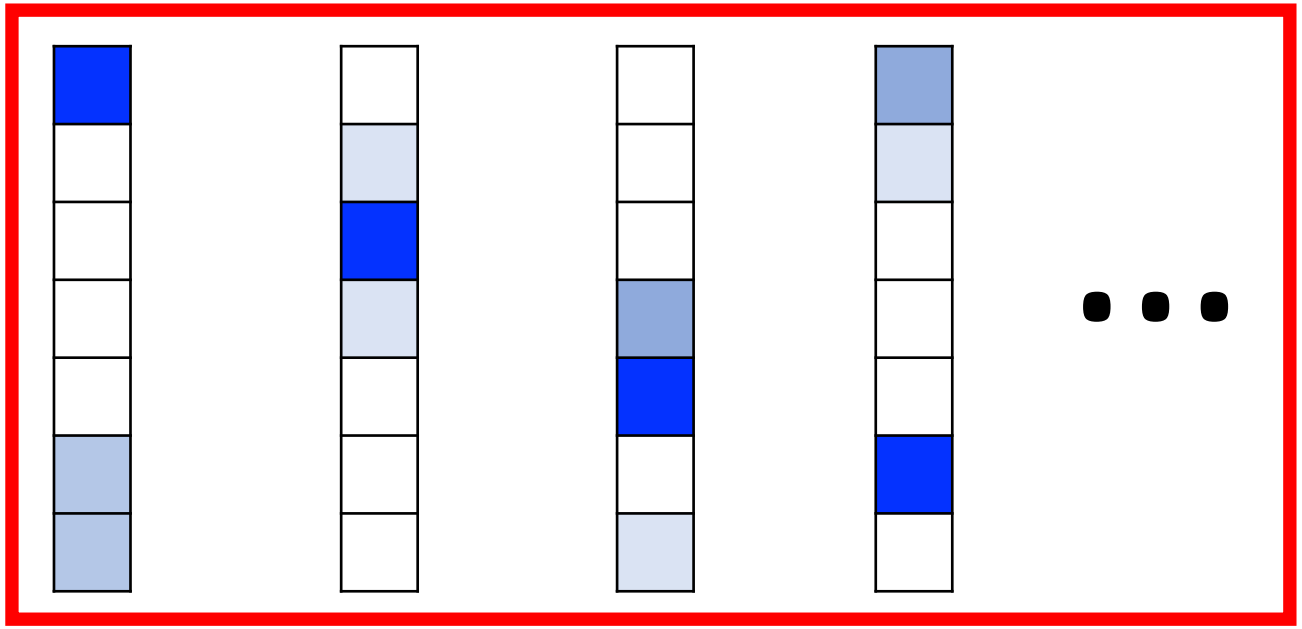
Approach

- Learn a dense (lower-dimensional) vector for each word by characterizing its **context**, which inherently will reflect similarity/differences to other words



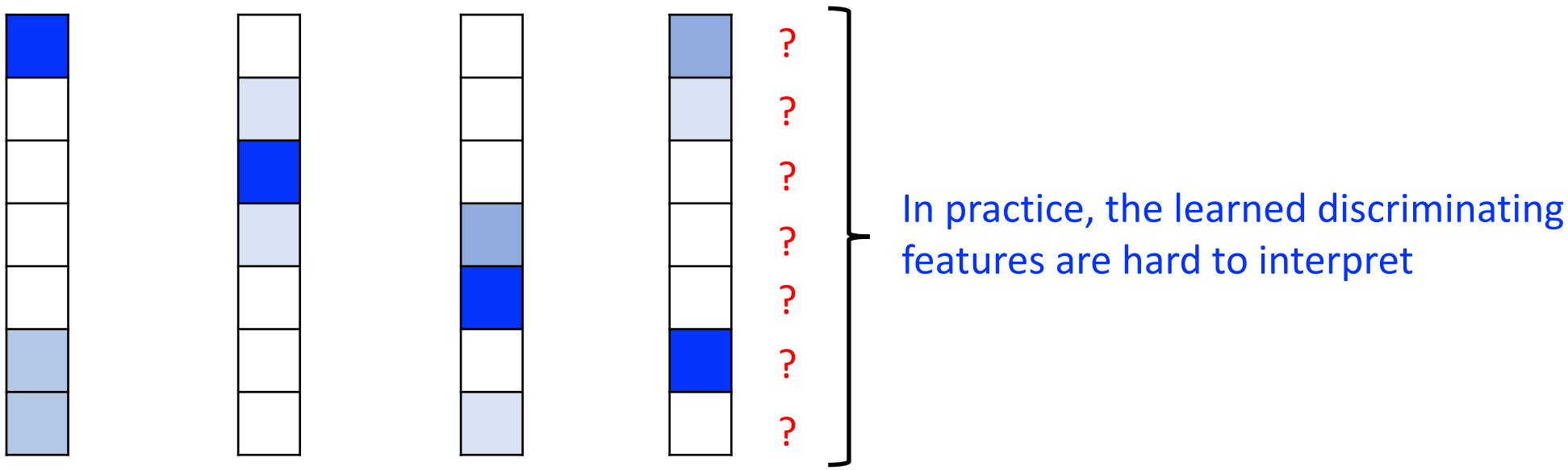
Approach: Learn Word Embedding Space

- An **embedding space** represents a finite number of words, decided in training
- A **word embedding** is represented as a vector indicating its context
- The dimensionality of all word embeddings in an embedding space match
  - What is the dimensionality for the shown example?



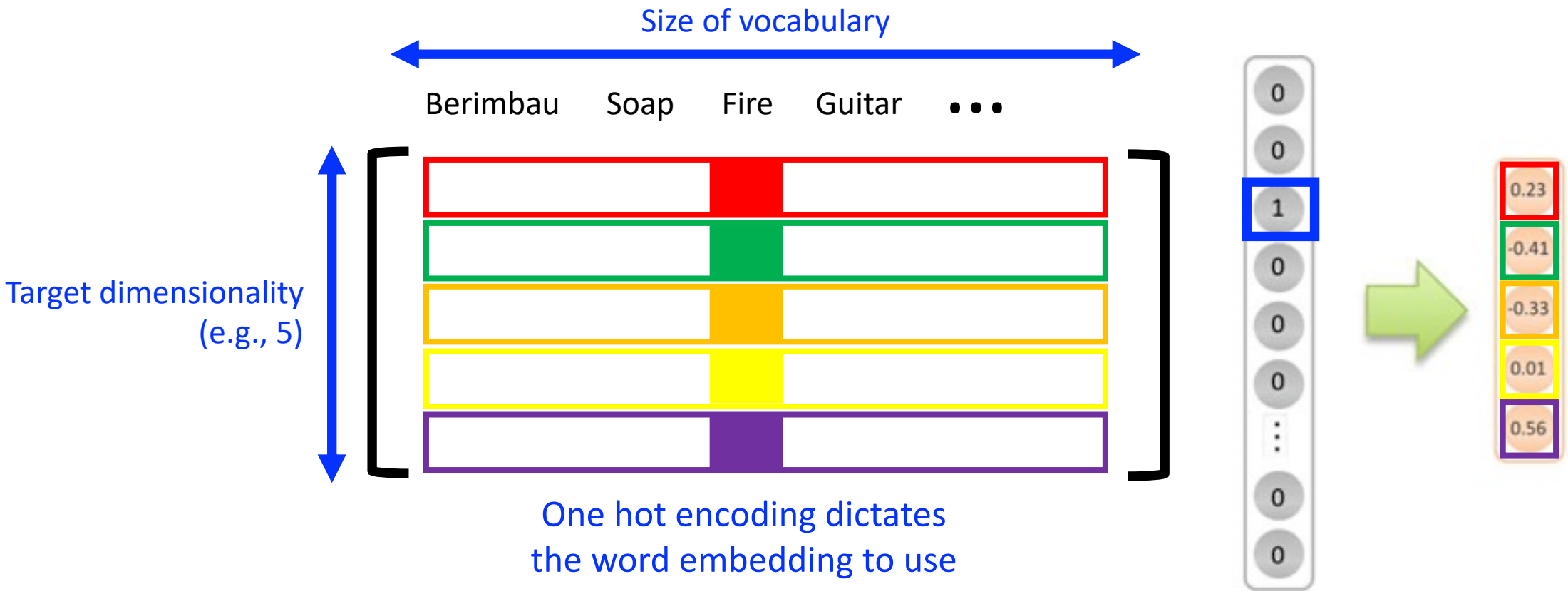
Approach: Learn Word Embedding Space

- An embedding space represents a finite number of words, defined in training
- A word embedding is represented as a vector indicating its context
- The dimensionality of all word embeddings in an embedding space match



Embedding Matrix

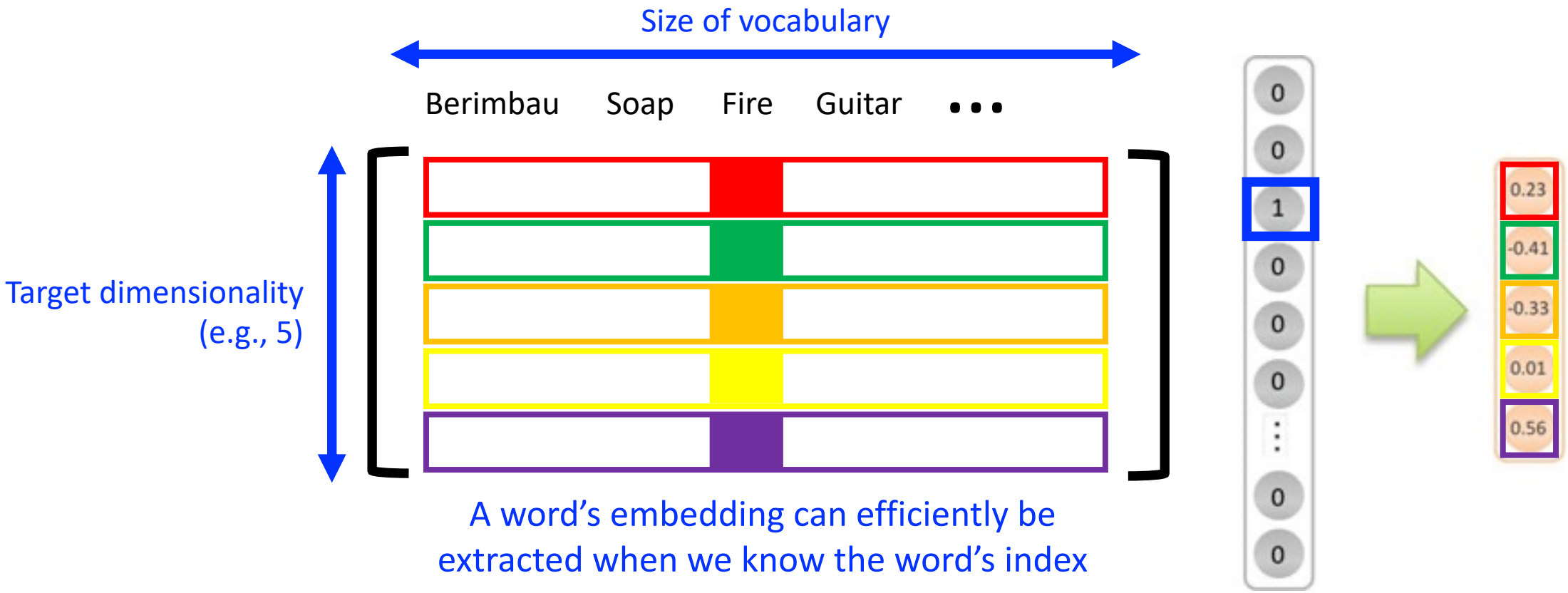
- The embedding matrix converts an input word into a dense vector



Kamath, Liu, and Whitaker. Deep Learning for NLP and Speech Recognition. 2019.

Embedding Matrix

- It converts an input word into a dense vector



Kamath, Liu, and Whitaker. Deep Learning for NLP and Speech Recognition. 2019.

Popular Word Embeddings

- Bengio method
- Word2vec (skip-gram model)
- And more…

Popular Word Embeddings

- Bengio method
- Word2vec (skip-gram model)
- And more…

Idea: Learn Word Embeddings That Help Predict Viable Next Words

e.g.,

1. Background music from a _____
2. Many people danced around the _____
3. I practiced for many years to learn how to play the _____

Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Task: Predict Next Word Given Previous Ones

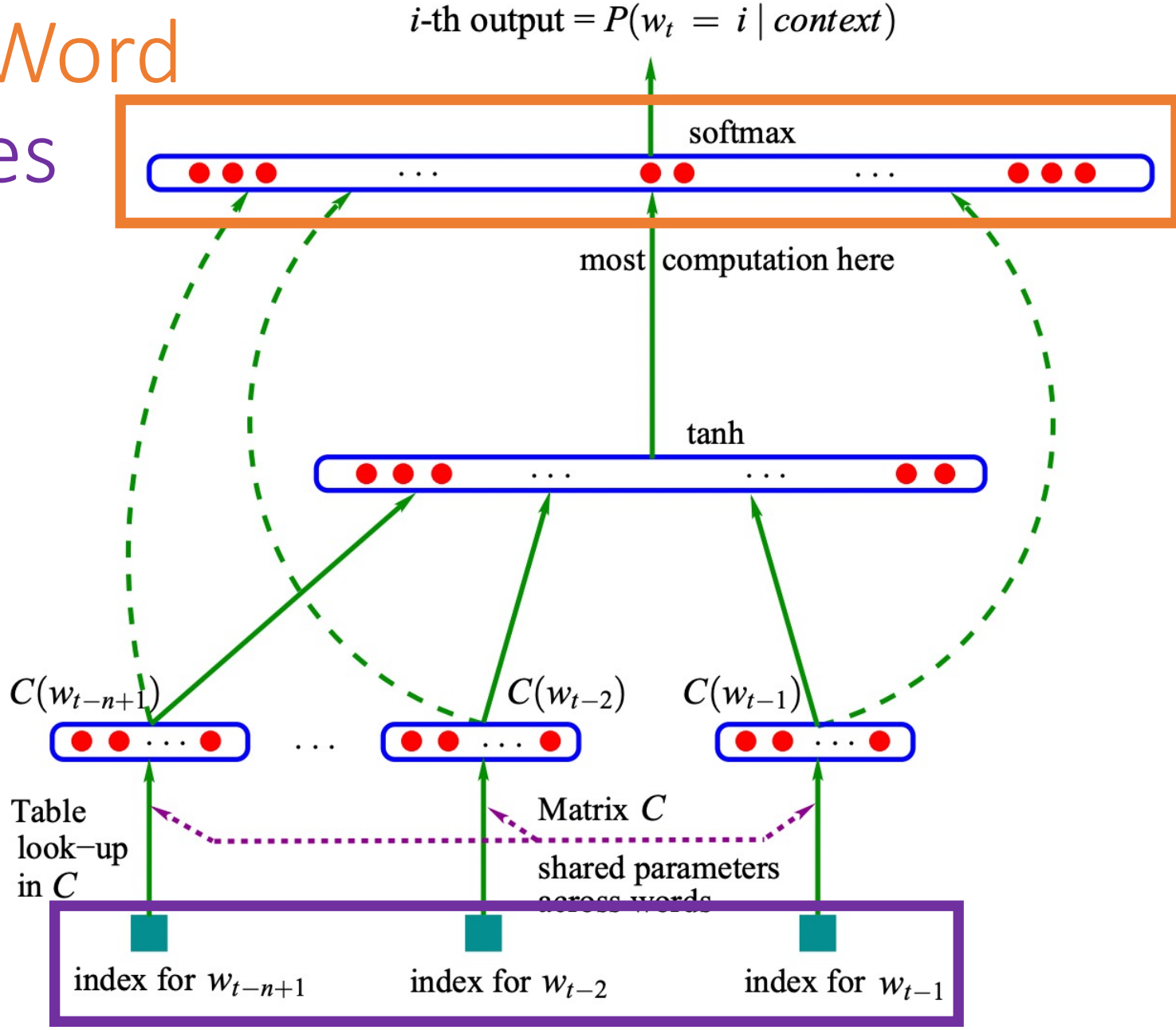
e.g.,

1. Background music from a _____
2. Many people danced around the _____
3. I practiced for many years to learn how to play the _____

Task: Predict Next Word Given Previous Ones

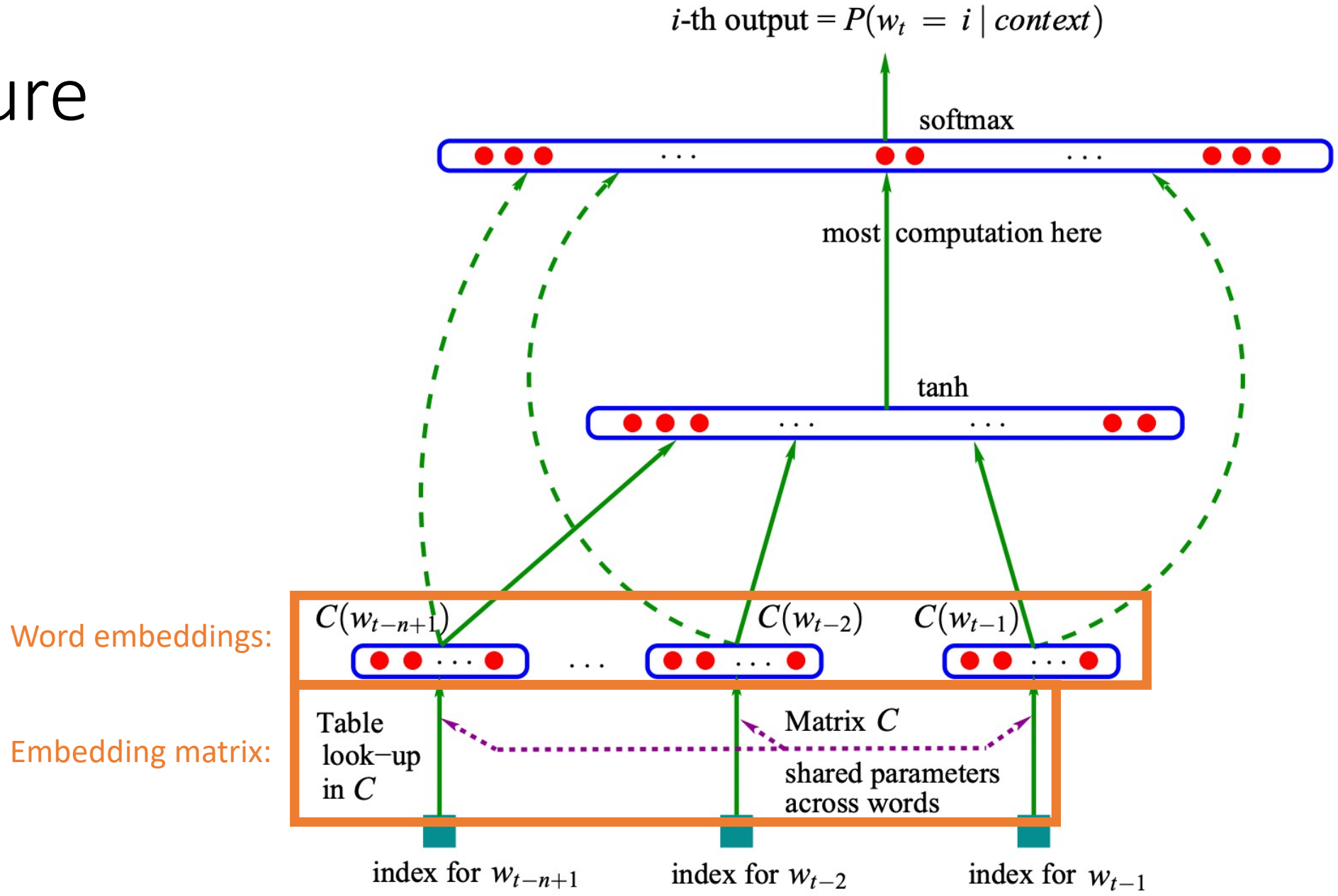
e.g., a vocabulary size of 17,000 was used in experiments

What is the dimensionality of the output layer?



Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Architecture



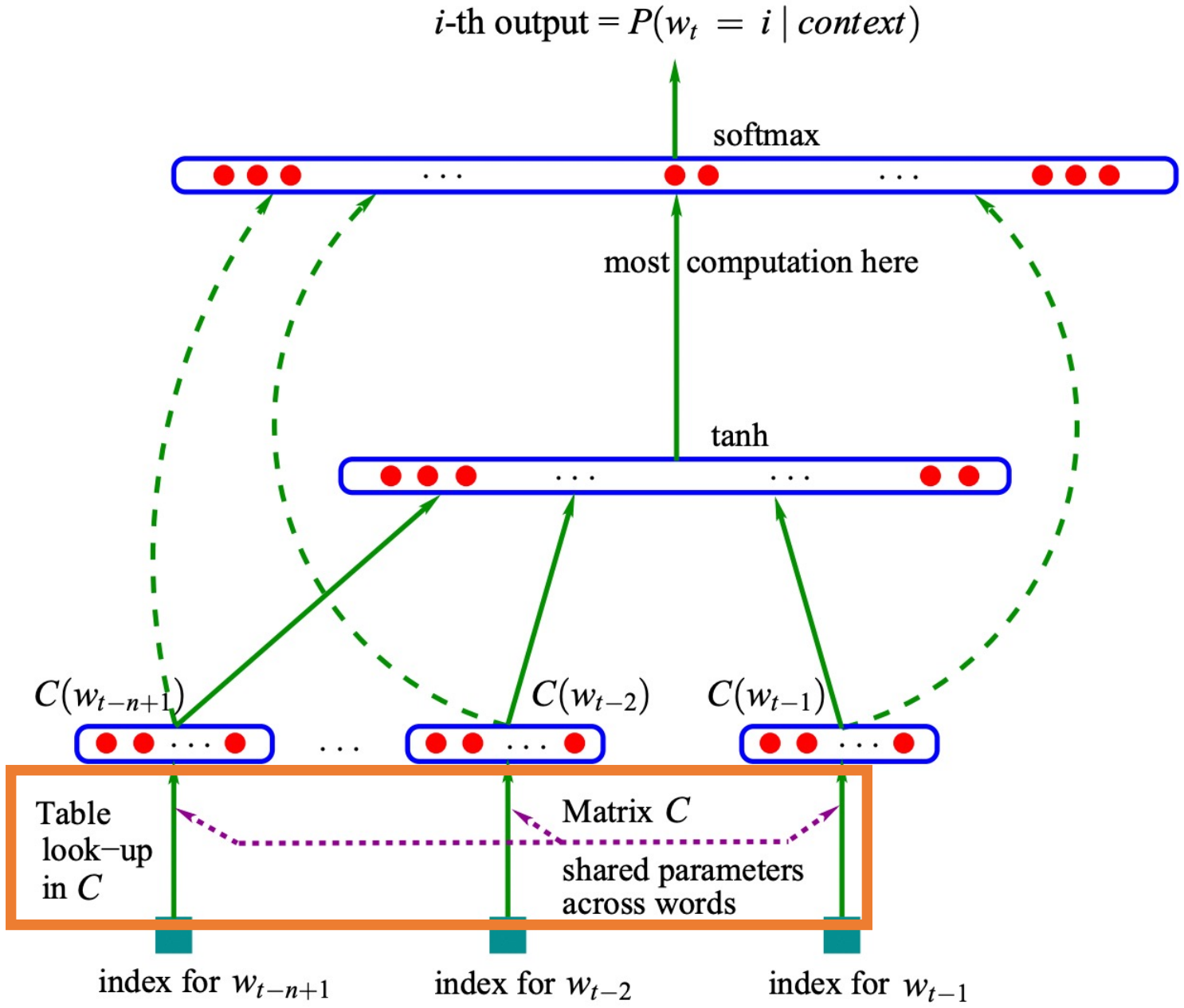
Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Architecture

e.g., a vocabulary size of 17,000 was used with embedding sizes of 30, 60, and 100 in experiments

Assume a 30-d word embedding
- what are the dimensions of the embedding matrix C?

30 x 17,000 (i.e., 510,000 weights)



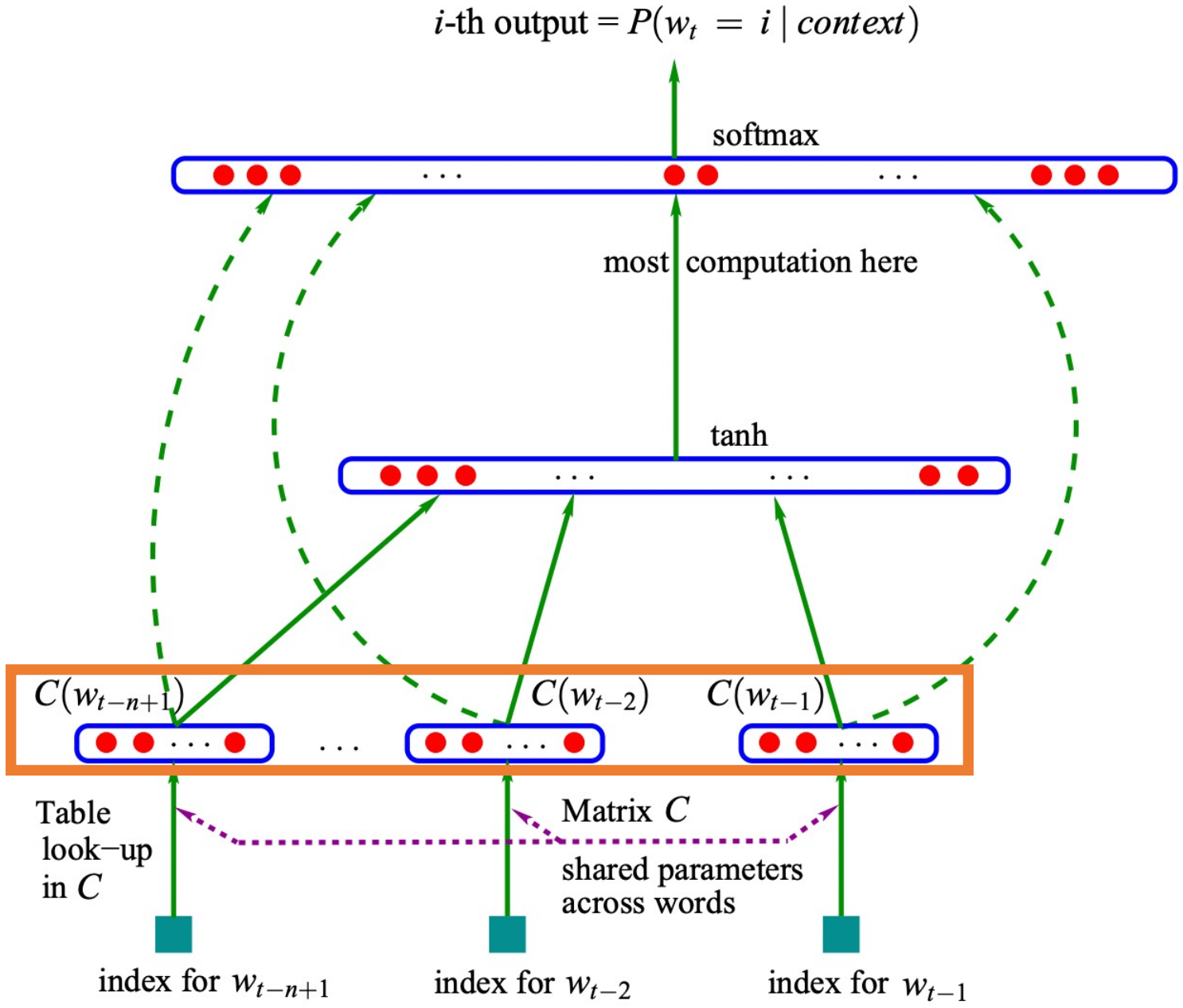
Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Architecture

e.g., a vocabulary size of 17,000 was used with embedding sizes of 30, 60, and 100 in experiments

Assume a 30-d word embedding
- what are the dimensions of each word embedding?

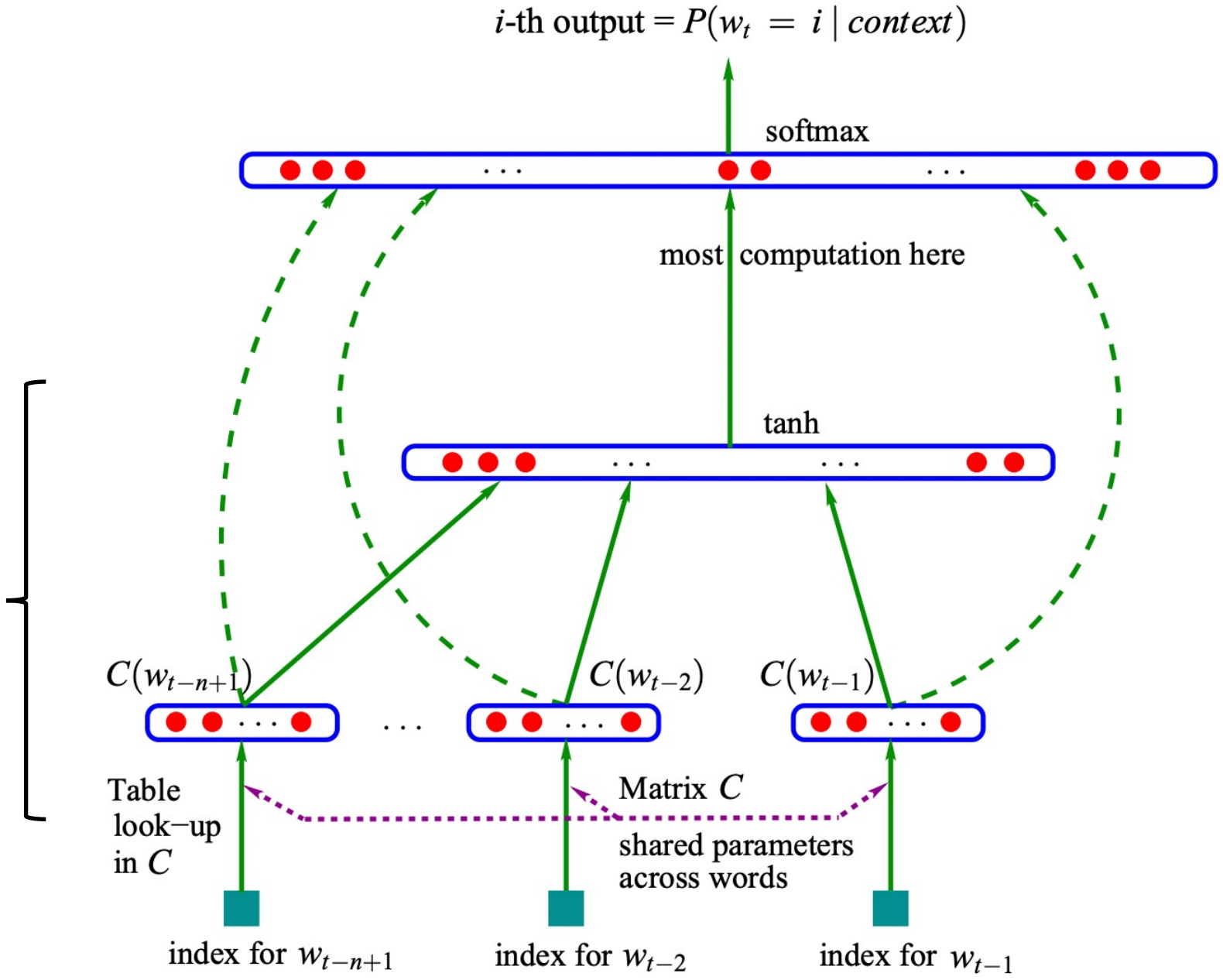
1 x 30



Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Architecture

Projection layer followed by a hidden layer with non-linearity



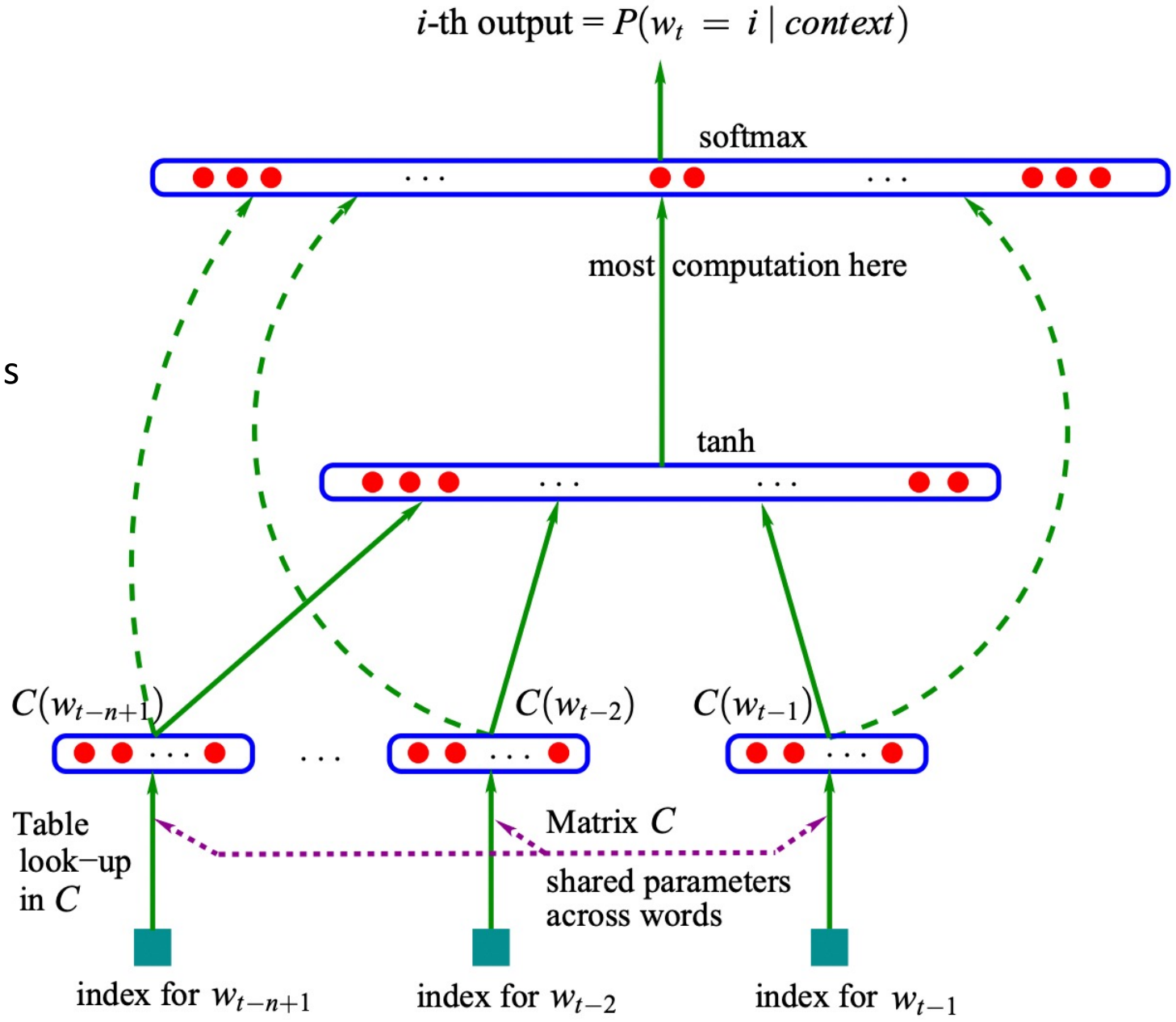
Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Training

Use sliding window on input data; e.g., 3 words

Background music from a berimbau offers a beautiful escape…

Input: tried 1, 3, 5, and 8 input words and used 2 datasets with ~1 million and ~34 million words respectively



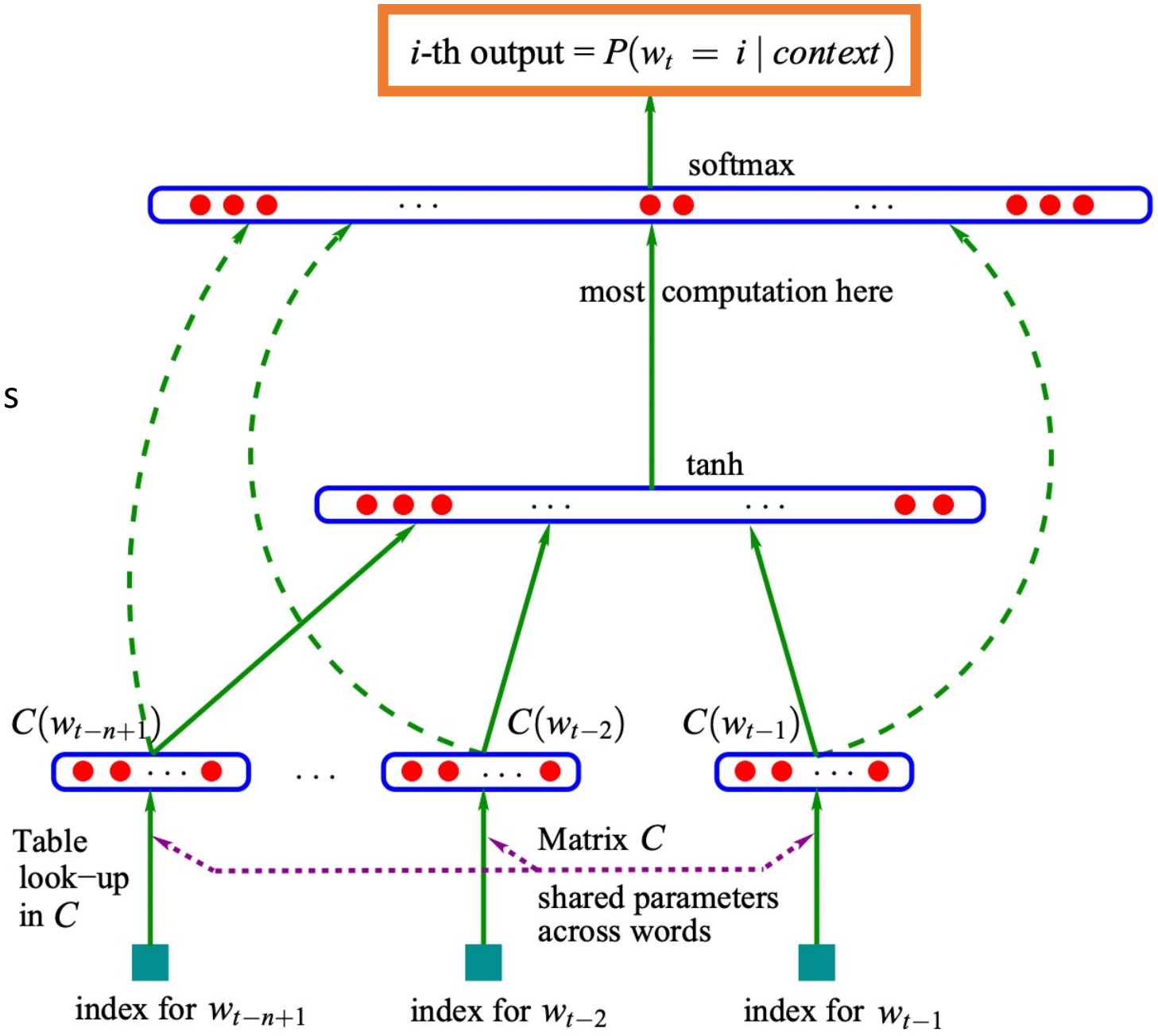
Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Training

Use sliding window on input data; e.g., 3 words

Background music from a berimbau offers a beautiful escape…

Input: tried 1, 3, 5, and 8 input words and used 2 datasets with ~1 million and ~34 million words respectively



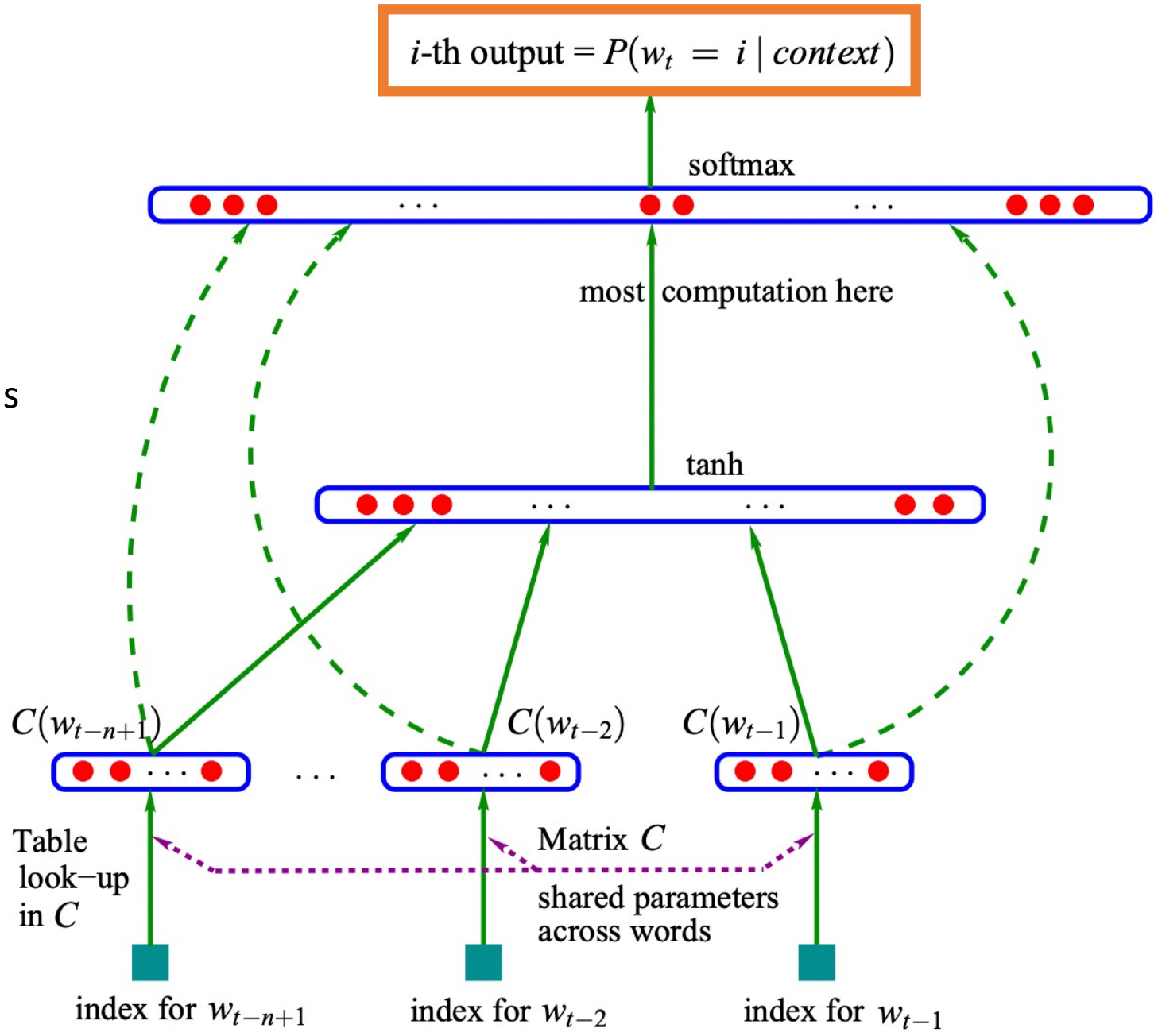
Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Training

Use sliding window on input data; e.g., 3 words

Background music from a berimbau offers a beautiful escape…

Input: tried 1, 3, 5, and 8 input words and used 2 datasets with ~1 million and ~34 million words respectively



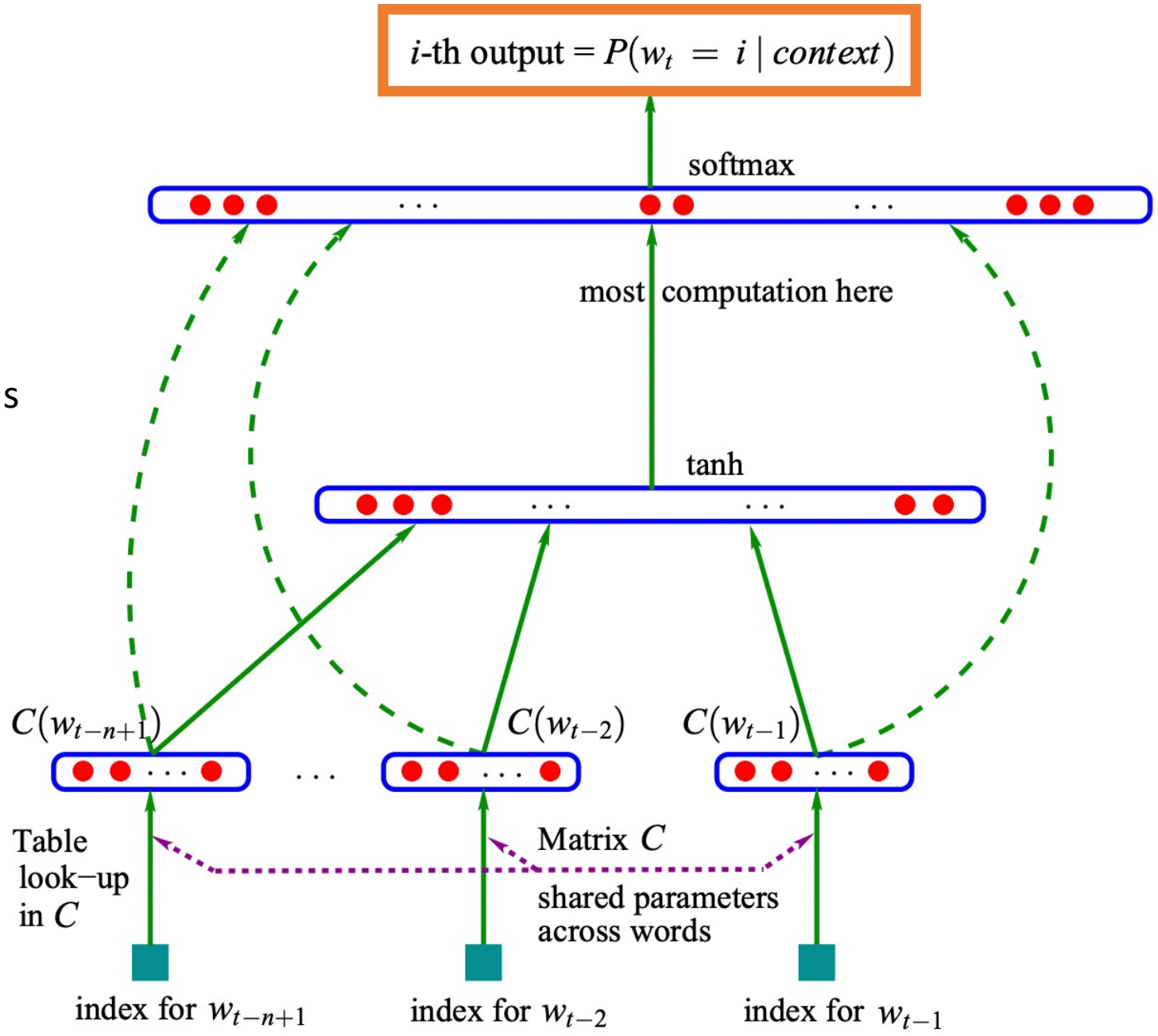
Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Training

Use sliding window on input data; e.g., 3 words

Background music from a berimbau offers a beautiful escape…

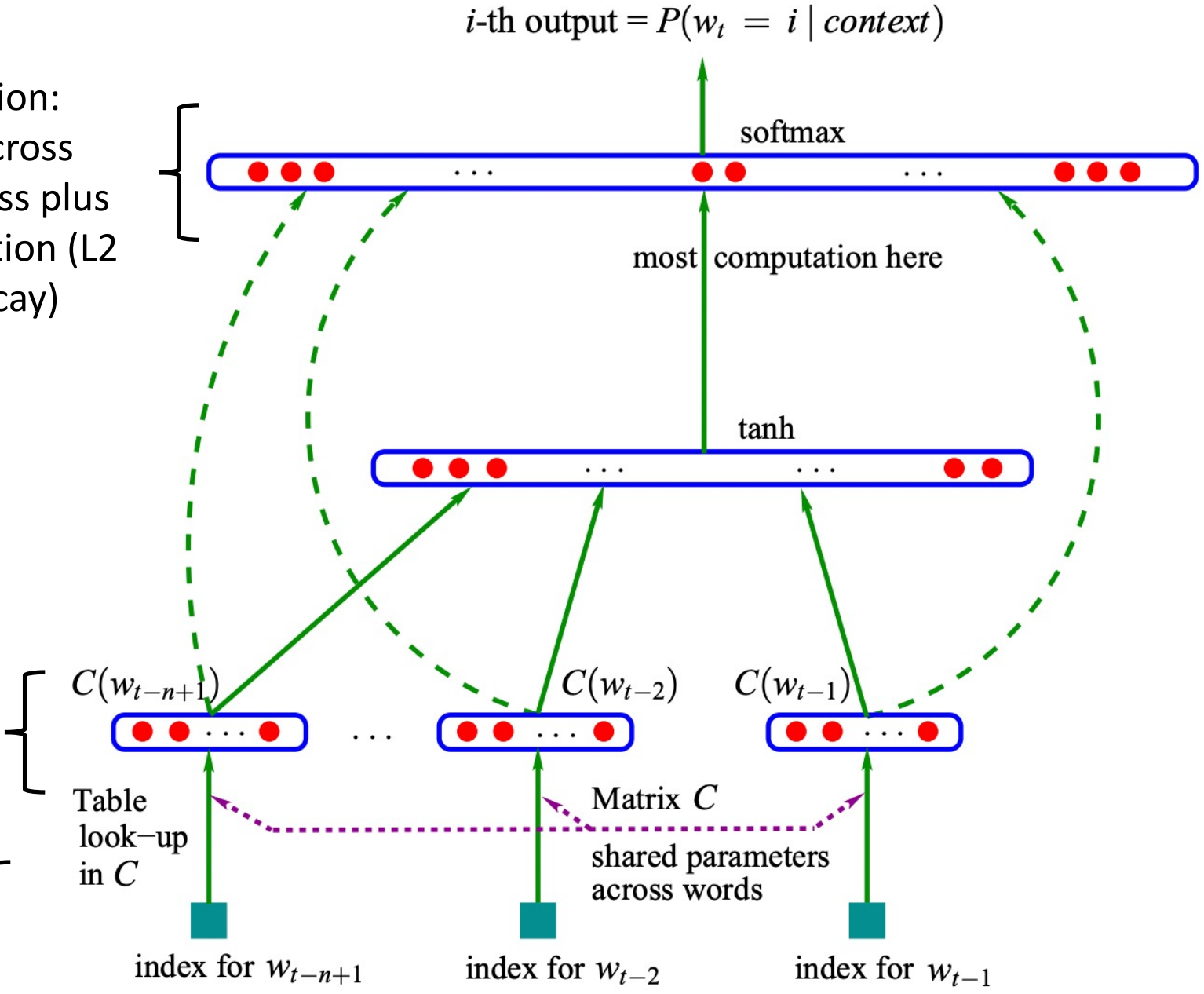
Input: tried 1, 3, 5, and 8 input words and used 2 datasets with ~1 million and ~34 million words respectively



Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Training

Cost function: minimize cross entropy loss plus regularization (L2 weight decay)



Word embedding iteratively updated

Input: tried 1, 3, 5, and 8 input words and used 2 datasets with ~1 million and ~34 million words respectively

Bengio et al. A Neural Probabilistic Language Model. JMLR 2003.

Summary: Word Embeddings Are Learned that Support Predicting Viable Next Words

e.g.,

1. Background music from a _____
2. Many people danced around the _____
3. I practiced for many years to learn how to play the _____

Popular Word Embeddings

- Bengio method
- Word2vec (skip-gram model)
- And more…

Idea: Learn Word Embeddings That Know What Are Viable Surrounding Words

e.g.,

1. ___ ___ ___ ___ berimbau ___ ___ ___ ___

2. ___ berimbau ___

Mikolov et al. Efficient Estimation of Word Representations in Vector Space. arXiv 2013.

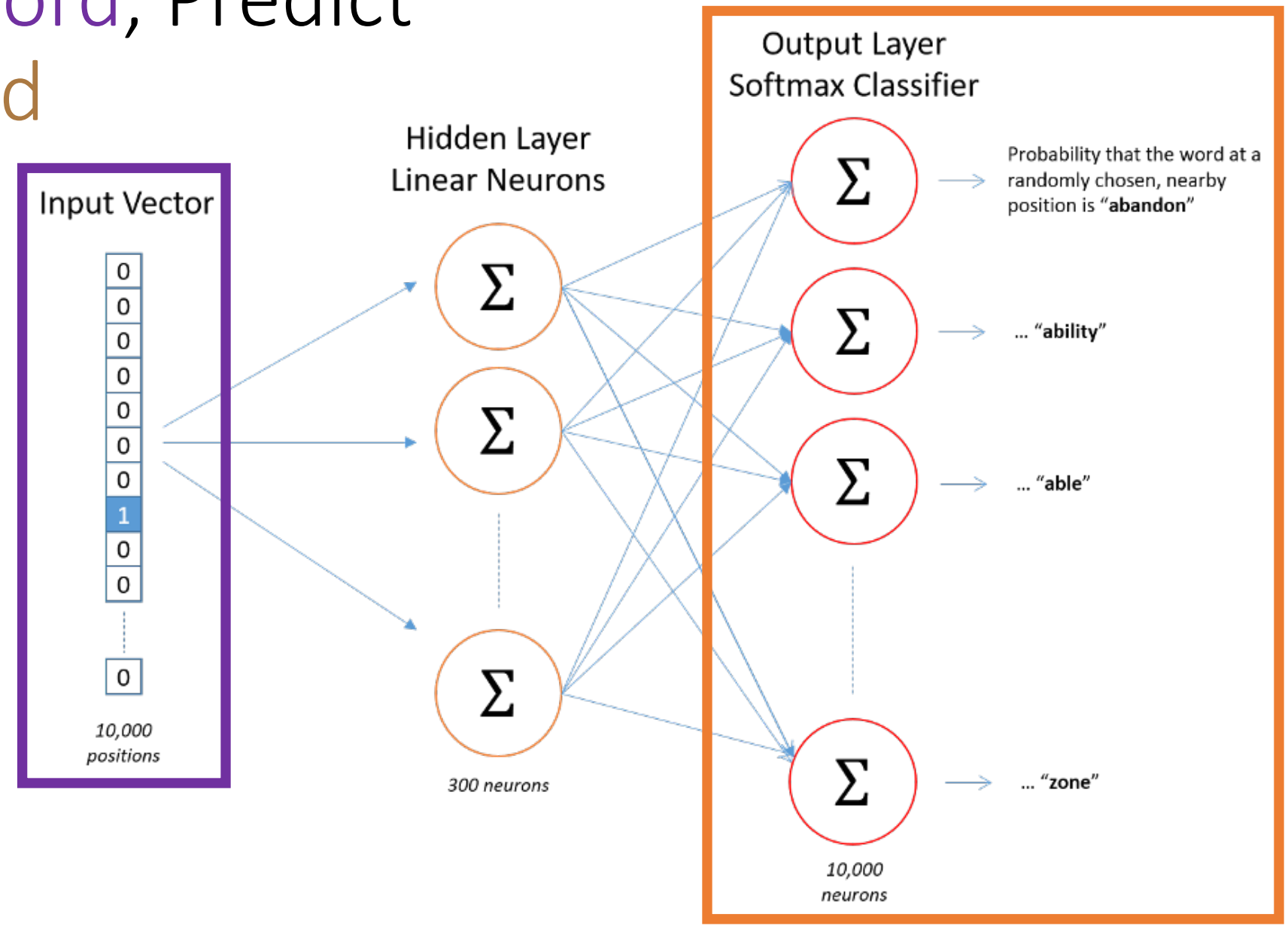
Task: Given Word, Predict a Nearby Word

e.g.,

1. ___ ___ ___ ___ berimbau ___ ___ ___ ___

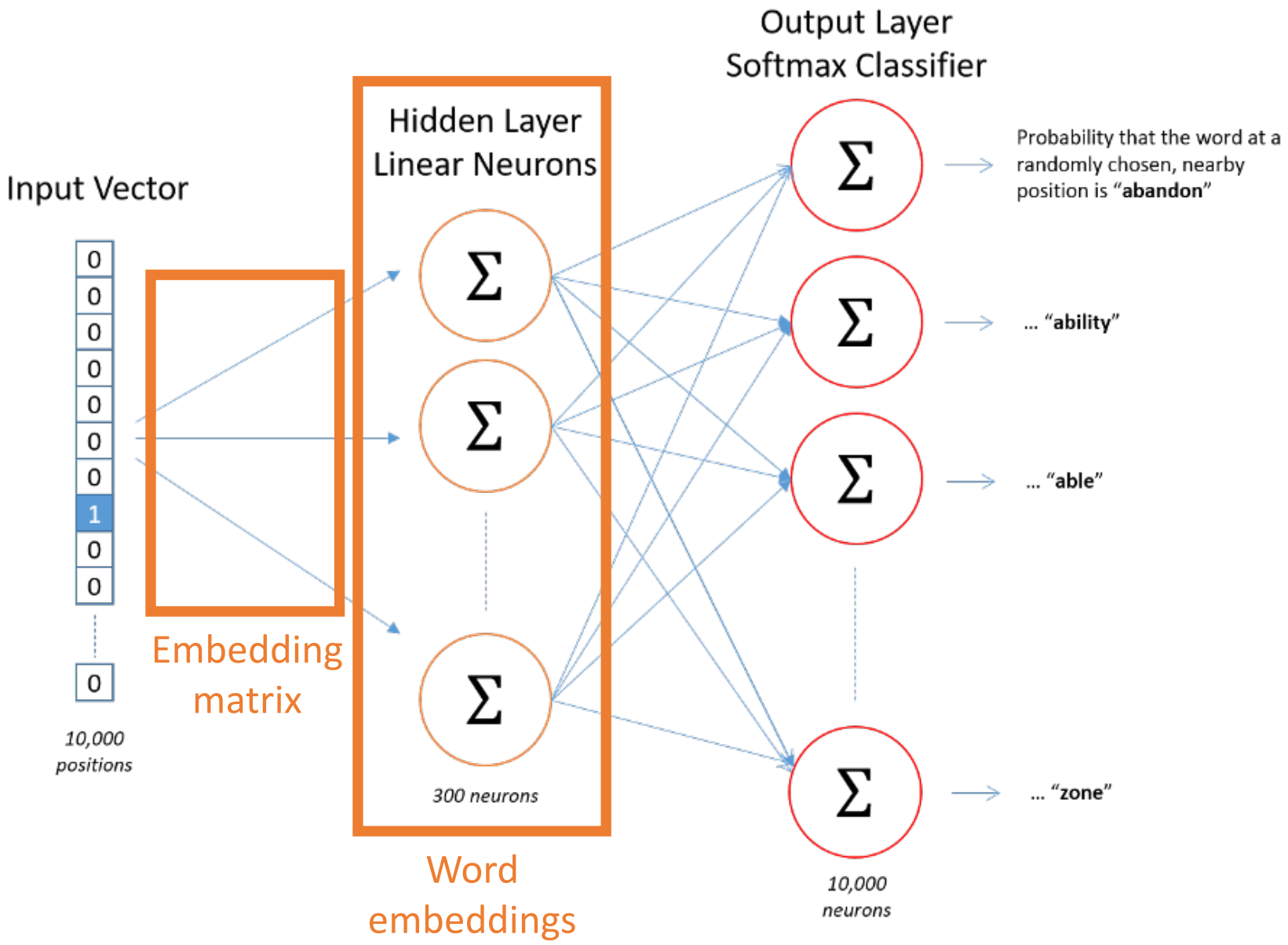
2. ___ berimbau ___

Task: Given Word, Predict a Nearby Word



https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b

Architecture



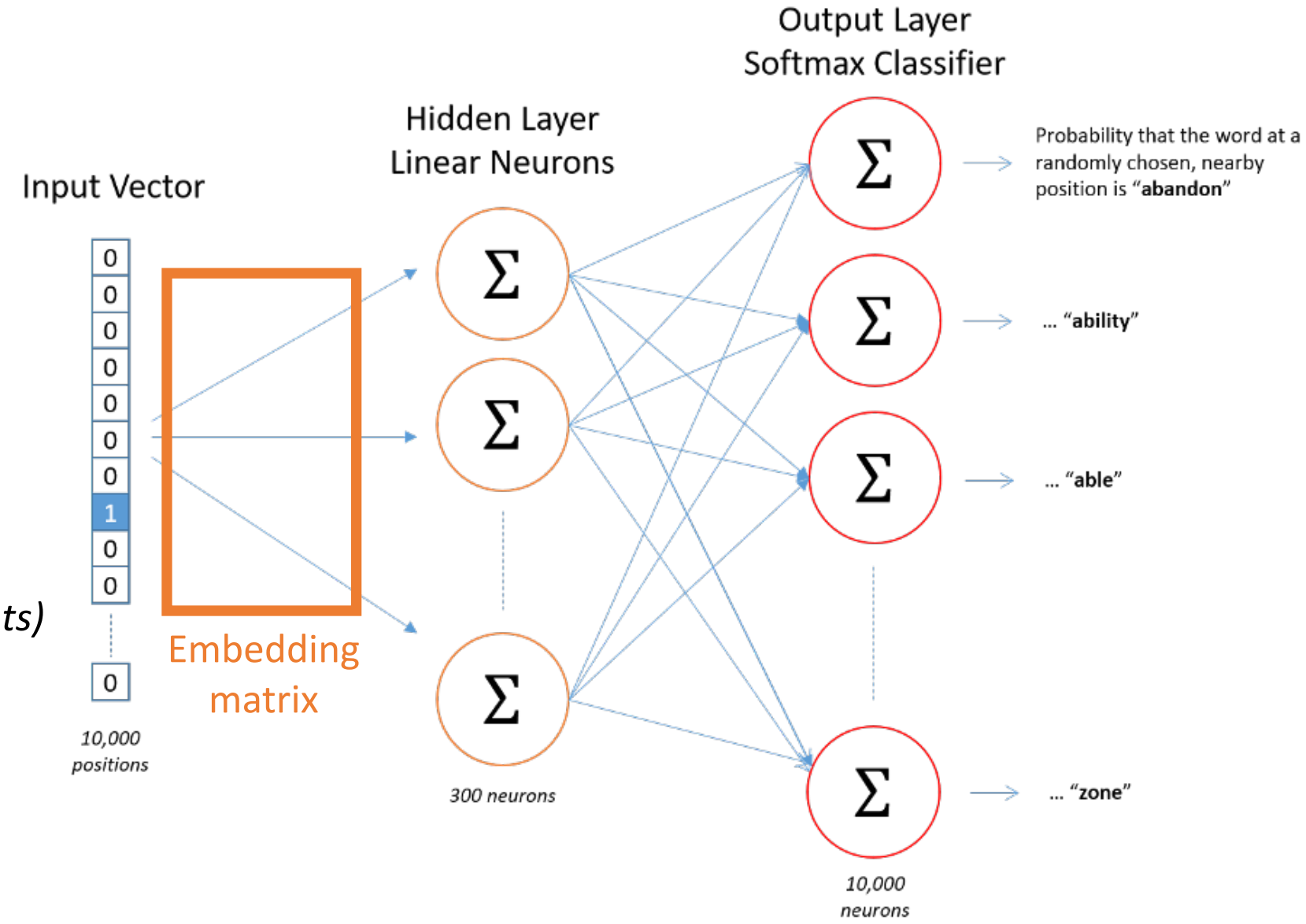
https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b

Architecture

e.g., a vocabulary size of 10,000 is used with embedding sizes of 300

What are the dimensions of the embedding matrix?

300 x 10,000 (i.e., 3,000,000 weights)



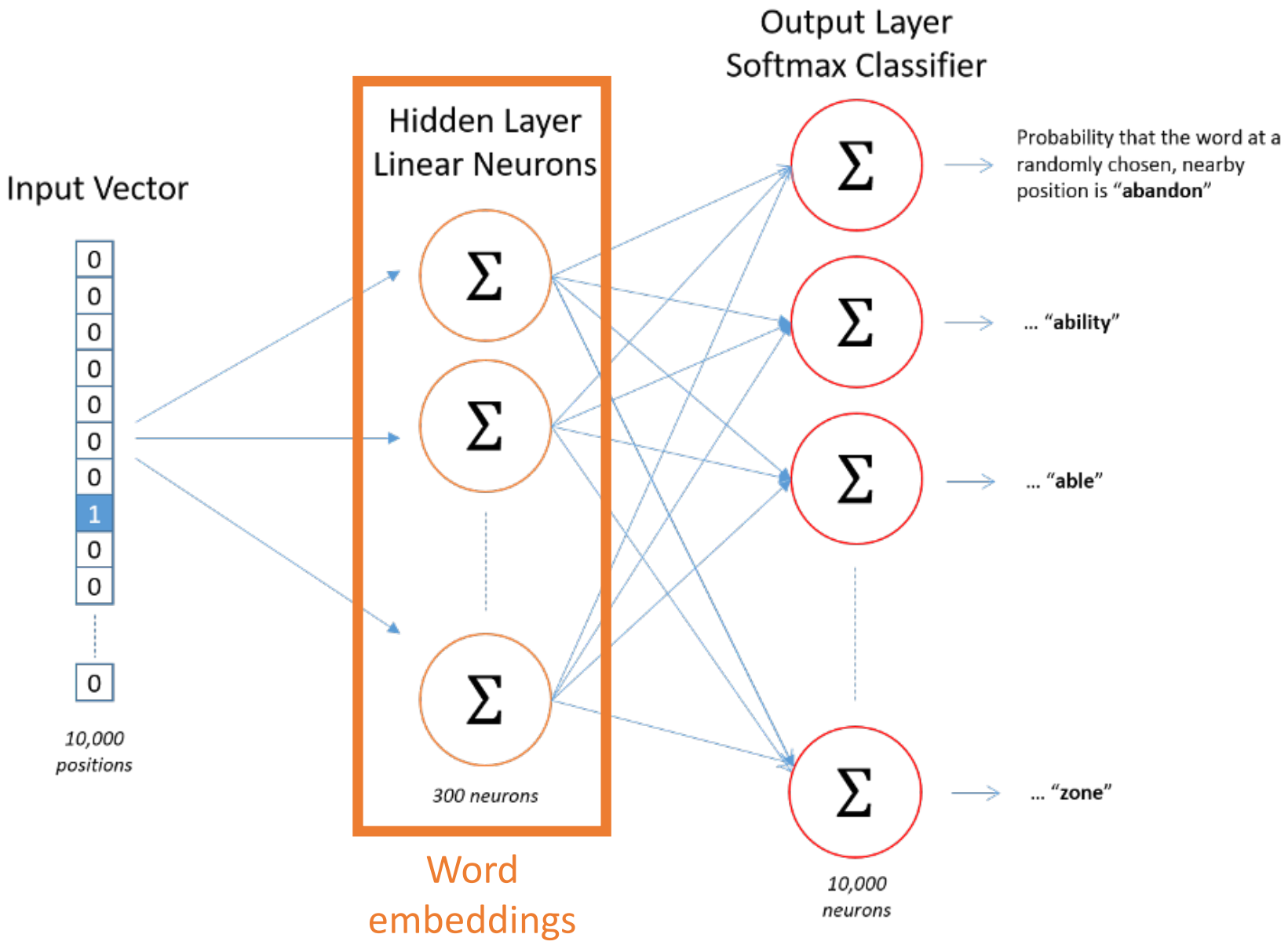
https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b

Architecture

e.g., a vocabulary size of 10,000 is used with embedding sizes of 300

What are the dimensions of each word embedding?

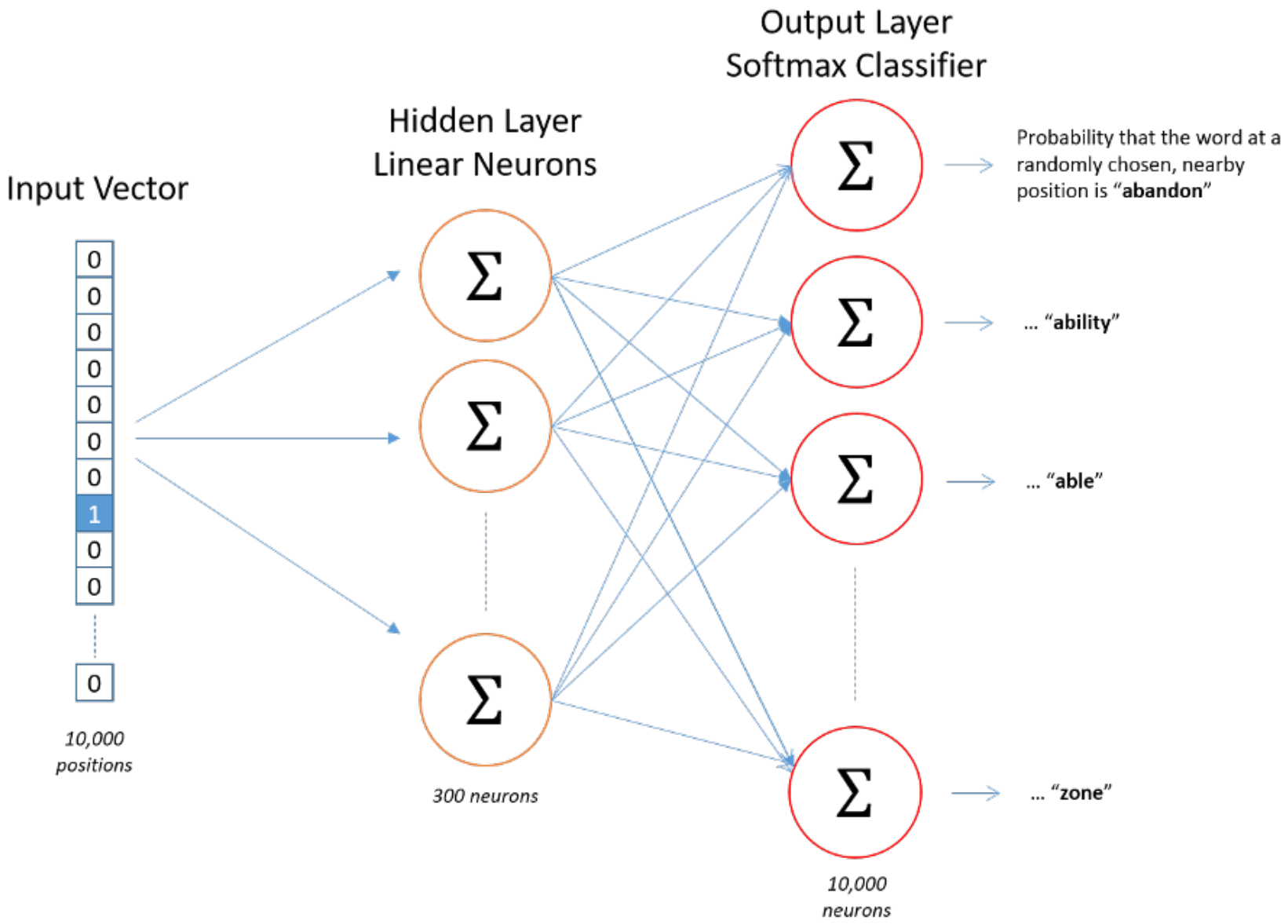
1 x 300



https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b

Architecture

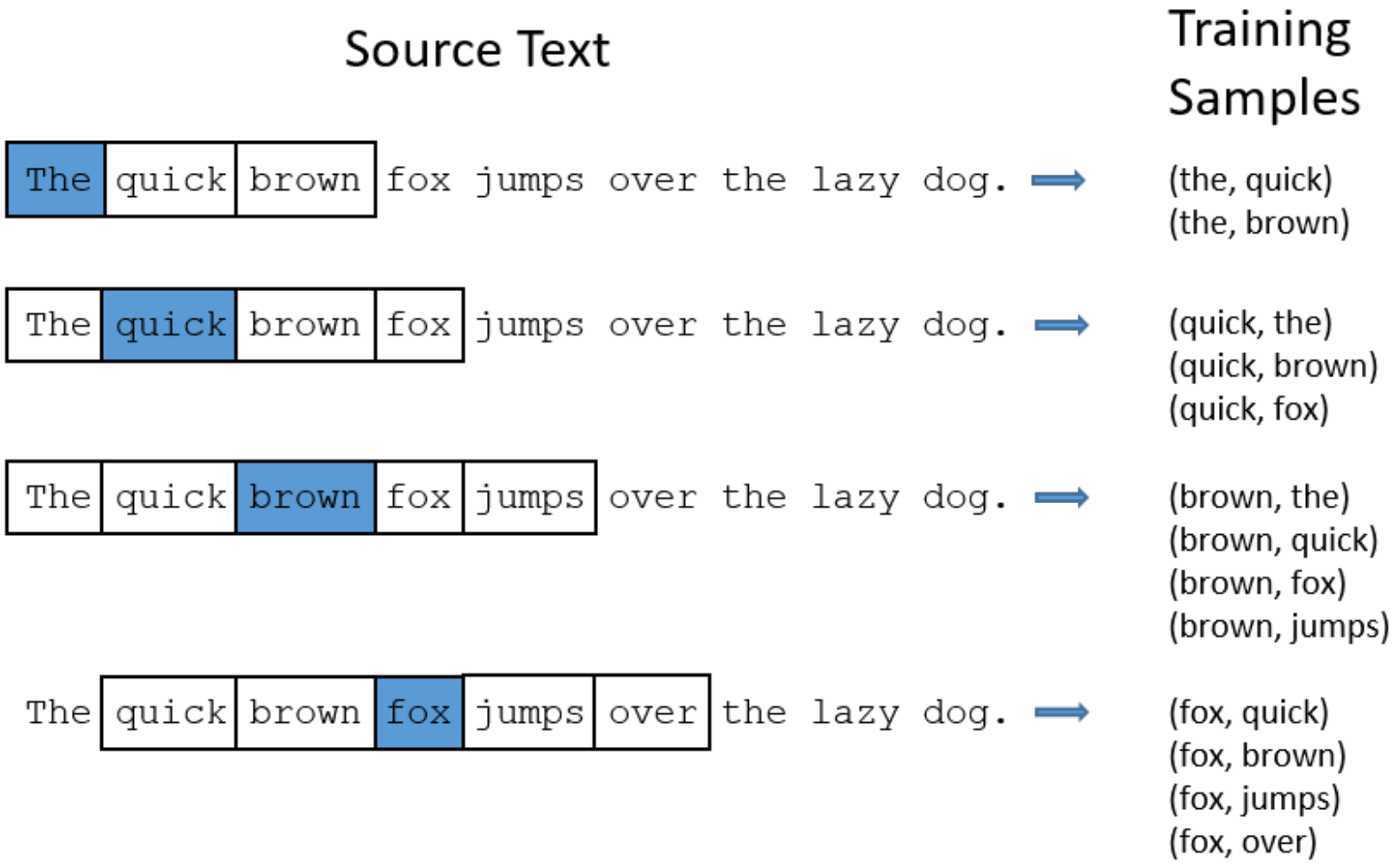
A shallower, simpler architecture than the Bengio approach (i.e., lacks a non-linear hidden layer)!



https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b

Training

Sliding window run on input data to sample neighbors of each target word (e.g., using window size of 2)

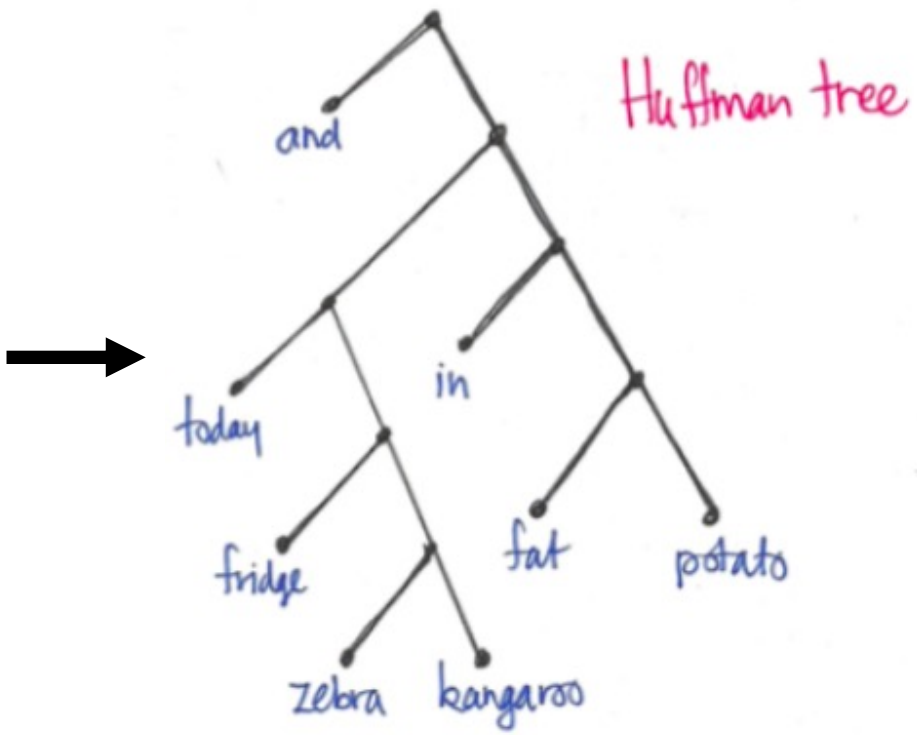


https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b

Extra Tricks: More Efficient Representations

1. Change output layer to hierarchical softmax

word	count
fat	3
fridge	2
zebra	1
potato	3
and	14
in	7
today	4
kangaroo	2



2. Reformulate problem to perform negative sampling

Binary classification: predict for a given word if another word is nearby

- Positive examples: observed target and neighboring words
- Negative examples: randomly sampled other words

https://www.cs.princeton.edu/courses/archive/spring20/cos598C/lectures/lec2-word-embeddings.pdf

Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality. Neurips 2013.

Hyperparameters: What Works Well?

- Word embedding dimensionality?
  - Dimensionality set between 100 and 1,000

- Context window size?
  - ~10

Mikolov et al. Efficient Estimation of Word Representations in Vector Space. arXiv 2013.

Very Exciting/Surprising Finding

- Vector arithmetic with word embeddings can solves many analogies

  (Full test list: http://download.tensorflow.org/data/questions-words.txt)

- Semantic relationships (meaning of words in a sentence):
  - Italy + (Paris - France) = Rome

- Syntactic relationships (rules for words in a sentence)
  - smallest + (big – small) = biggest
  - think + (read – reading) = thinking
  - mouse + (dollars – dollar) = mice

Mikolov et al. Efficient Estimation of Word Representations in Vector Space. arXiv 2013.

Summary: Word Embeddings Are Learned that Support Predicting Viable Surrounding Words!

e.g.,

1. ___ ___ ___ ___ berimbau ___ ___ ___ ___

2. ___ berimbau ___

Popular Word Embeddings

- Bengio method
- Word2vec (skip-gram model)
- And more…

Variants for Learning Word Embeddings

- Capture global context rather than just local context of previous or surrounding words; e.g.,
  - GloVe for Global Vectors (Pennington et al., 2014)

- Capture that the same word can have different word vectors under different contexts; e.g.,
  - Elmo for embeddings from language models (Peters et al., arXiv 2018)

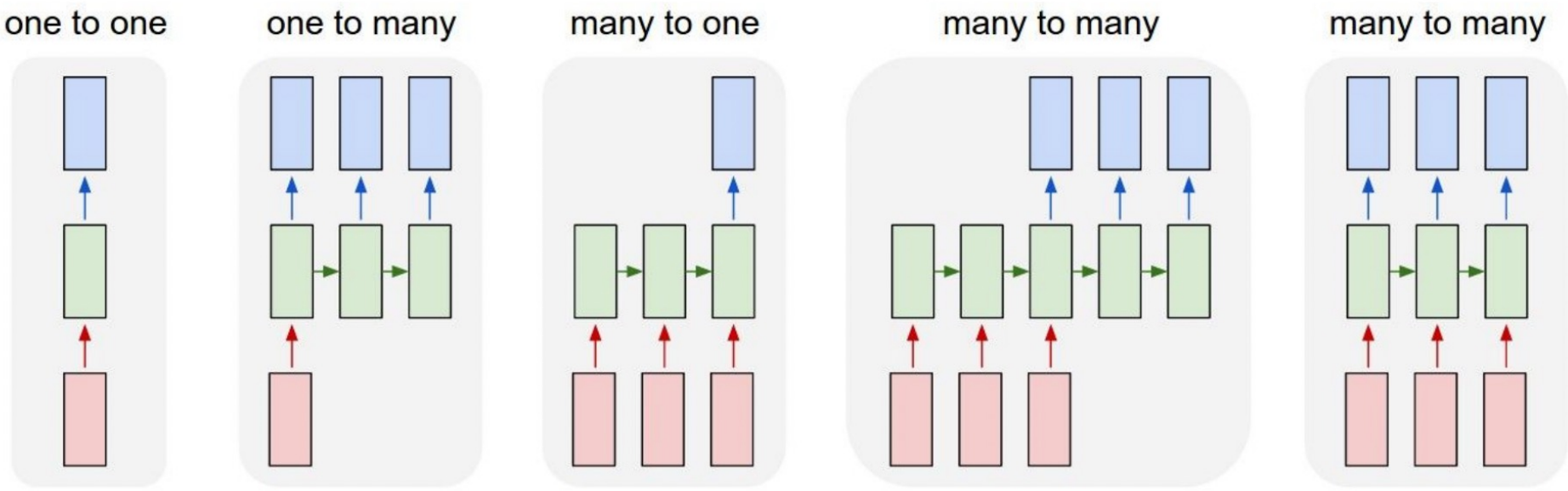
- Support multiple languages; e.g.,
  - Fast-text (Bojanowski et al., 2016)

Popular Word Embeddings

- Bengio method
- Word2vec (skip-gram model)
- And more…

Recap of Big Picture

- Convert words into compact vectors as input to neural networks; e.g., RNNs



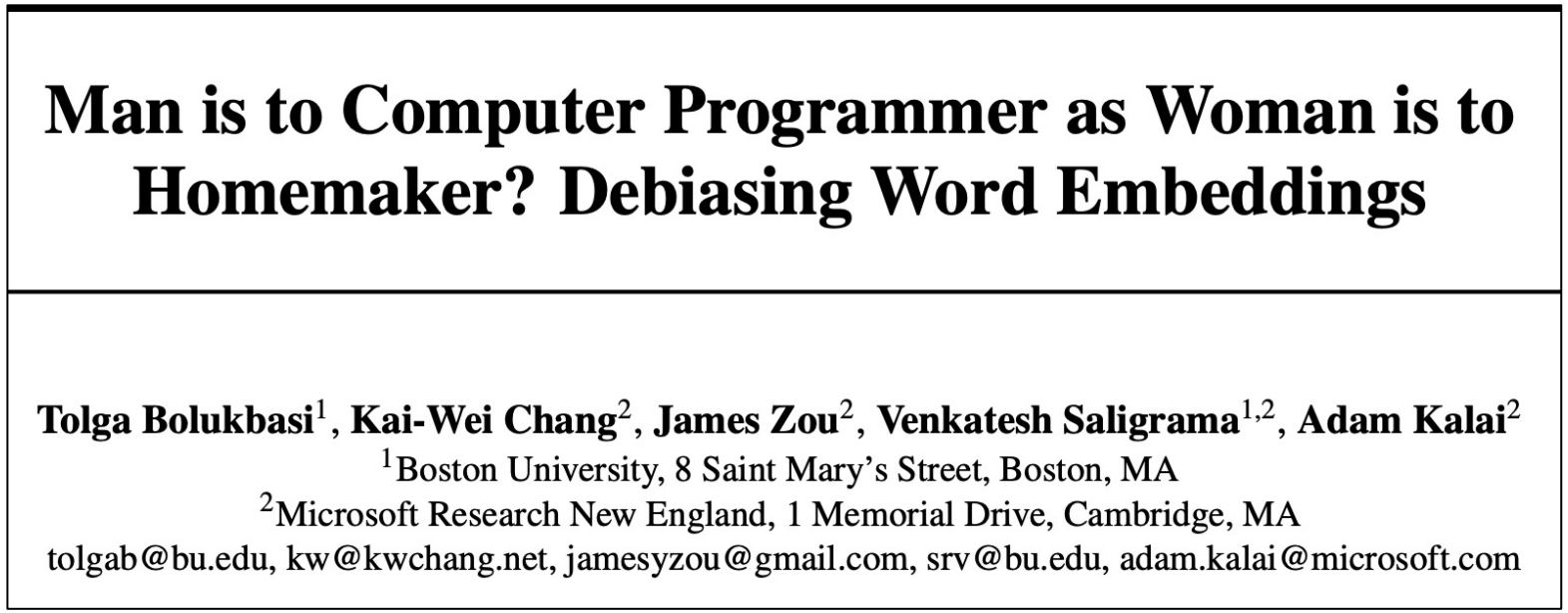
  - Implementation detail: may need to learn extra tokens such as “UNK” and “EOS” to represent out of vocabulary words and signify end of the string respectively
- Also, can fine-tune word embedding matrices for different applications

https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

Word Embedding Limitations/Challenges

- Distinguish antonyms from synonyms
  - Antonyms are learned near each other in the embedding space since they are commonly used in similar contexts: “I hate math” vs “I love math” or “Take a right turn” vs “Take a left turn”

- Gender bias:



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

Word Embedding Limitations/Challenges

- Distinguish antonyms from synonyms
  - Antonyms are learned near each other in the embedding space since they are commonly used in similar contexts: “I hate math” vs “I love math” or “Take a right turn” vs “Take a left turn”

- Gender bias:

Extreme *she*	**Extreme *he***
1. homemaker	1. maestro
2. nurse	2. skipper
3. receptionist	3. protege
4. librarian	4. philosopher
5. socialite	5. captain
6. hairdresser	6. architect
7. nanny	7. financier
8. bookkeeper	8. warrior
9. stylist	9. broadcaster
10. housekeeper	10. magician

Gender stereotype *she-he* analogies

sewing-carpentry	registered nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	lovely-brilliant

Gender appropriate *she-he* analogies

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

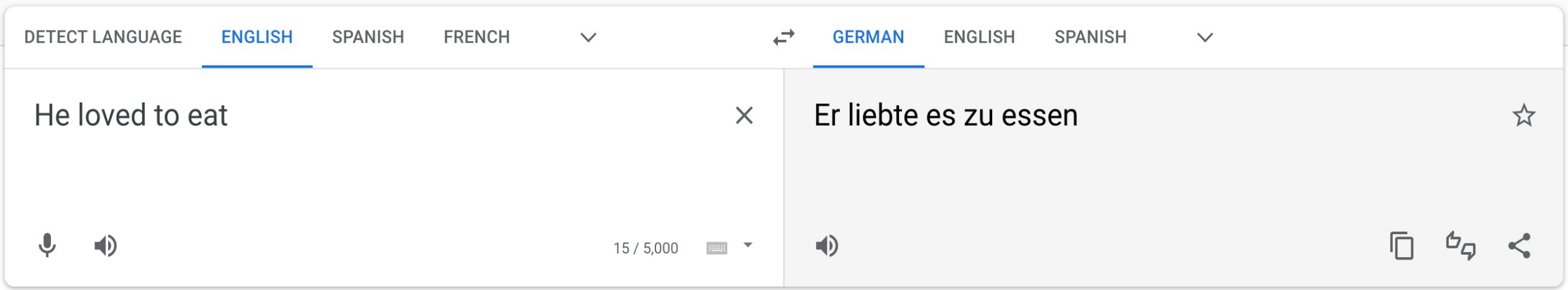
Bolukbasi et al. Neurips 2016.

Word Embedding Limitations/Challenges

- Distinguish antonyms from synonyms
  - Antonyms are learned near each other in the embedding space since they are commonly used in similar contexts: “I hate math” vs “I love math” or “Take a right turn” vs “Take a left turn”
- Gender bias
- What other language biases do you think could be learned?

Bolukbasi et al. Neurips 2016.

Task: Machine Translation



Pioneering Neural Network Approach

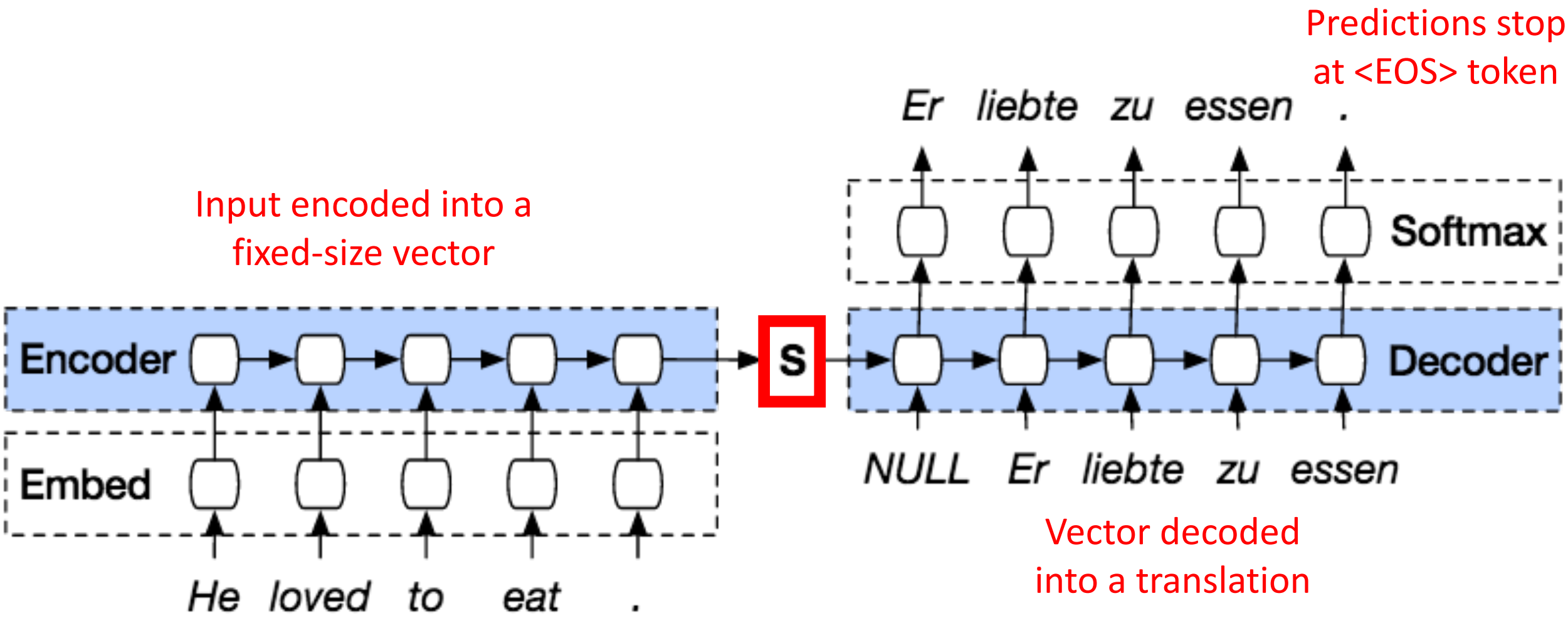


Image source: https://smerity.com/articles/2016/google_nmt_arch.html

seq2seq: Sutskever et al. Sequence to Sequence Learning with Neural Networks. Neurips 2014.

Pioneering Neural Network Approach

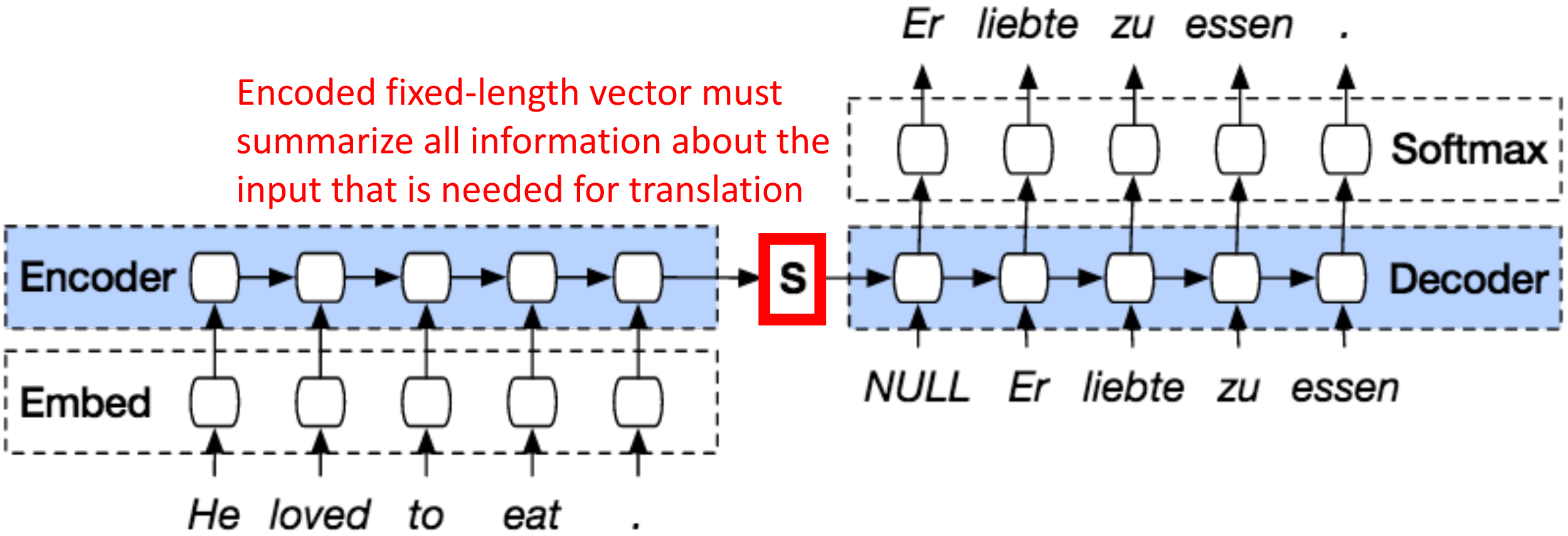
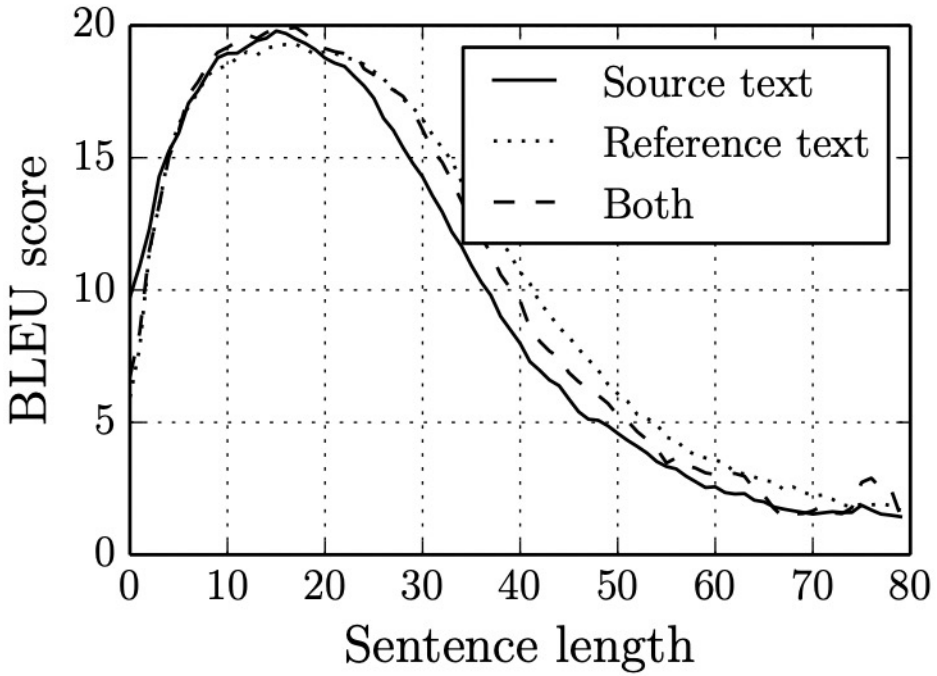


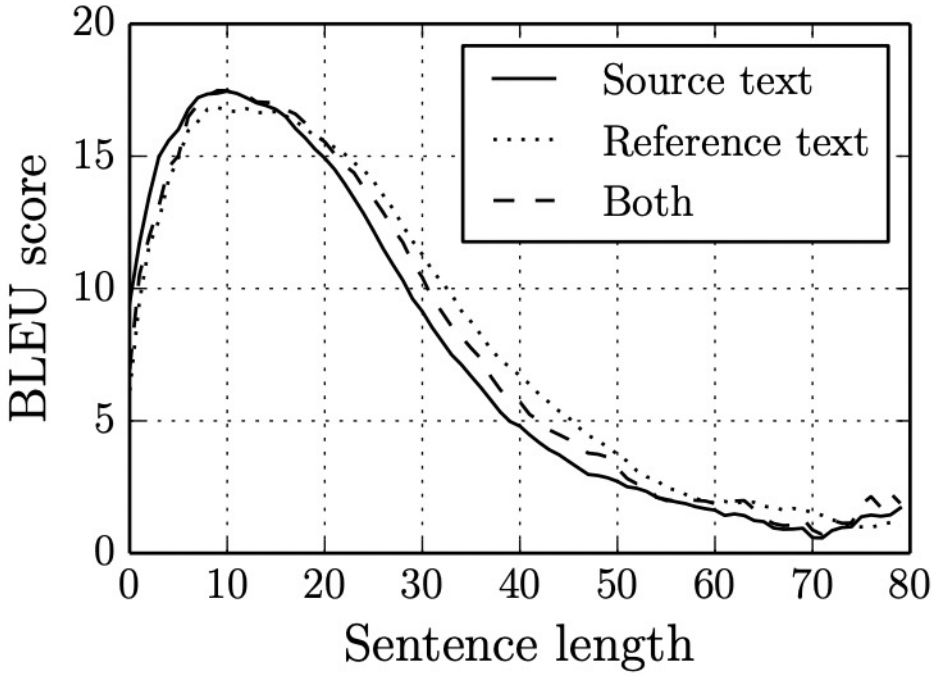
Image source: https://smerity.com/articles/2016/google_nmt_arch.html

Sutskever et al. Sequence to Sequence Learning with Neural Networks. Neurips 2014.

Analysis of Two Models

(larger scores are better)



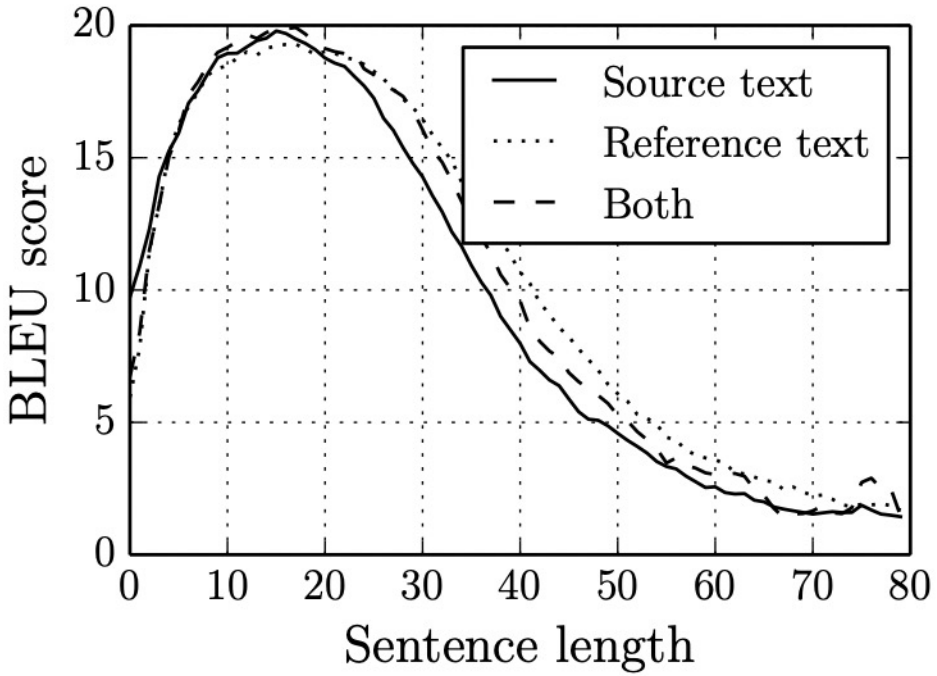


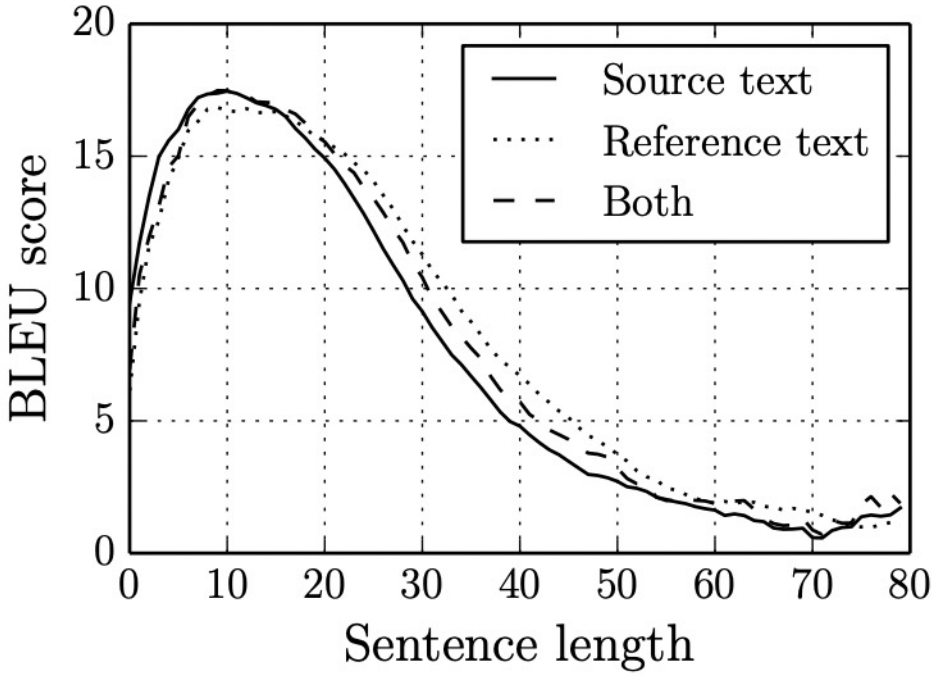
What performance trend is observed for inputs (source) and outputs (reference) as the number of words in each sentence grows?

Cho et al. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. SSST 2014.

Analysis of Two Models

(larger scores are better)





Performance drops for longer sentences!

Cho et al. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. SSST 2014.

Problem: Performance Drops As Sentence Length Grows

Hypothesis: fixed-length vector lacks sufficient capacity to capture all relevant information for long sentences

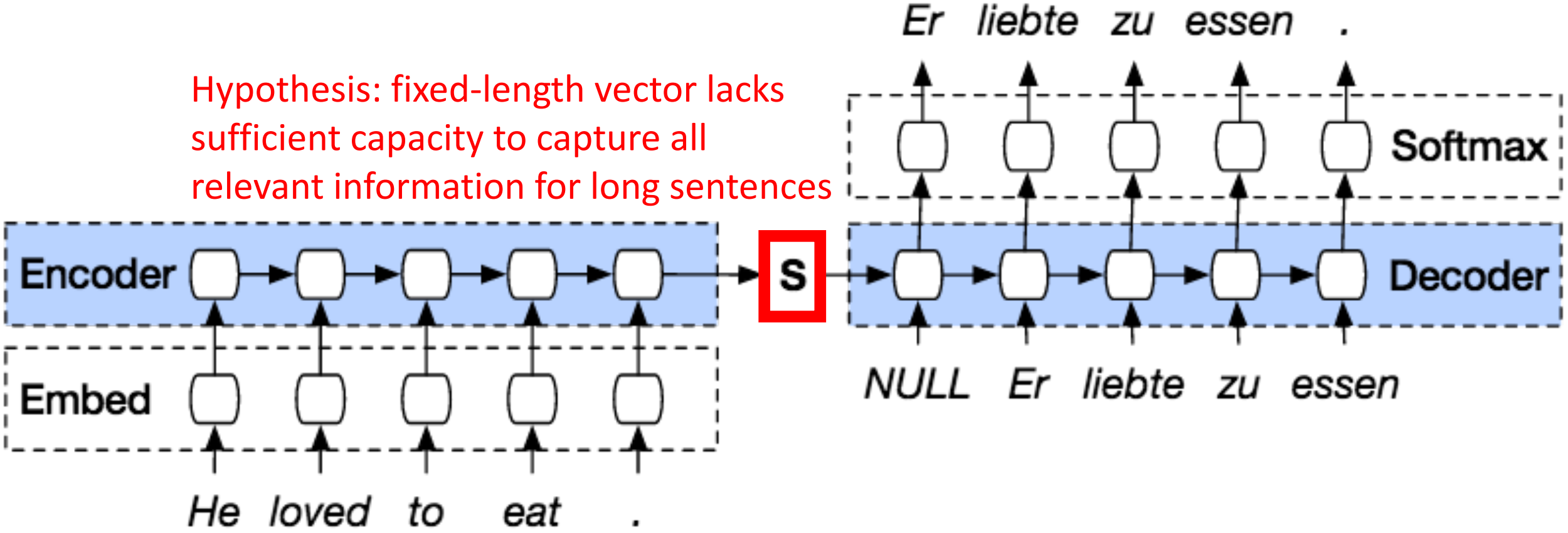


Image source: https://smerity.com/articles/2016/google_nmt_arch.html

Cho et al. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. SSST 2014.

Idea to Preserve Performance for Long Sentences: **Attention**

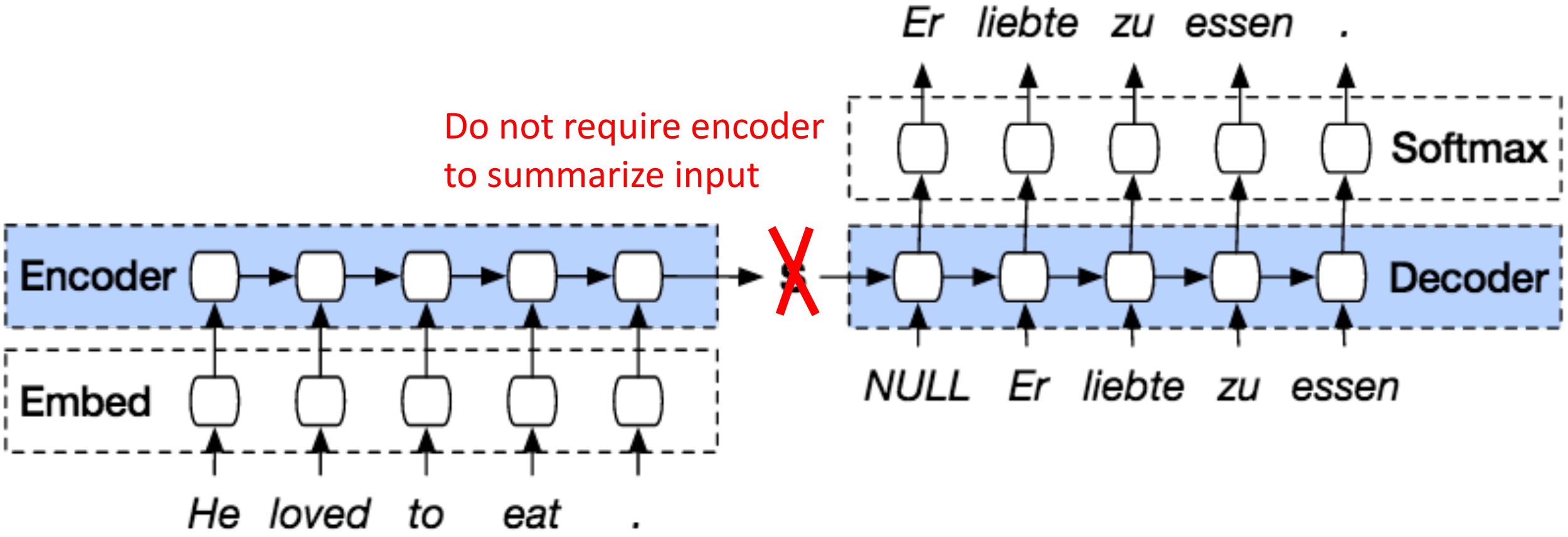


Image source: https://smerity.com/articles/2016/google_nmt_arch.html

Idea to Preserve Performance for Long Sentences: **Attention**

Instead, have the encoder pass **all** input's hidden states to the decoder to decide which to use for prediction at each time step

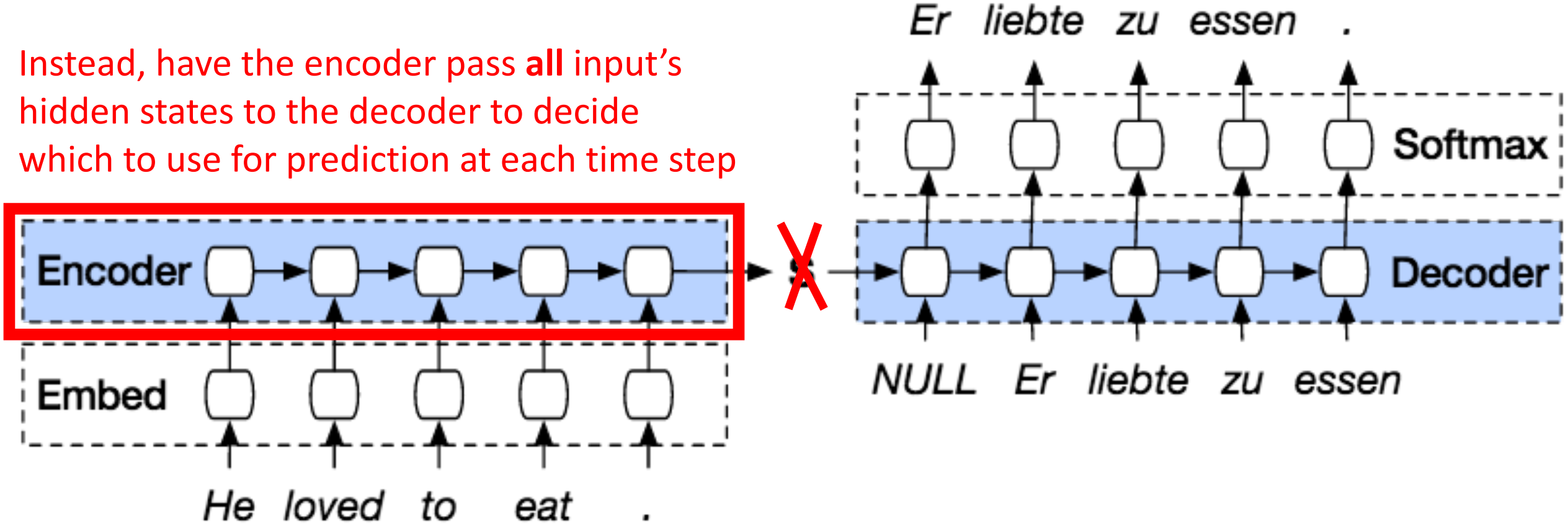
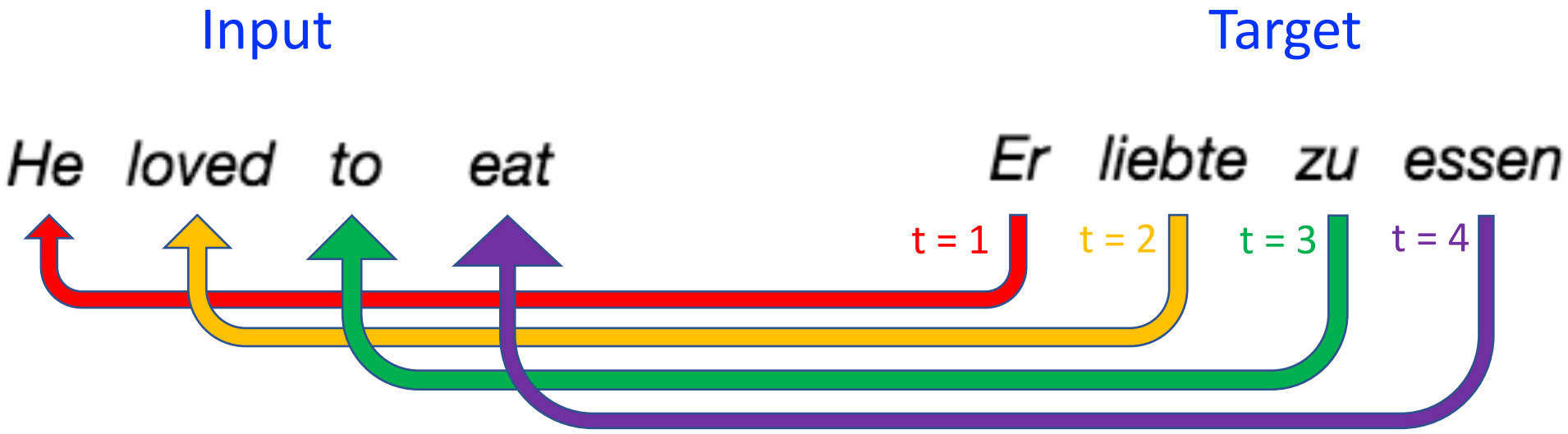


Image source: https://smerity.com/articles/2016/google_nmt_arch.html

Idea to Preserve Performance for Long Sentences: **Attention**

Decoder decides which inputs are needed for prediction at each time step; e.g., “hard attention” focuses on one input

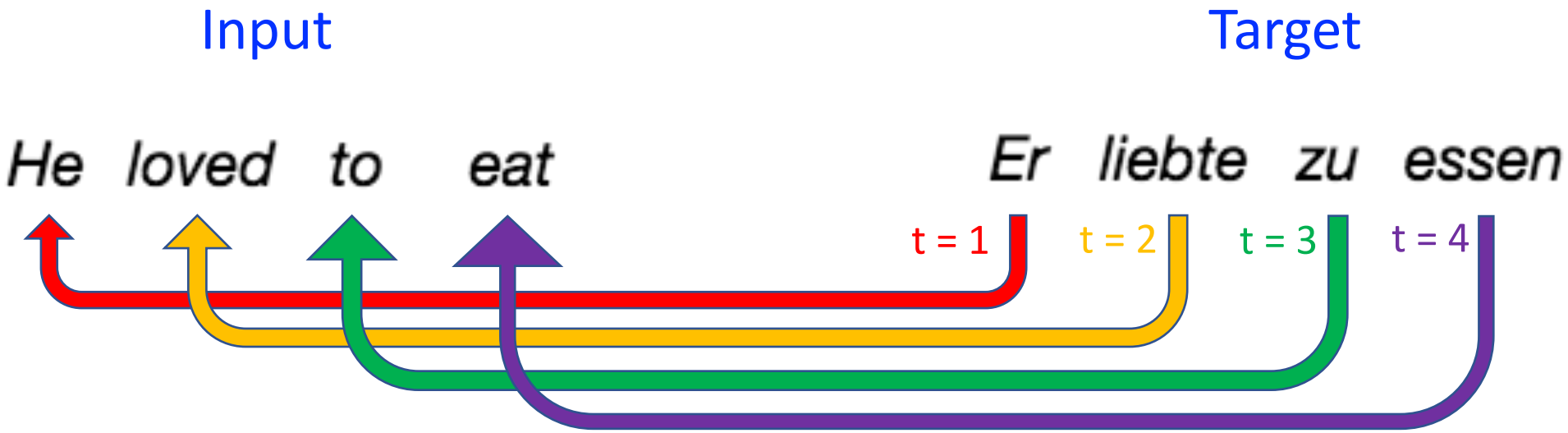


Note: while word order between the input and target align in this example, it can differ

https://deeplearning.cs.cmu.edu/F21/document/slides/lec18.attention.pdf

Idea to Preserve Performance for Long Sentences: **Attention**

Decoder decides which inputs are needed for prediction at each time step; e.g., “hard attention” focuses on one input

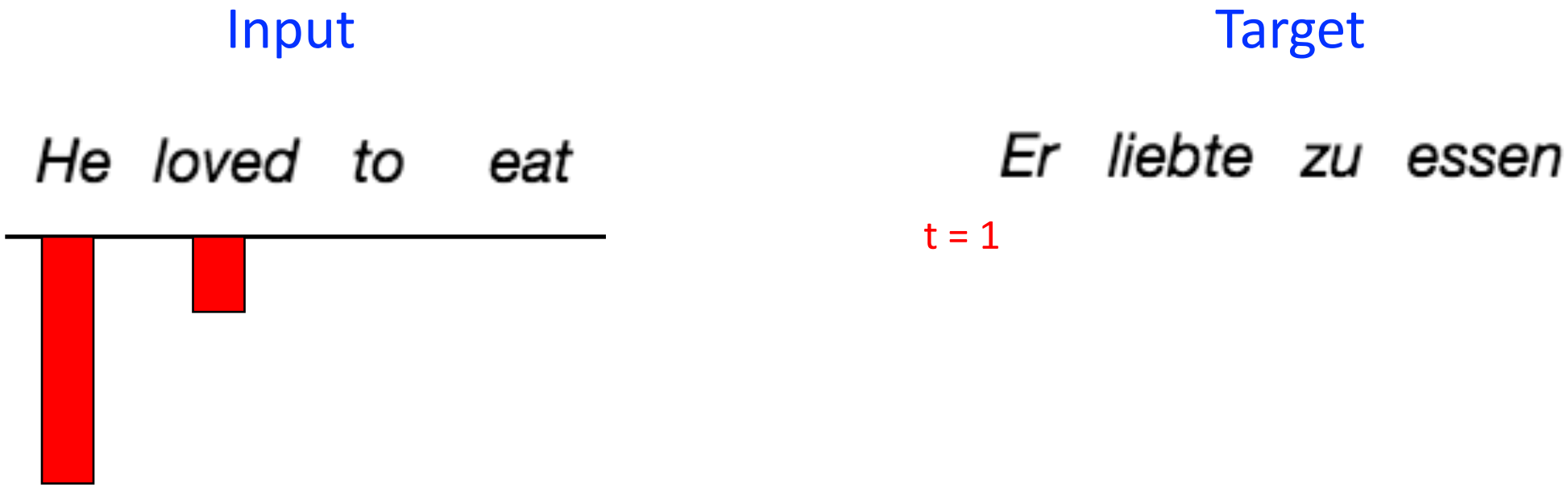


Limitations: a target word relies on information about one input word and “hard attention” is not differentiable

https://deeplearning.cs.cmu.edu/F21/document/slides/lec18.attention.pdf

Idea to Preserve Performance for Long Sentences: **Attention**

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input



Idea to Preserve Performance for Long Sentences: **Attention**

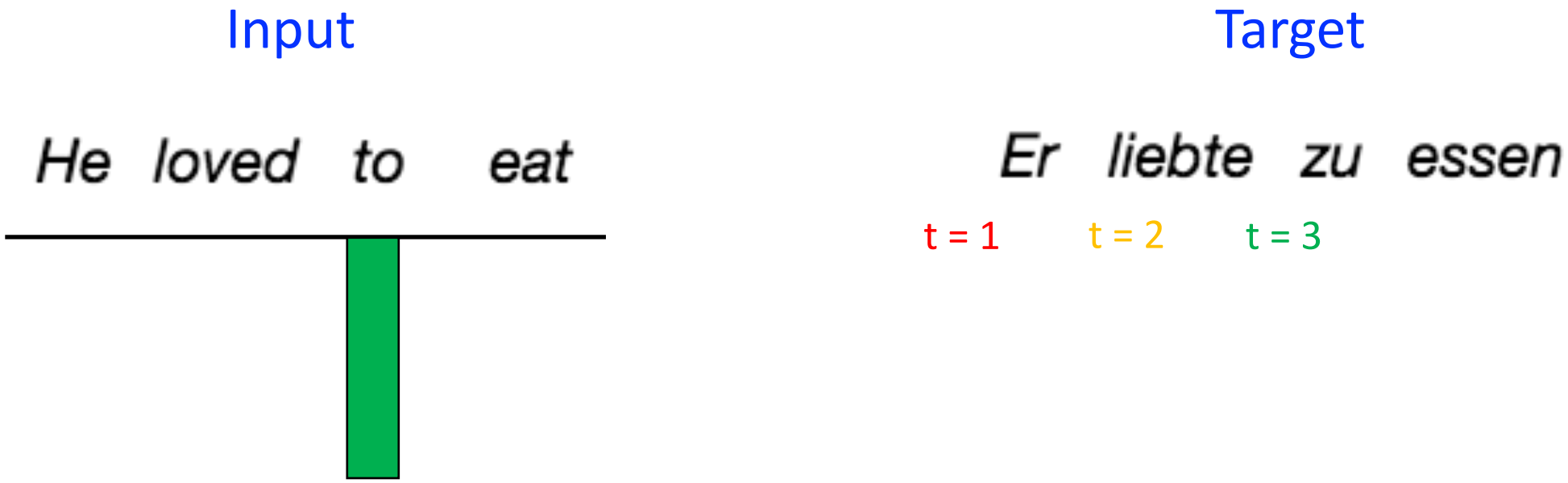
Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input

Input

Target

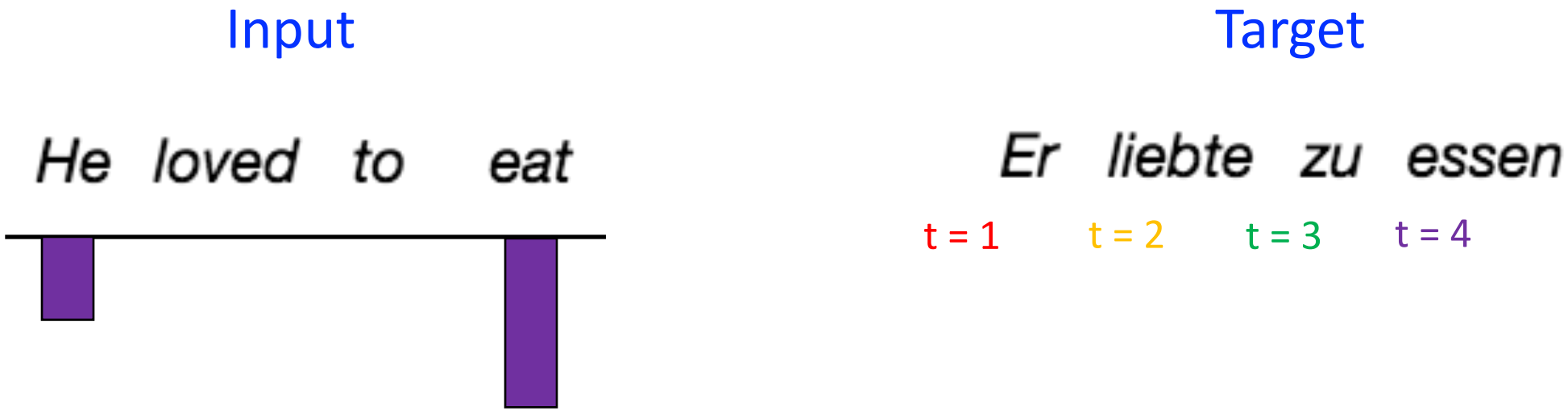
Idea to Preserve Performance for Long Sentences: **Attention**

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input



Idea to Preserve Performance for Long Sentences: **Attention**

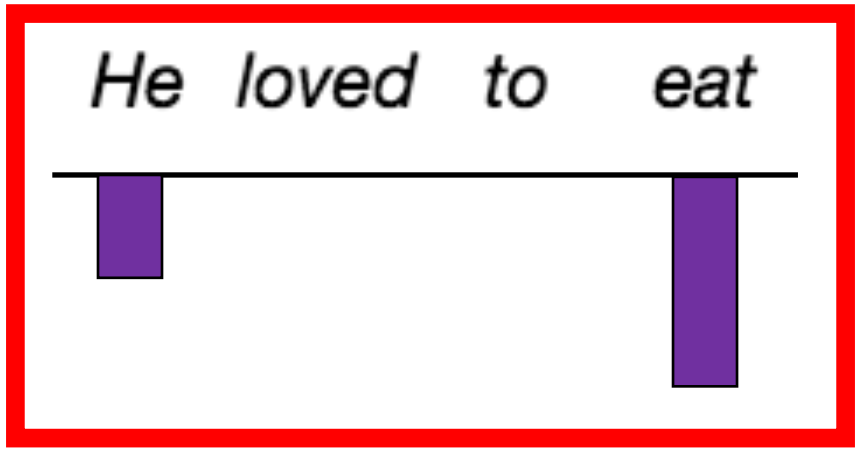
Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input



“Soft” Attention: Challenge

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input

Input



Target

Er liebte zu essen

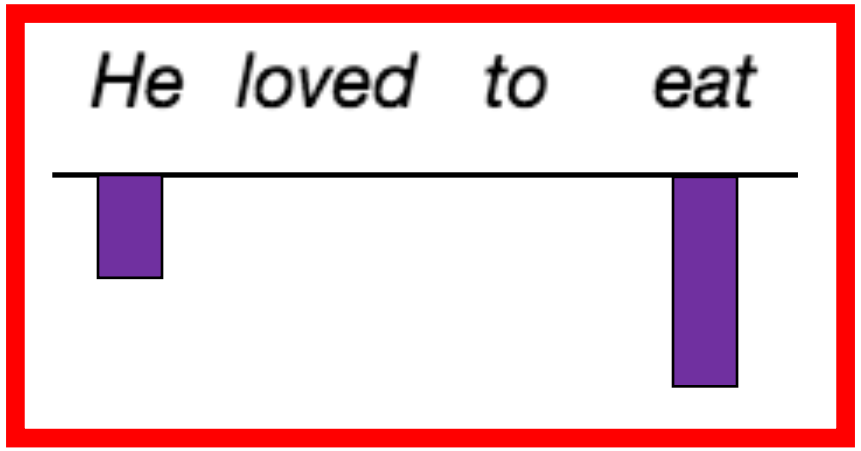
t = 1 t = 2 t = 3 t = 4

How should weights be chosen for each input?

“Soft” Attention: Challenge

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input

Input



Target

Er liebte zu essen

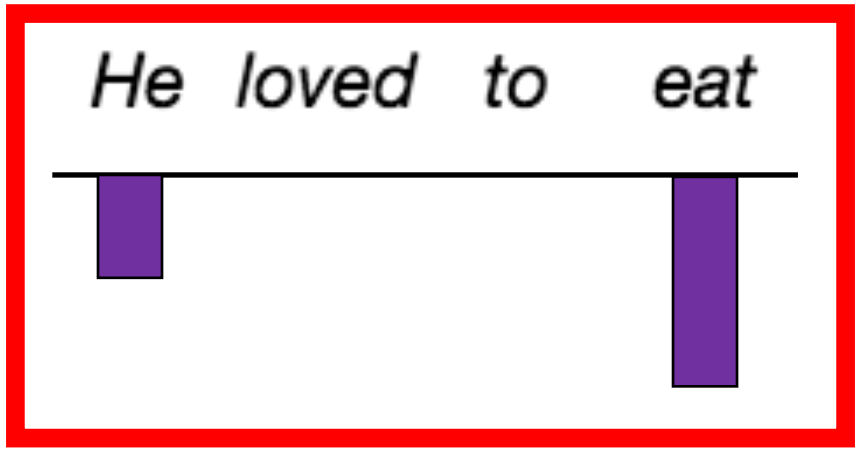
t = 1 t = 2 t = 3 t = 4

Could collect manual annotations and then incorporate into the loss function that predicted weights should match ground truth weights… but this approach is impractical

“Soft” Attention: Challenge

Decoder decides which inputs are needed for prediction at each time step; e.g., “soft attention” uses a weighted combination of the input

Input



Instead, have the model learn how to weight each input!

Target

Er liebte zu essen

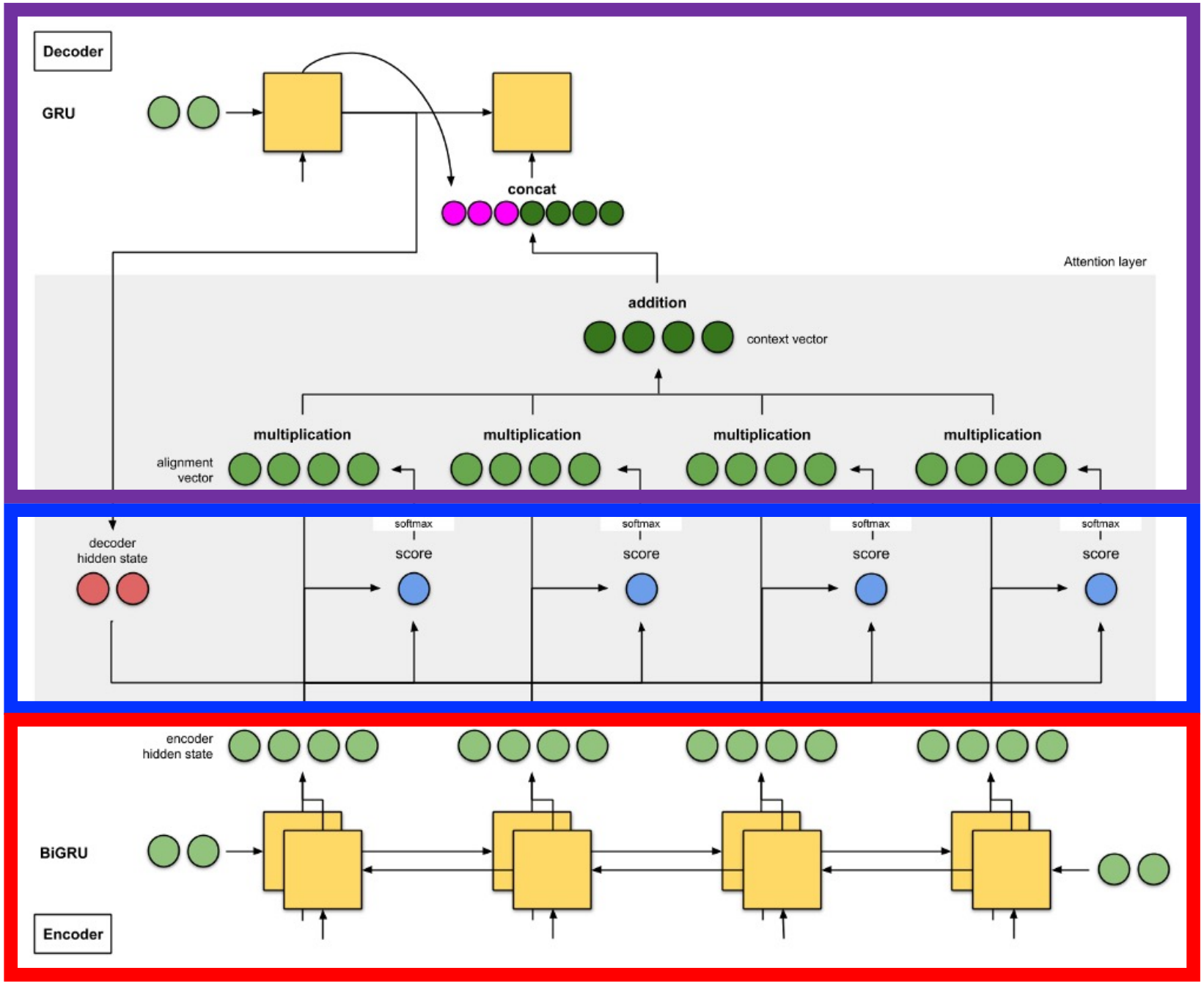
t = 1 t = 2 t = 3 t = 4

Solution

3. At each decoder time step, a prediction is made based on the weighted sum of the inputs

2. At each decoder time step, attention weights are computed that determine each input's relevance for the prediction

1. Encoder produces hidden state for every input

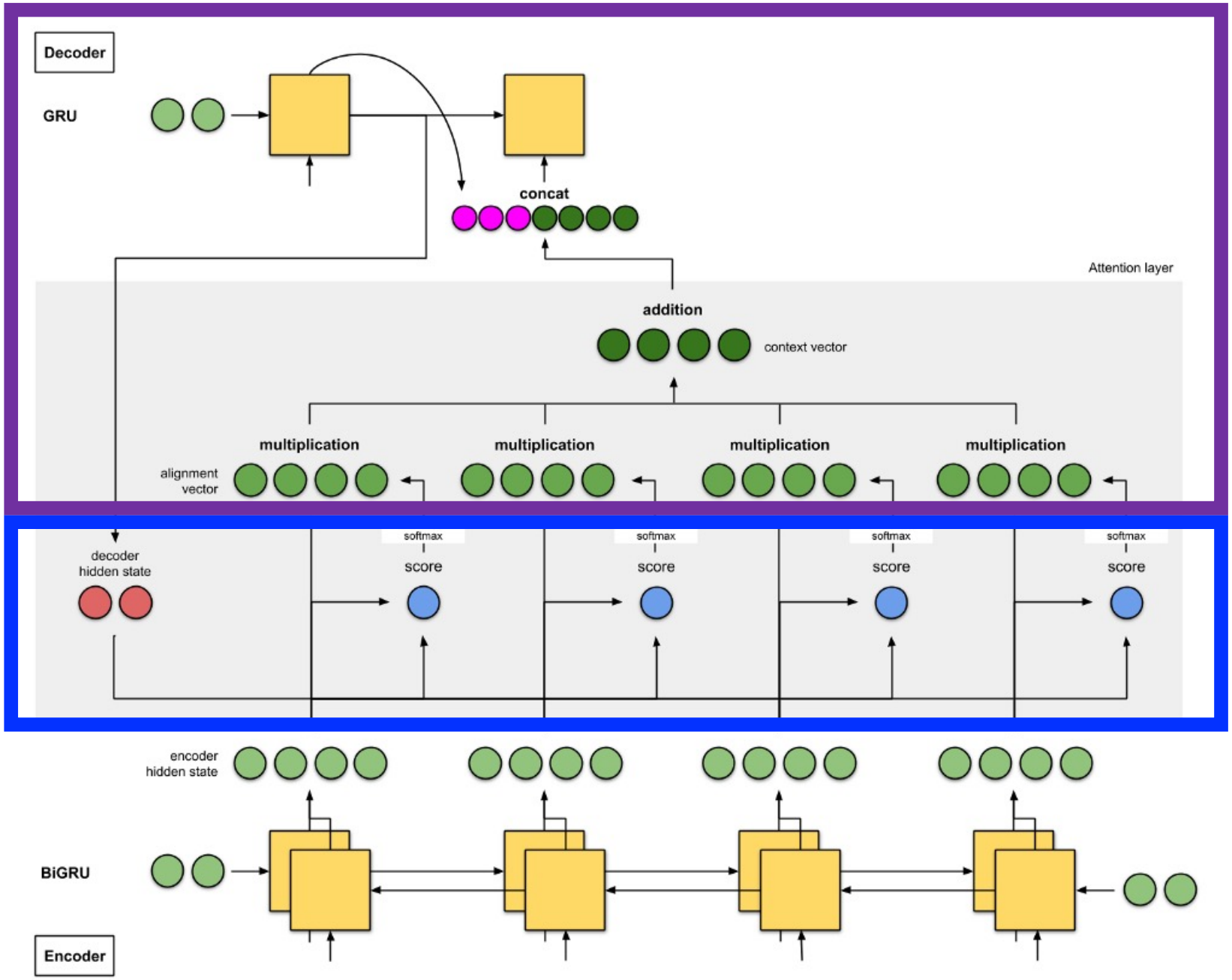


https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Solution

3. At each decoder time step, a prediction is made based on the weighted sum of the inputs

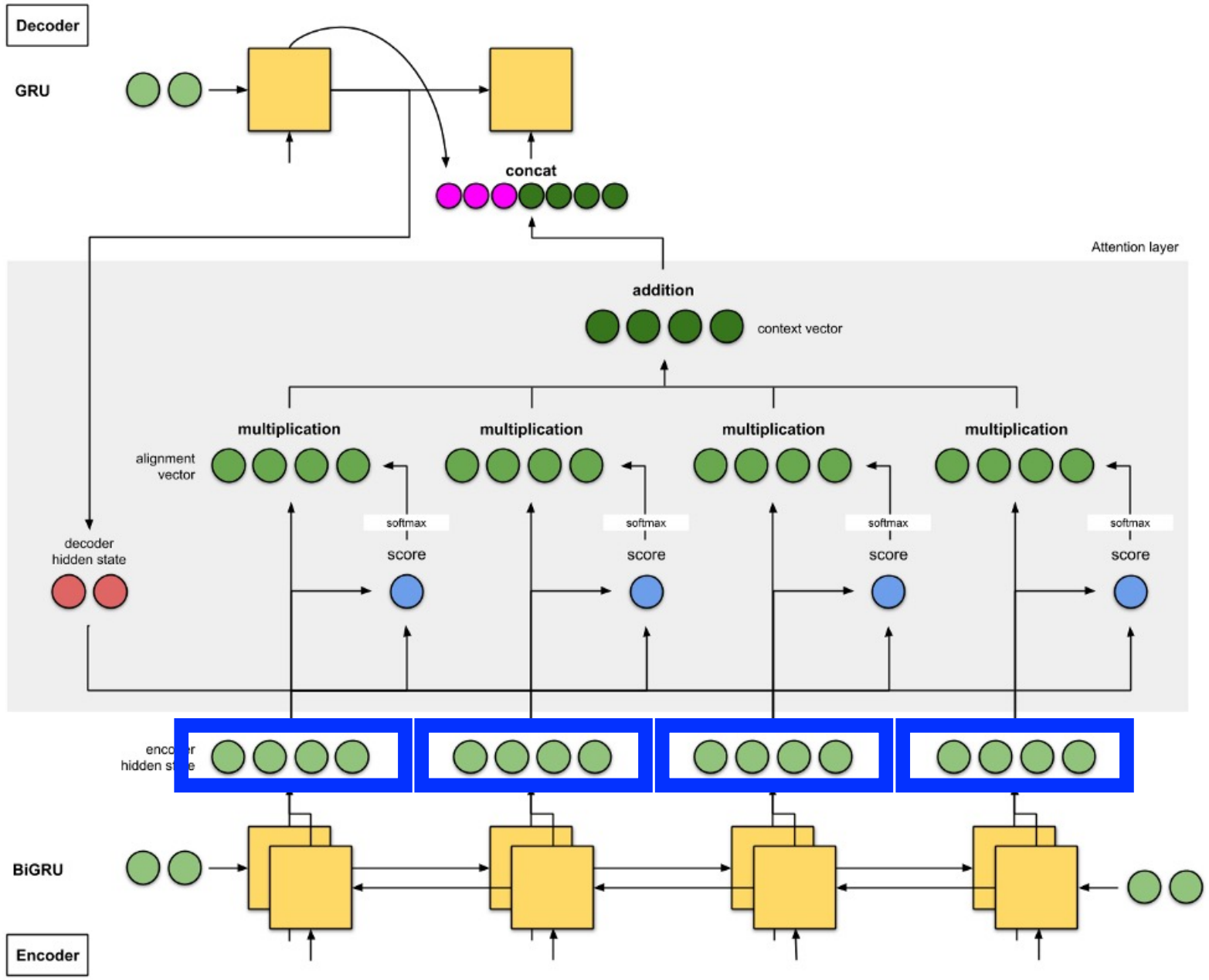
2. At each decoder time step, attention weights are computed that determine each input's relevance for the prediction



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

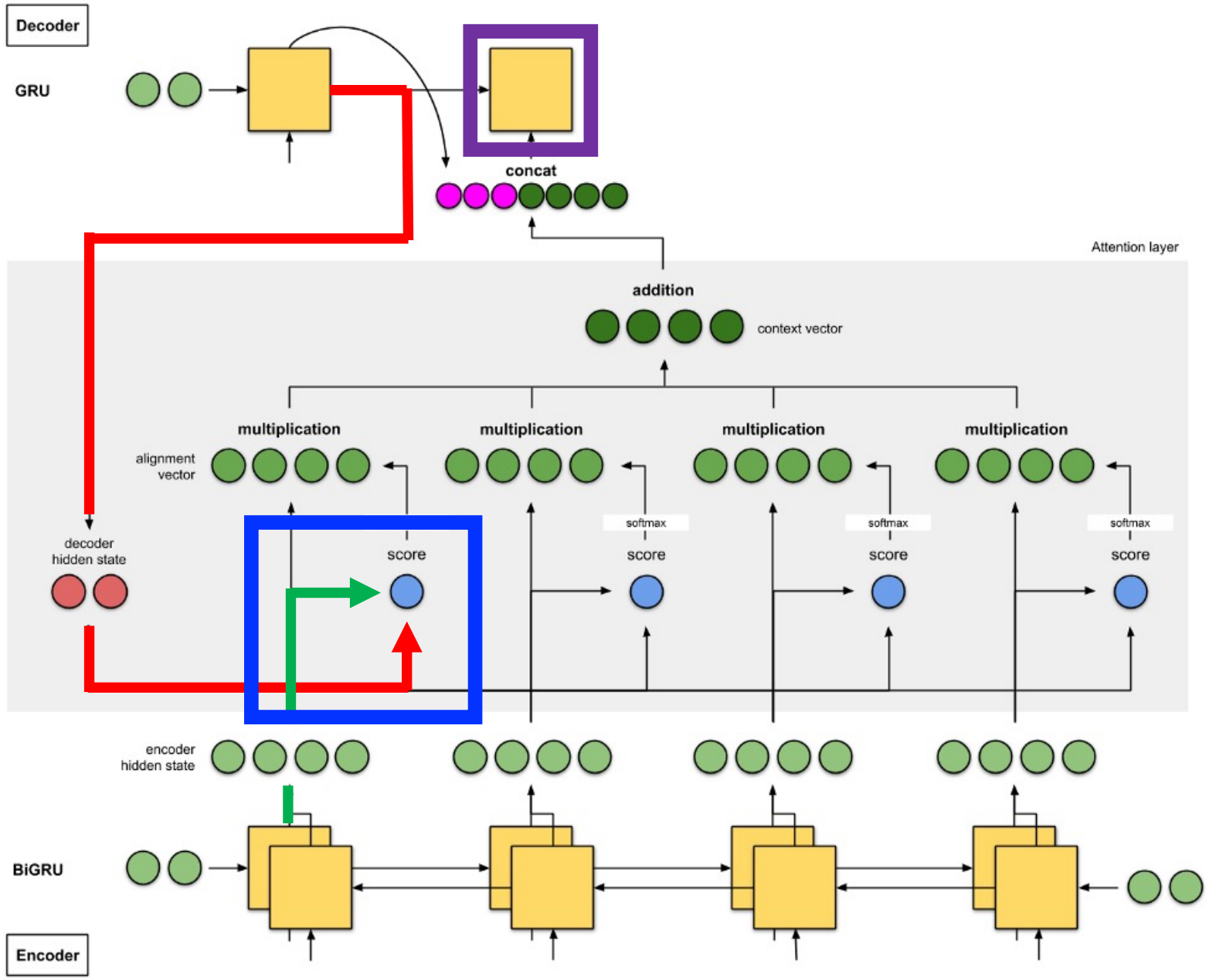
How many inputs are in this example?



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

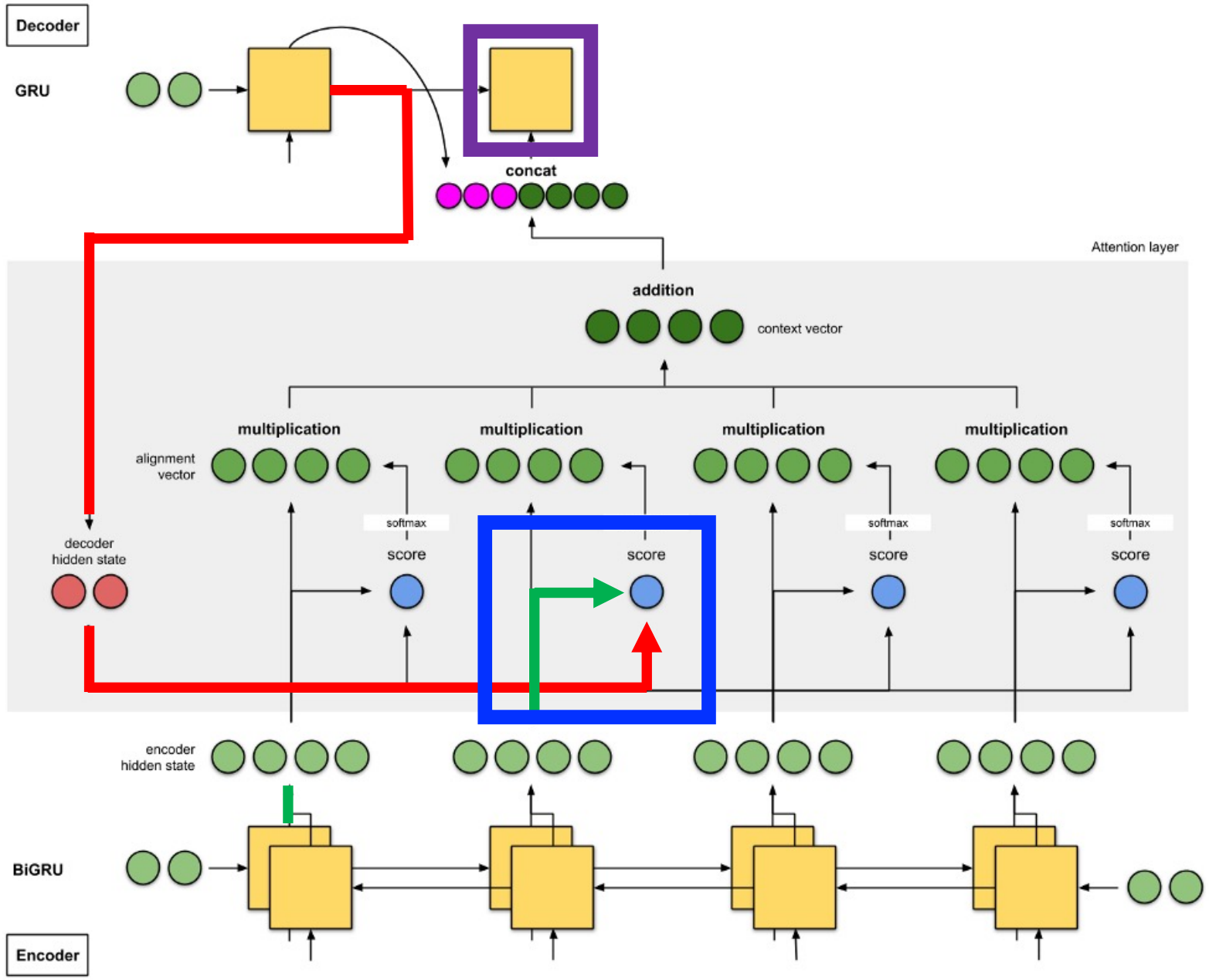
At each decoder time step, the similarity between the decoder's hidden state and each input's hidden state is computed to decide each input's score at the time step



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

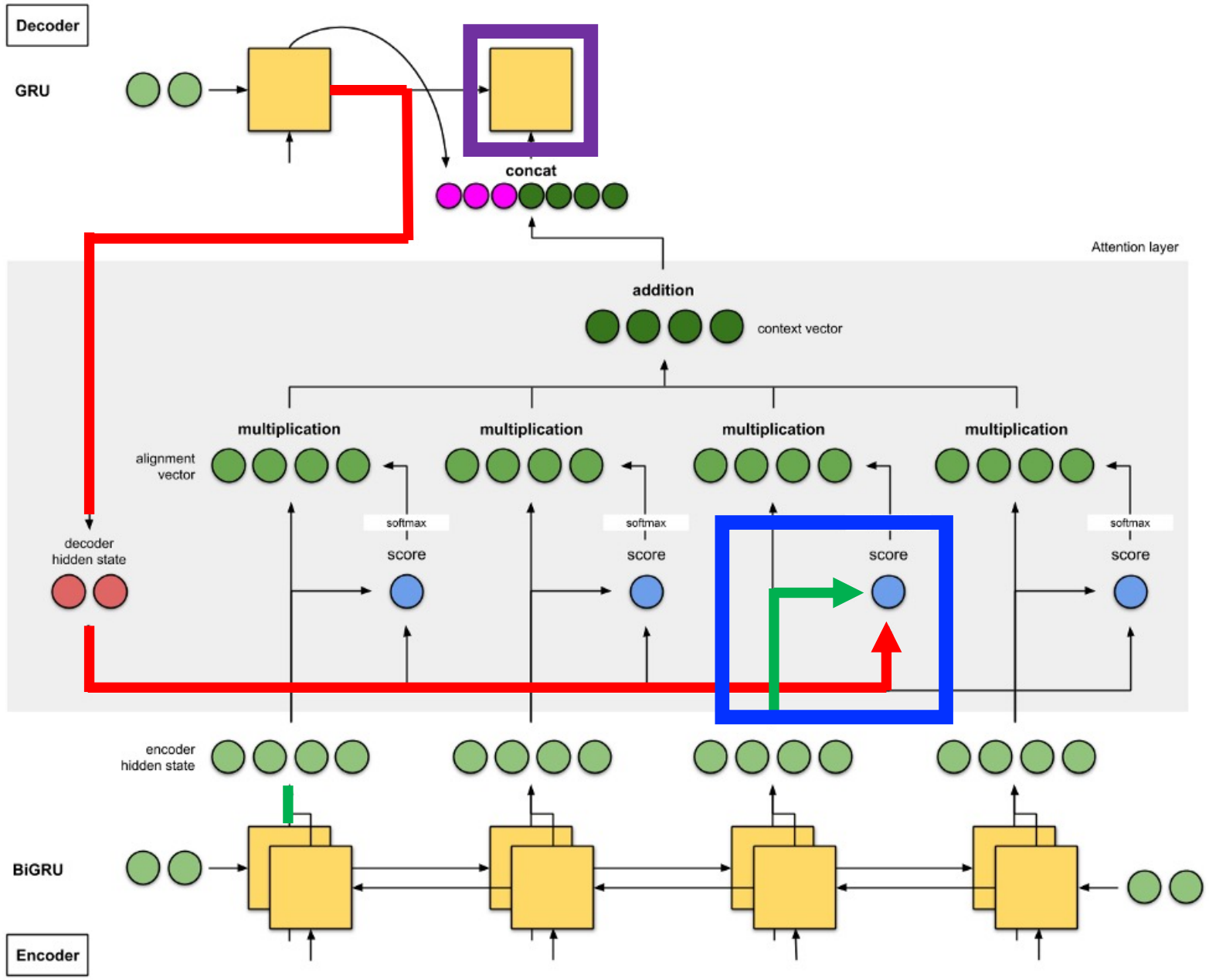
At each decoder time step, the similarity between the decoder's hidden state and each input's hidden state is computed to decide each input's score at the time step



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

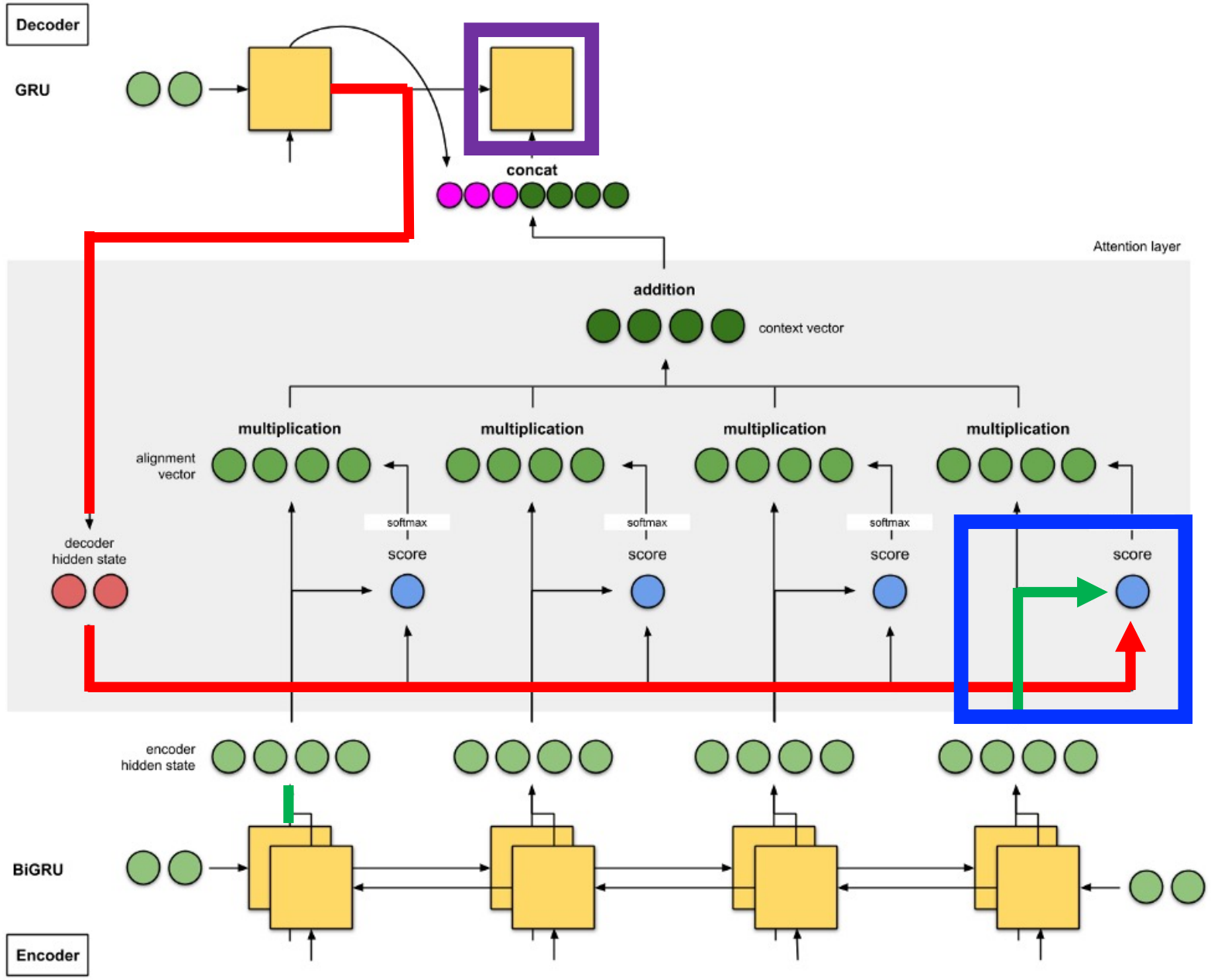
At each decoder time step, the similarity between the decoder's hidden state and each input's hidden state is computed to decide each input's score at the time step



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

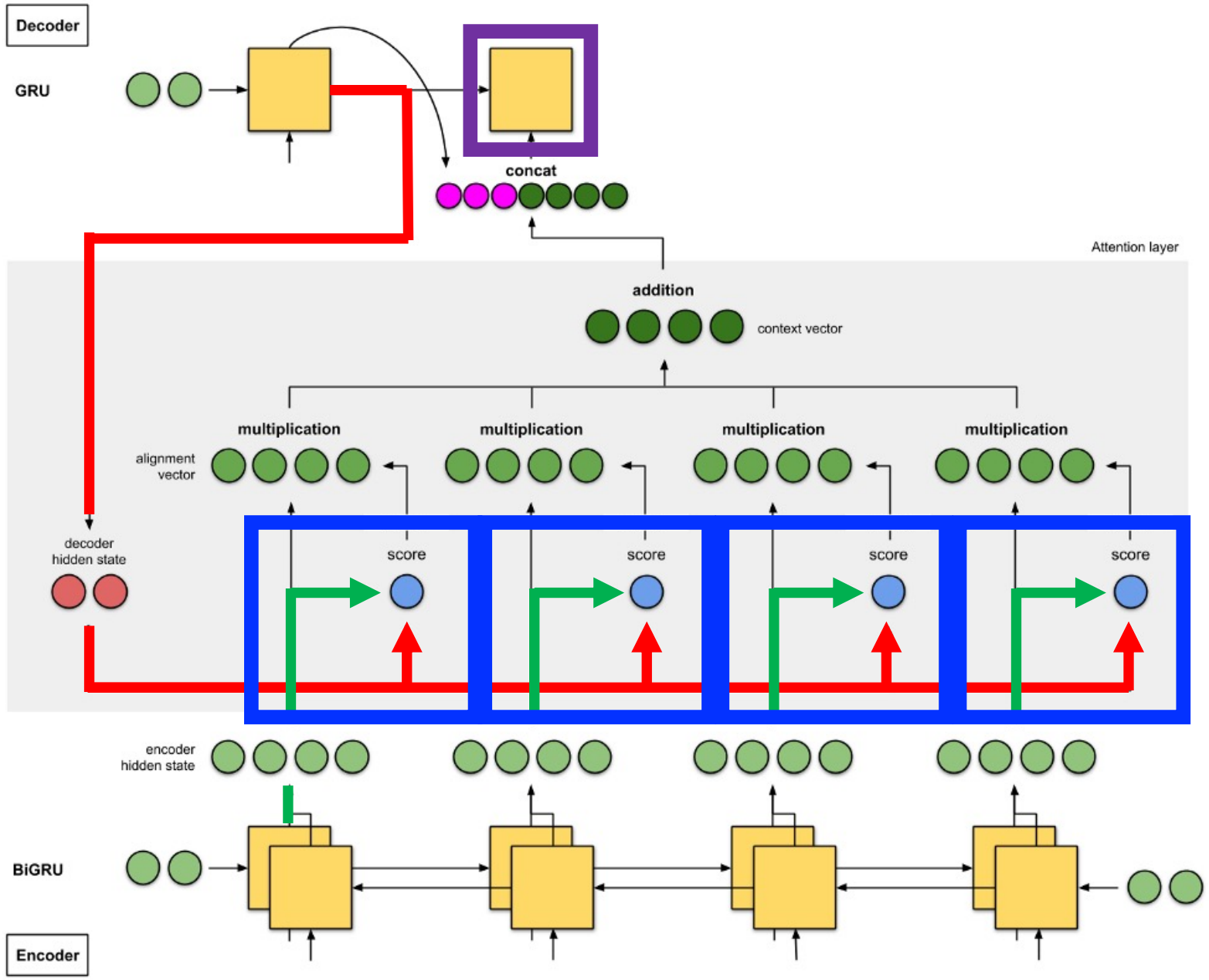
At each decoder time step, the similarity between the decoder's hidden state and each input's hidden state is computed to decide each input's score at the time step



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

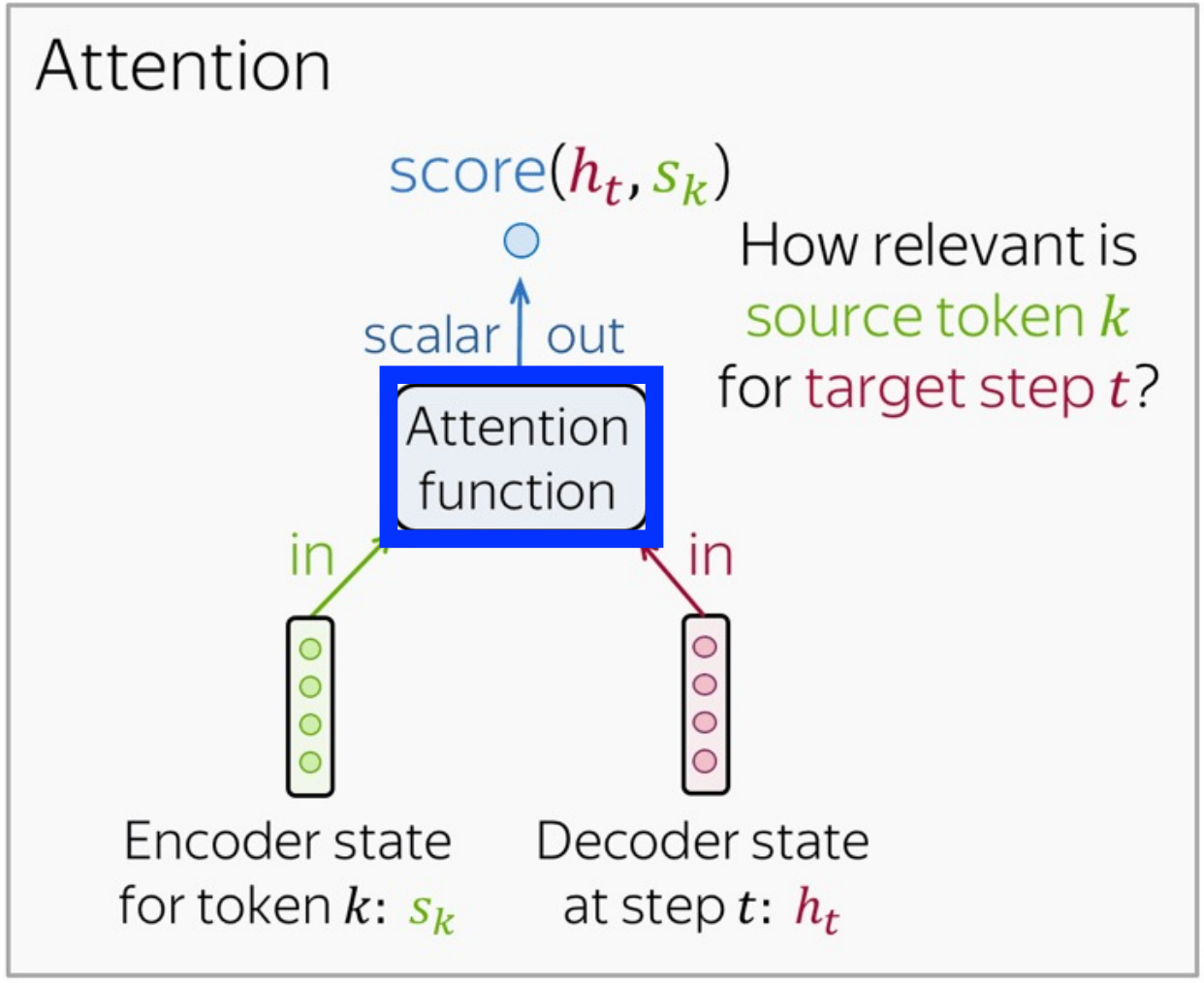
Measuring Each Input's Influence on the Prediction

How to measure the similarity between hidden states of the decoder and input?



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

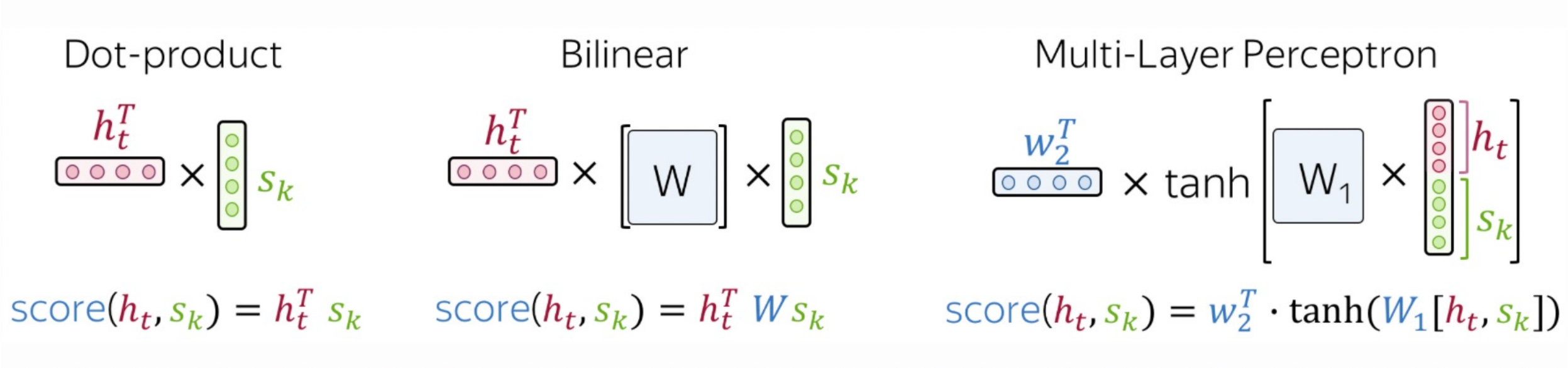
Similarity Measure for Hidden States of the Decoder and Encoder



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Similarity Measure for Hidden States of the Decoder and Encoder

- Many options (function should be differentiable)

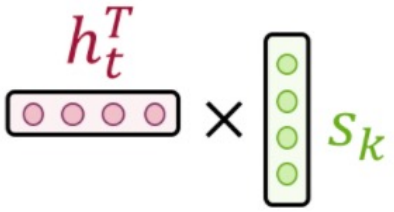


https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Similarity Measure for Hidden States of the Decoder and Encoder

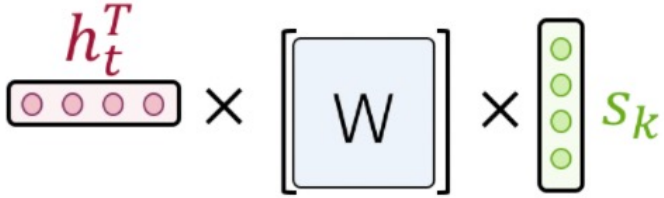
- Many options (function should be differentiable)

Dot-product



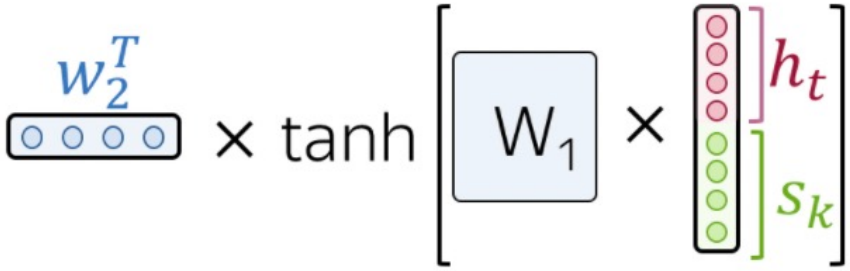
$$\text{score}(h_t, s_k) = h_t^T \, s_k$$

Bilinear



$$\text{score}(h_t, s_k) = h_t^T \, W \, s_k$$

Multi-Layer Perceptron



$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1 [h_t, s_k])$$

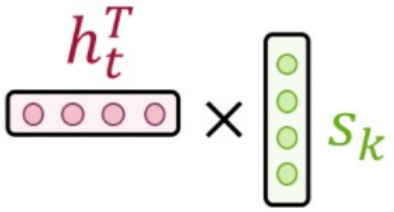
What model parameters must be learned when using dot-product?

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Similarity Measure for Hidden States of the Decoder and Encoder

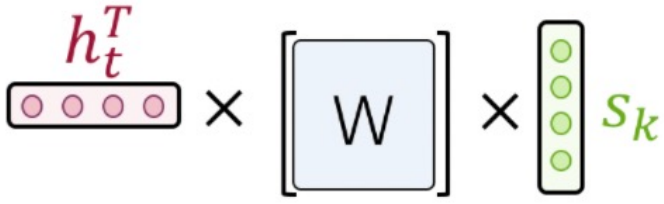
- Many options (function should be differentiable)

Dot-product



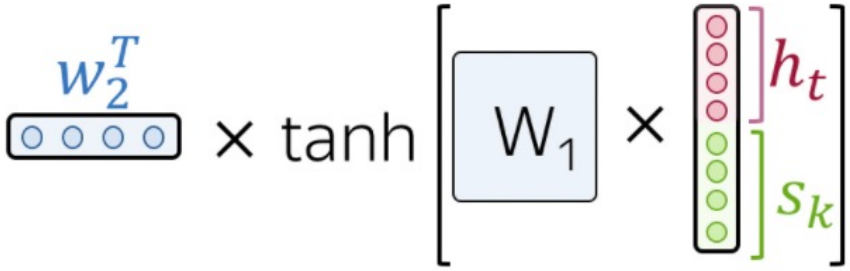
$$\text{score}(h_t, s_k) = h_t^T \, s_k$$

Bilinear



$$\text{score}(h_t, s_k) = h_t^T \, W \, s_k$$

Multi-Layer Perceptron



$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1 [h_t, s_k])$$

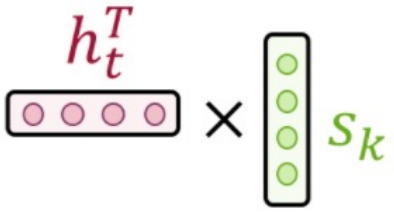
What model parameters must be learned when using bilinear?

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Similarity Measure for Hidden States of the Decoder and Encoder

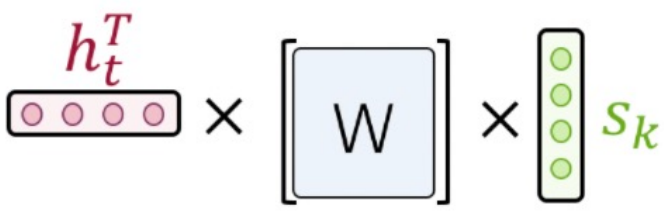
- Many options (function should be differentiable)

Dot-product



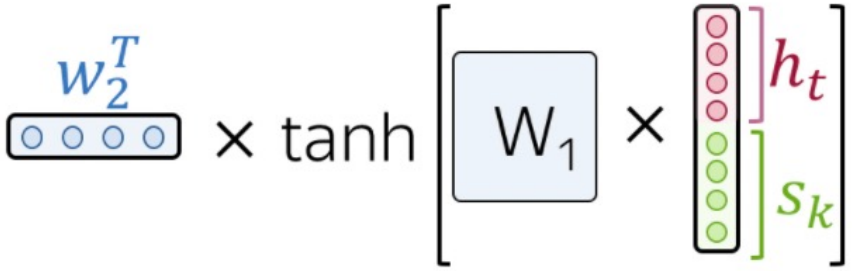
$$\text{score}(h_t, s_k) = h_t^T \, s_k$$

Bilinear



$$\text{score}(h_t, s_k) = h_t^T \, W \, s_k$$

Multi-Layer Perceptron



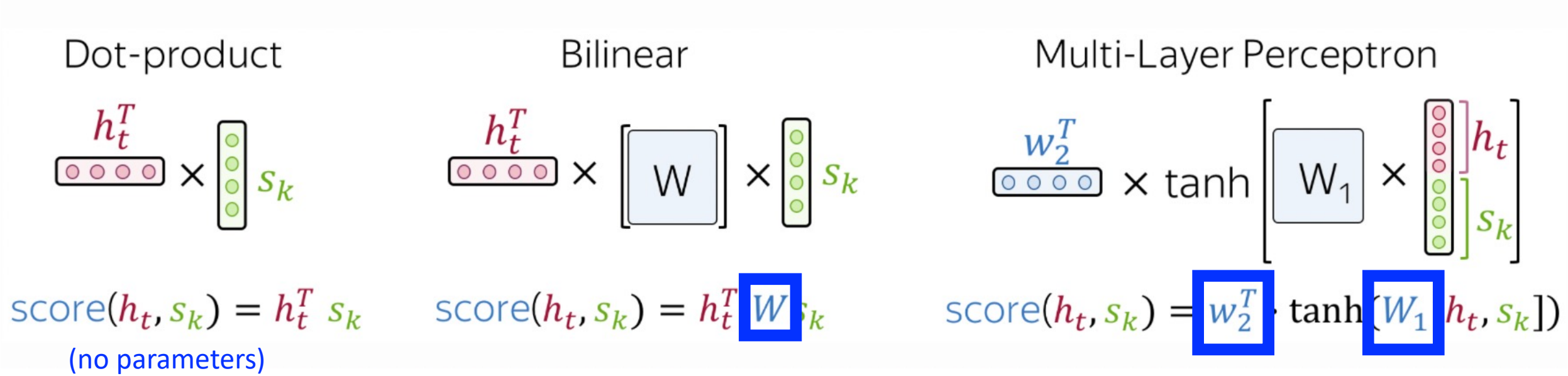
$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1 [h_t, s_k])$$

What model parameters must be learned when using multi-layer perceptron?

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Similarity Measure for Hidden States of the Decoder and Encoder

- Many options (function should be differentiable)

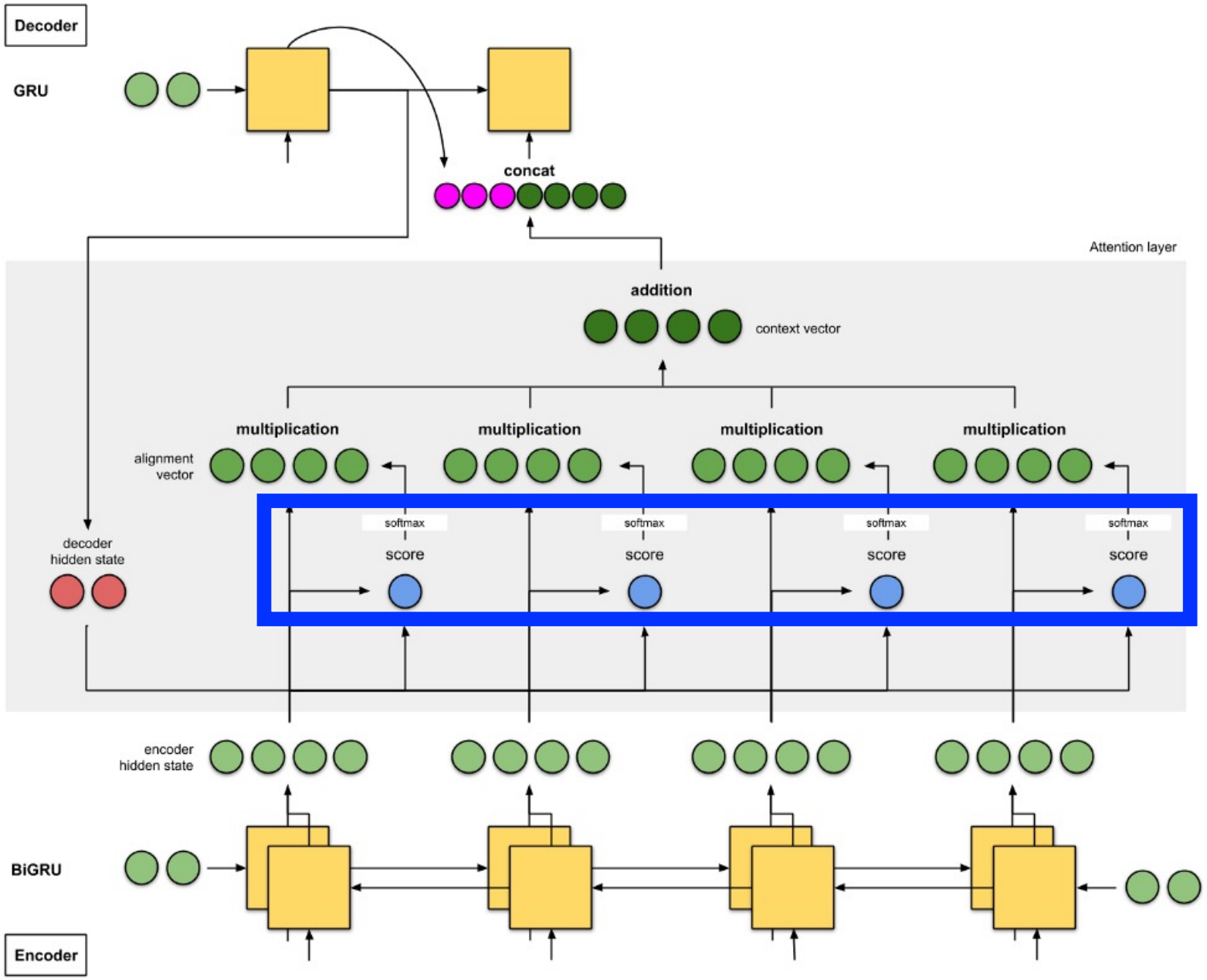


Model parameters that must be learned

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Measuring Each Input's Influence on the Prediction

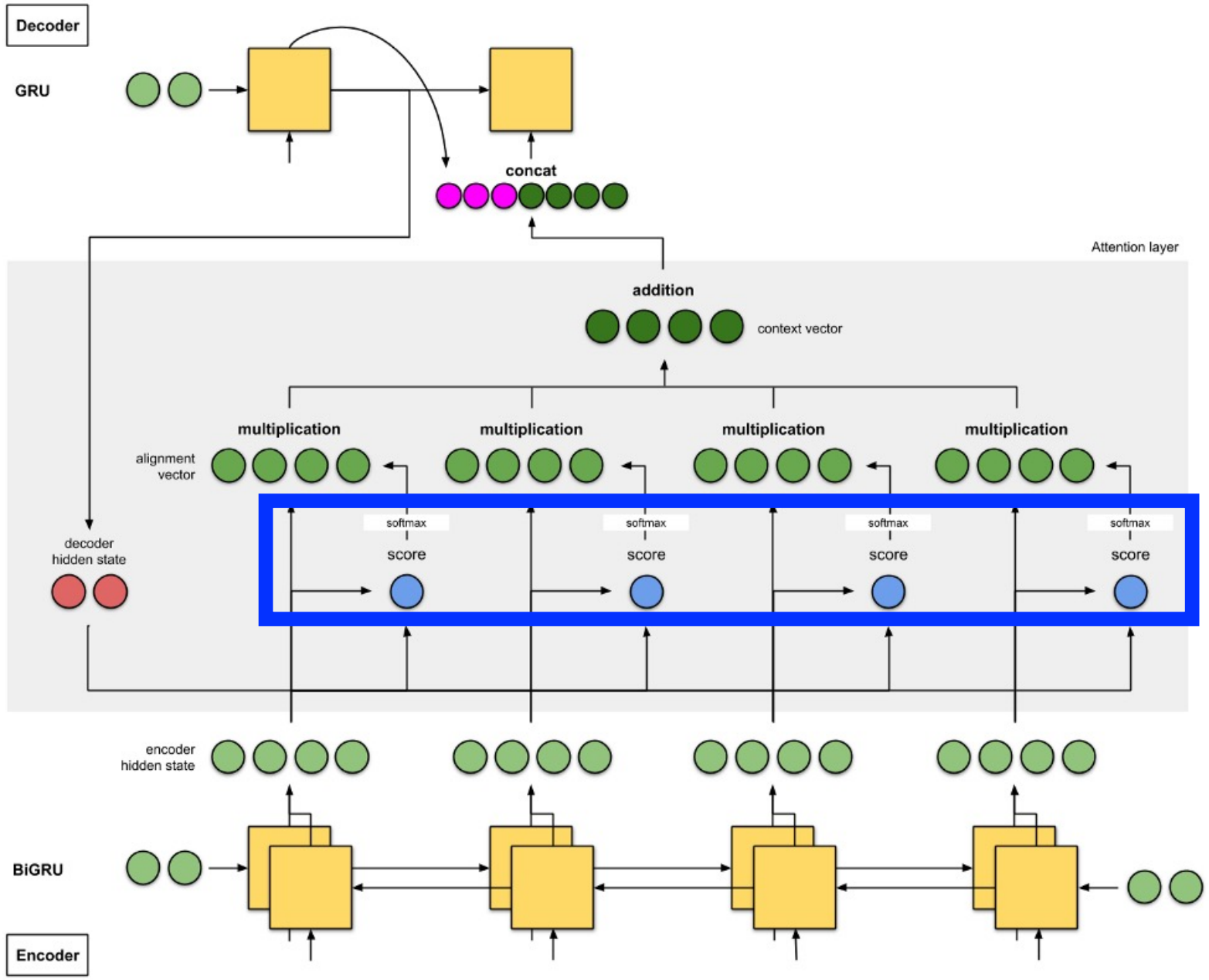
After computing the similarity scores for each input, then apply softmax so all inputs' weights sum to 1



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

We now have our attention weights!

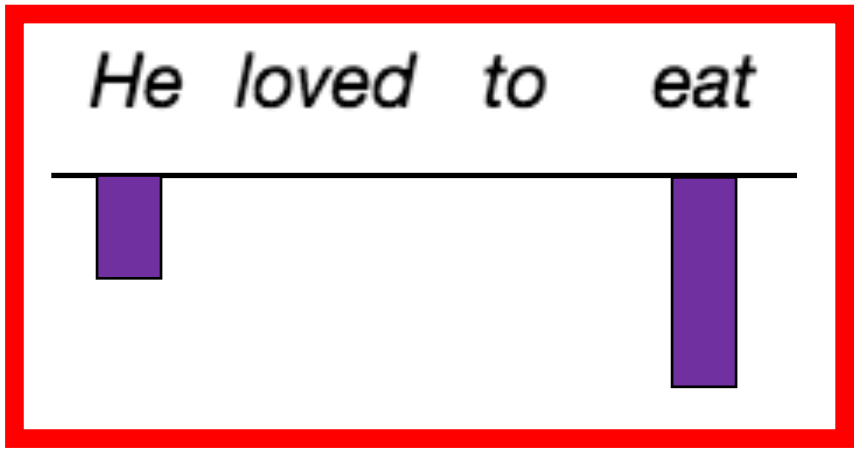


https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Measuring Each Input's Influence on the Prediction

Intuitively:

Input



Target

Er liebte zu essen

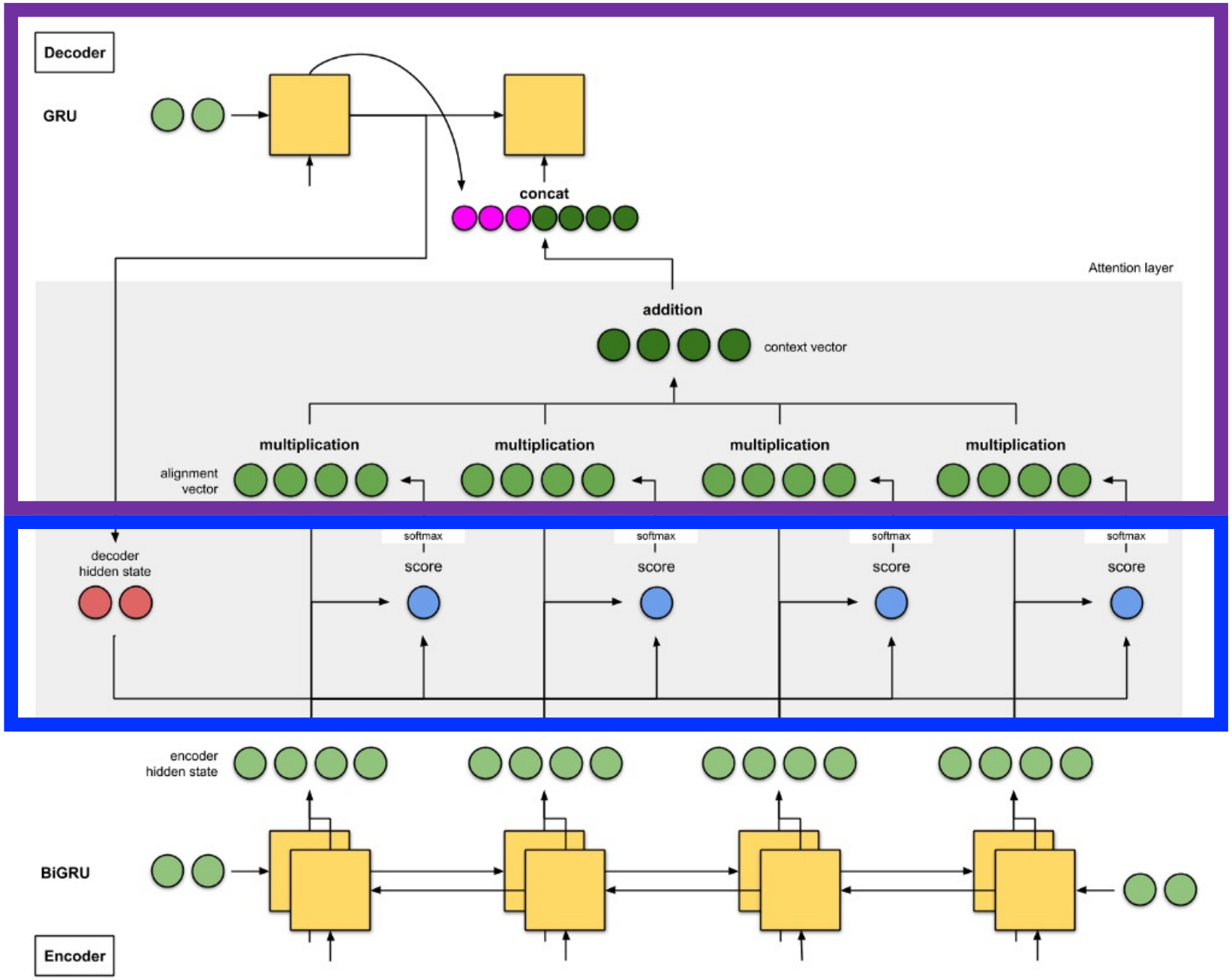
t = 4

The model can weight each input at each time step!

Solution

3. At each decoder time step, a prediction is made based on the weighted sum of the inputs

2. At each decoder time step, attention weights are computed that determine each input's relevance for the prediction



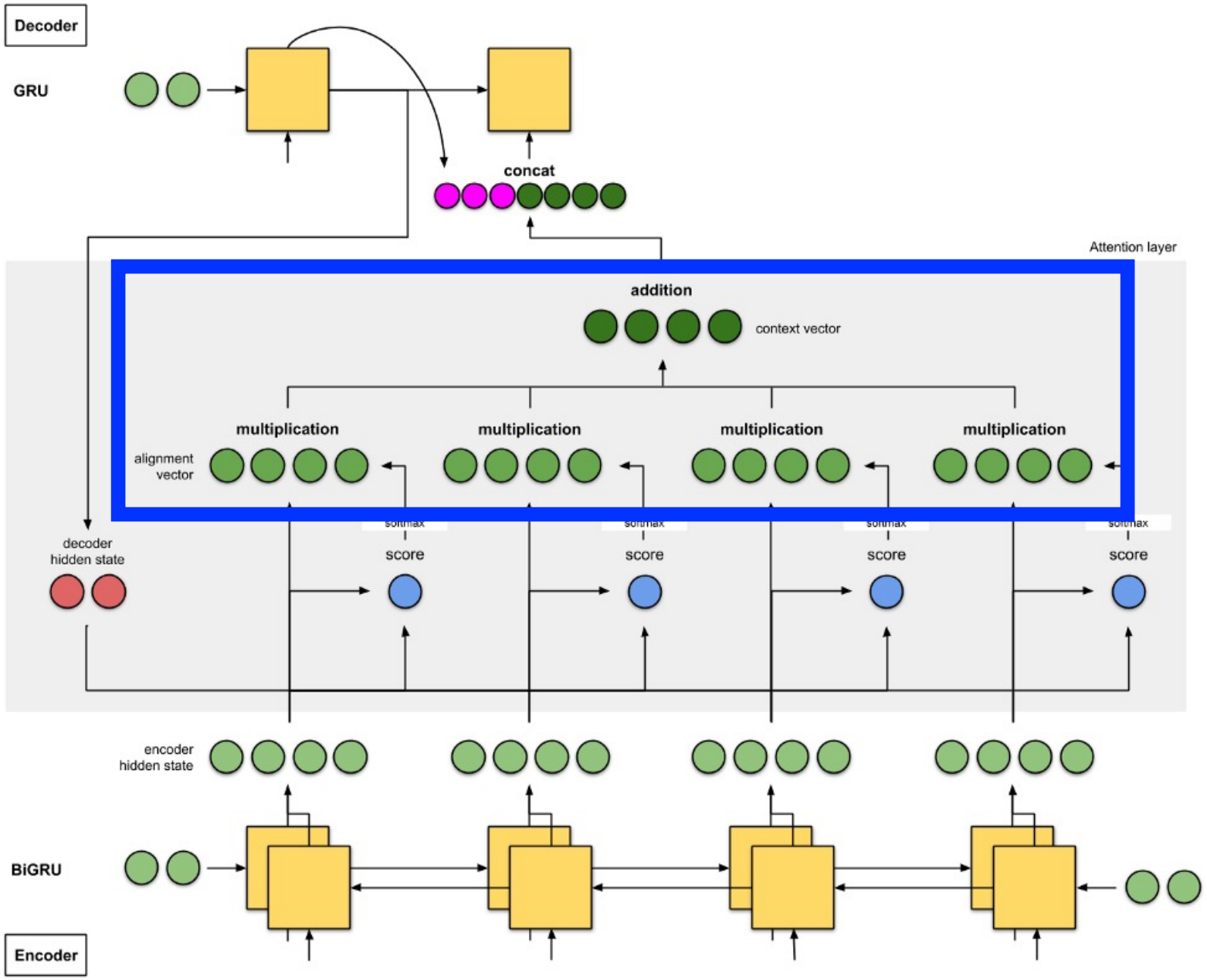
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Word Prediction

We compute at time step *t* for all *n* inputs a weighted sum:

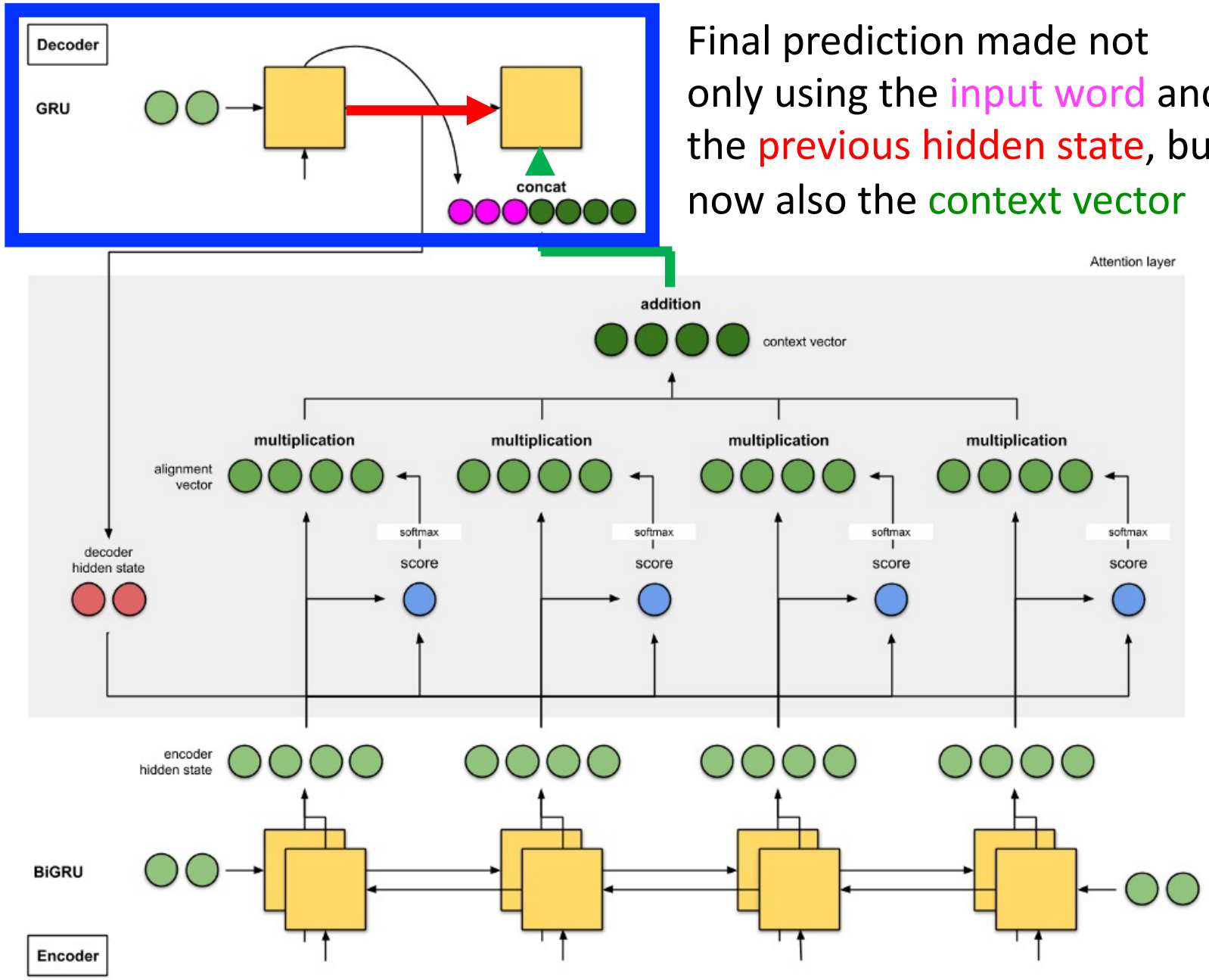
$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$

The influence of inputs are **amplified** for large attention weights and repressed otherwise



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

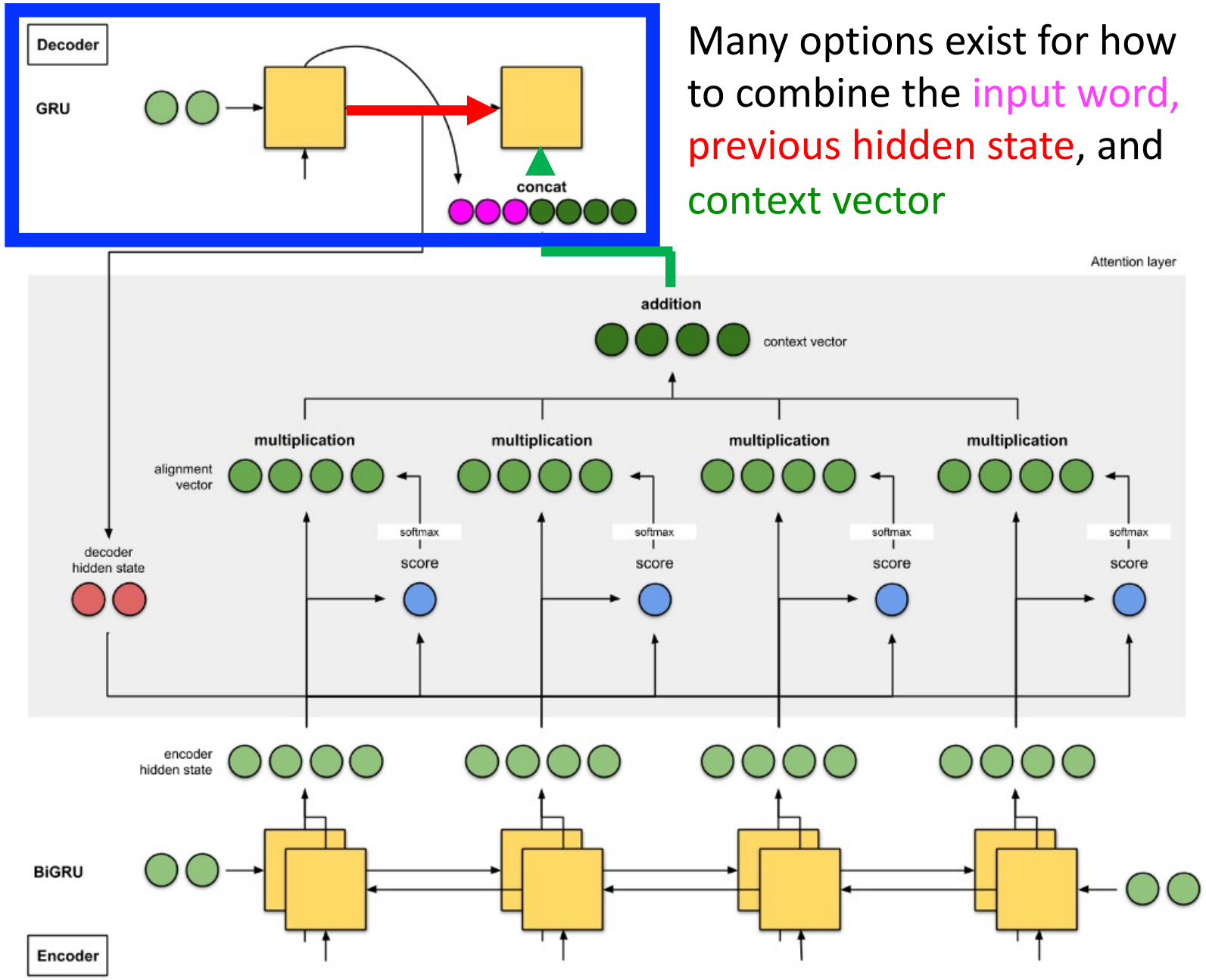
Word Prediction



Final prediction made not only using the input word and the previous hidden state, but now also the context vector

https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Word Prediction



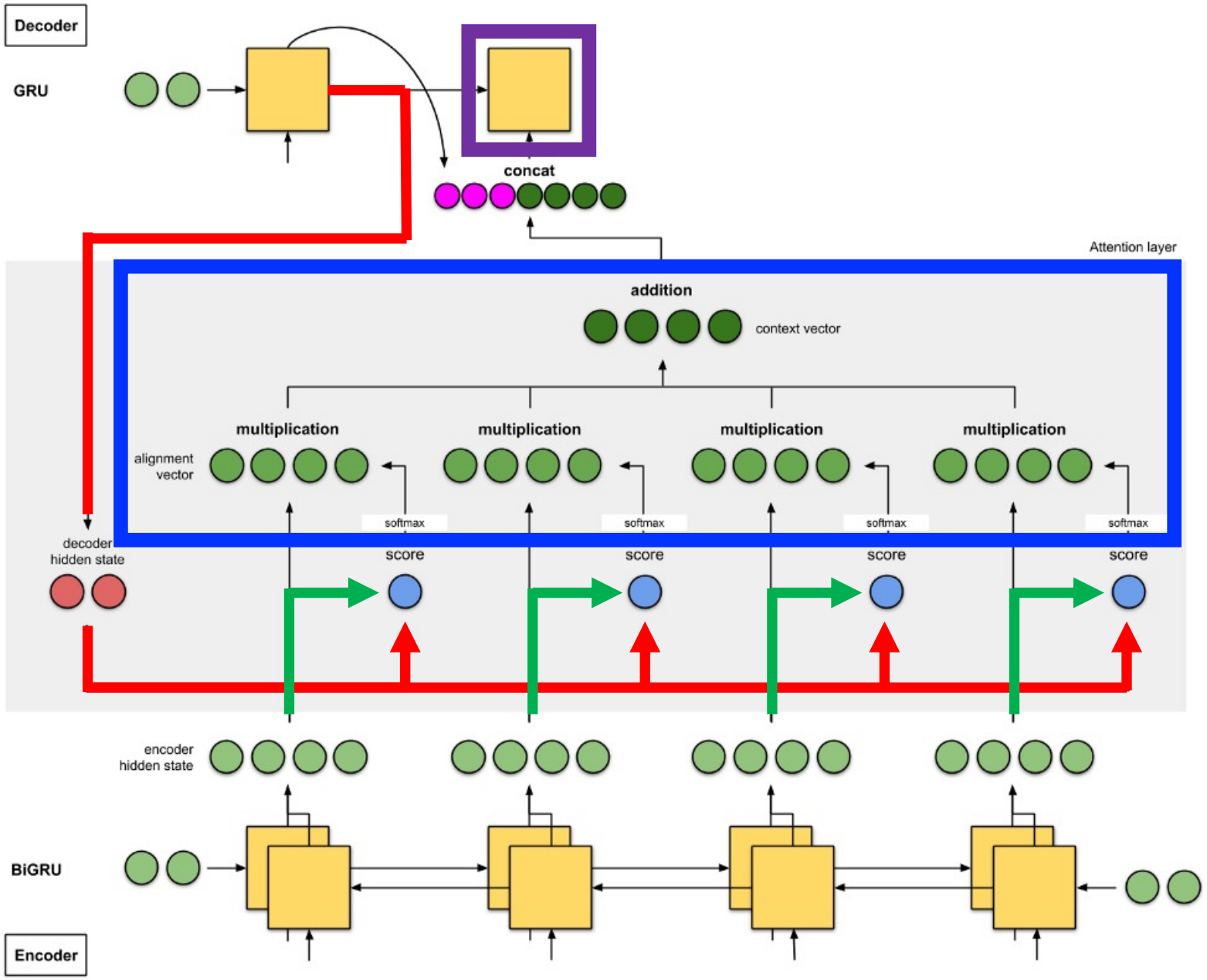
Many options exist for how to combine the input word, previous hidden state, and context vector

https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Solution

What stays the same at each decoder time step?
- input's hidden state

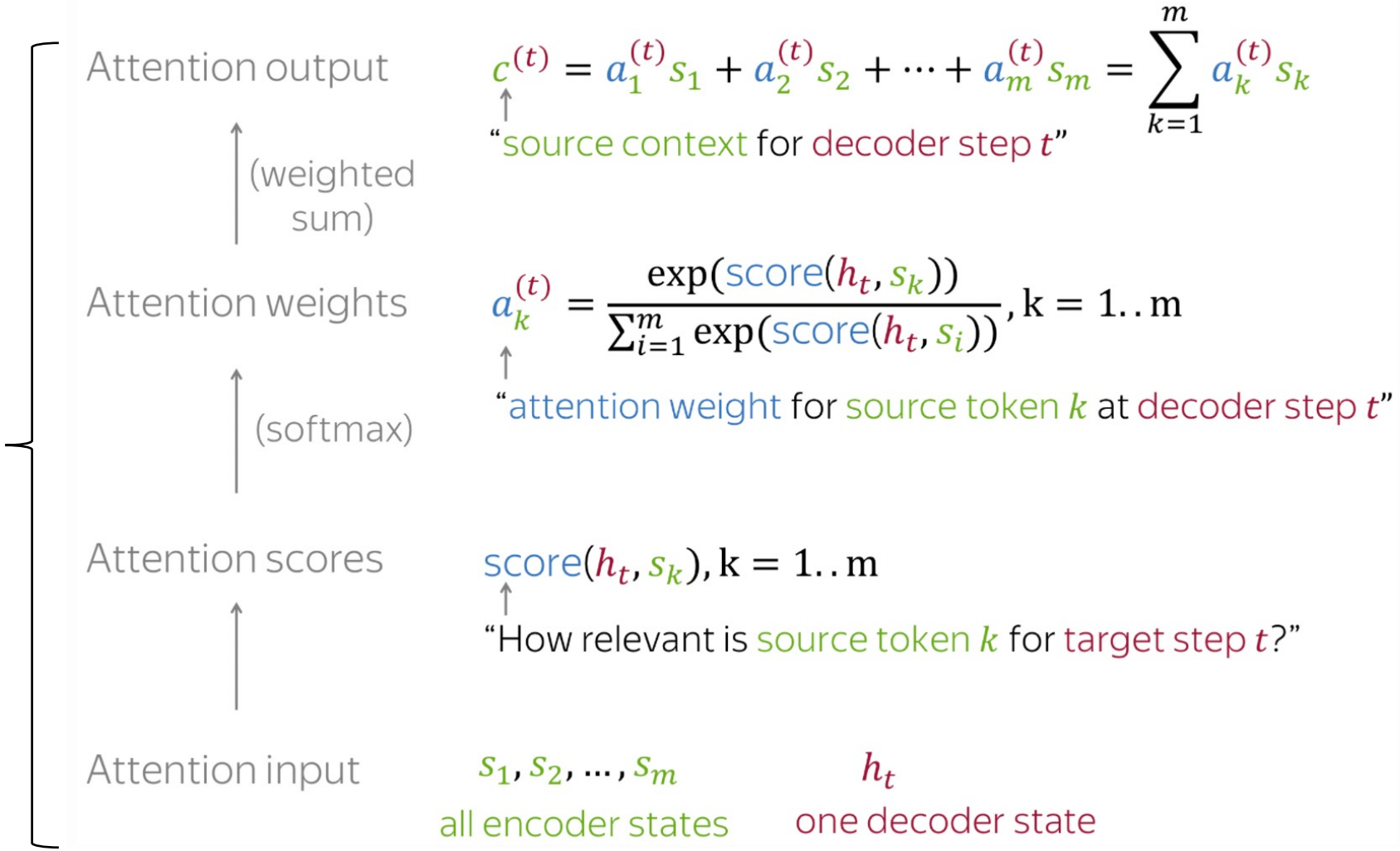
What changes at each decoder time step?
- decoder's hidden state
- and so attention weights and context vector



https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Summary: Attention (Computations at Each Decoder Step)

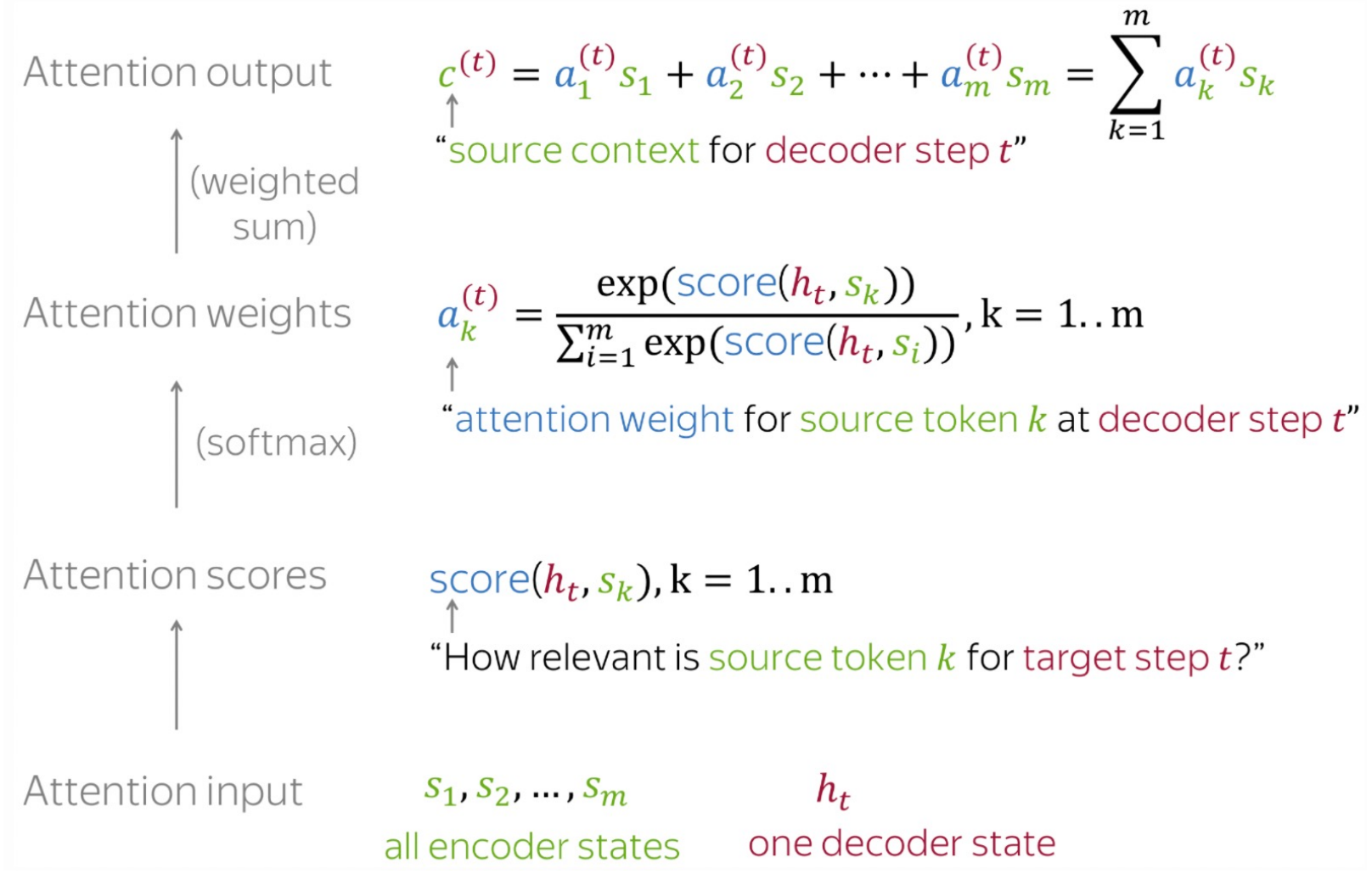
Decoder decides which inputs are needed for prediction at each time step with "soft attention", which results in a weighted combination of the input



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Summary: Attention (Computations at Each Decoder Step)

All parts are differentiable which means end-to-end training is possible



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Popular Choices for Encoding Input

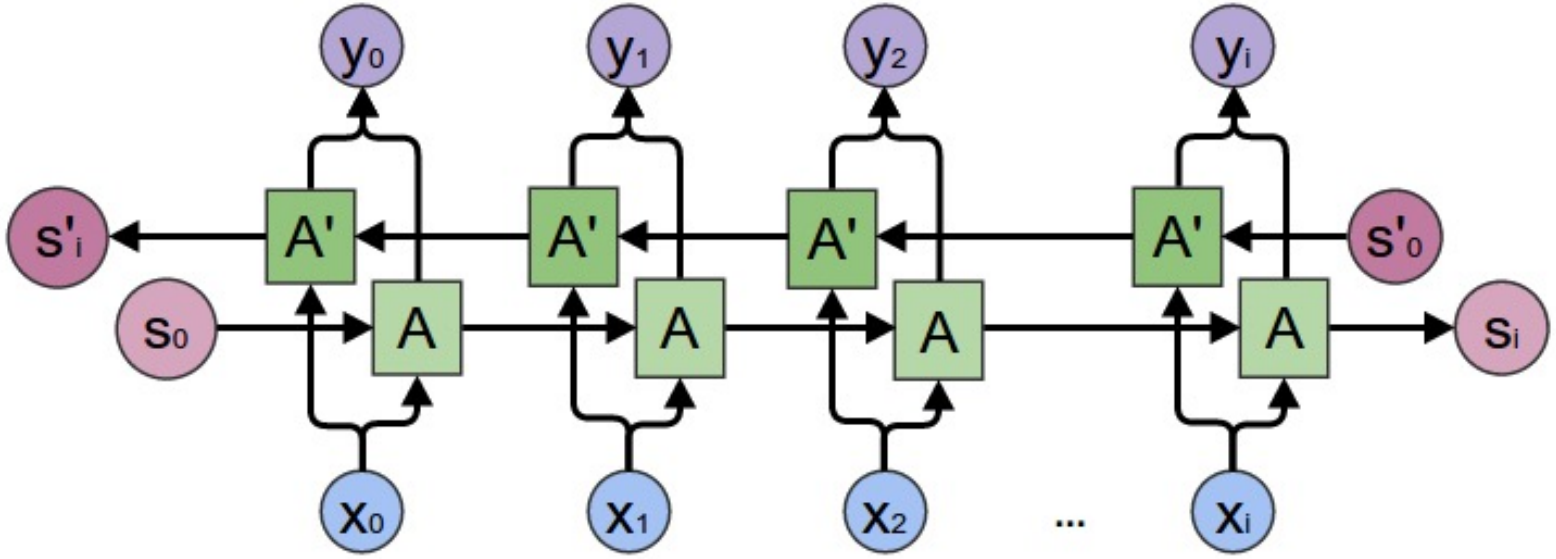
- Bi-directional RNN
- Stacked RNNs

Popular Choices for Encoding Input

- Bi-directional RNN
- Stacked RNNs

Many Options for How to Encode Input

- Two RNNs where input is fed forward and backward respectively and then the hidden states (typically) are concatenated into a hidden state

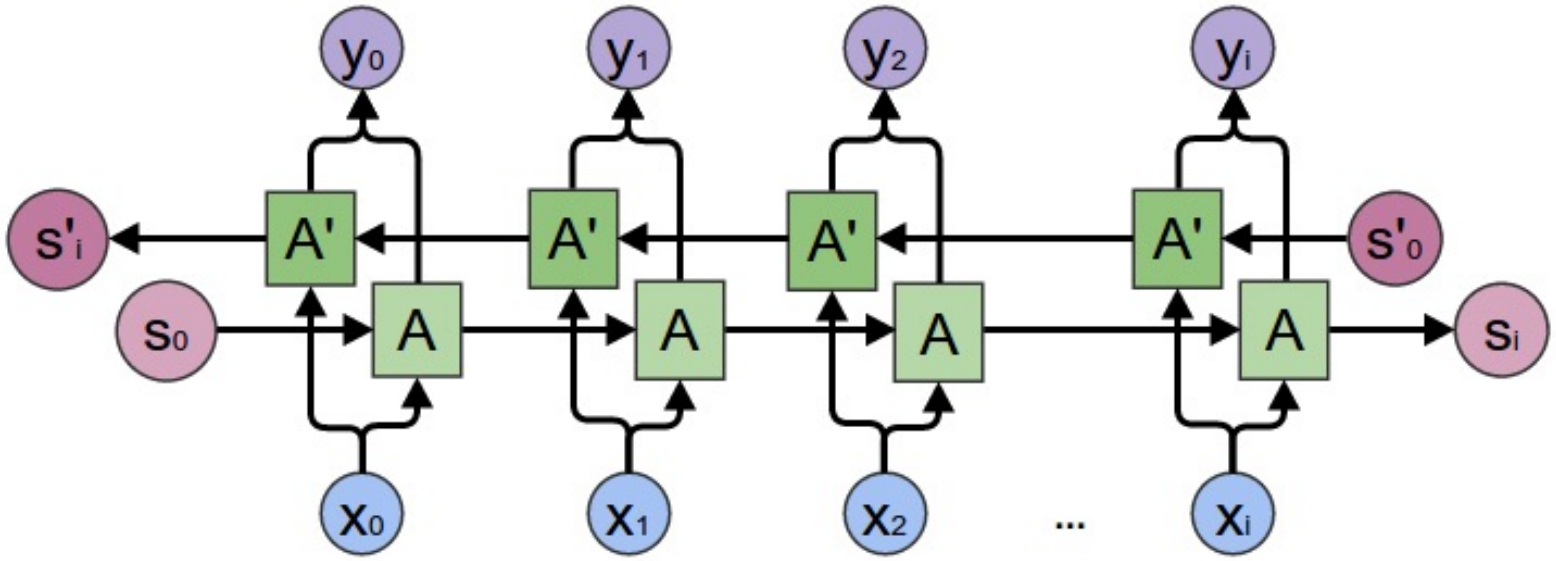


What are advantages of a bi-directional RNN compared to a single RNN?

https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

Many Options for How to Encode Input

- Two RNNs where input is fed forward and backward respectively and then the hidden states (typically) are concatenated into a hidden state

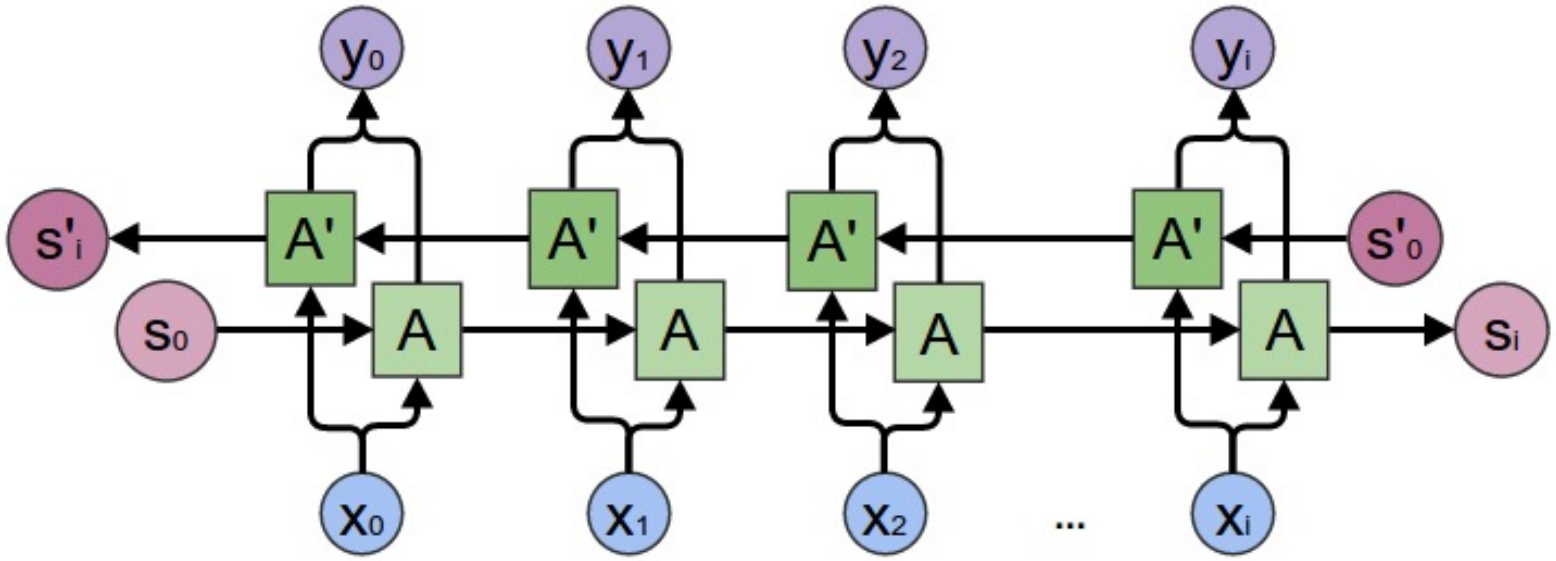


Can use information from the past and **future** to make predictions: e.g., can resolve for "Teddy is a …?" if Teddy refers to a "bear" or former US President Roosevelt

https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

Many Options for How to Encode Input

- Two RNNs where input is fed forward and backward respectively and then the hidden states (typically) are concatenated into a hidden state

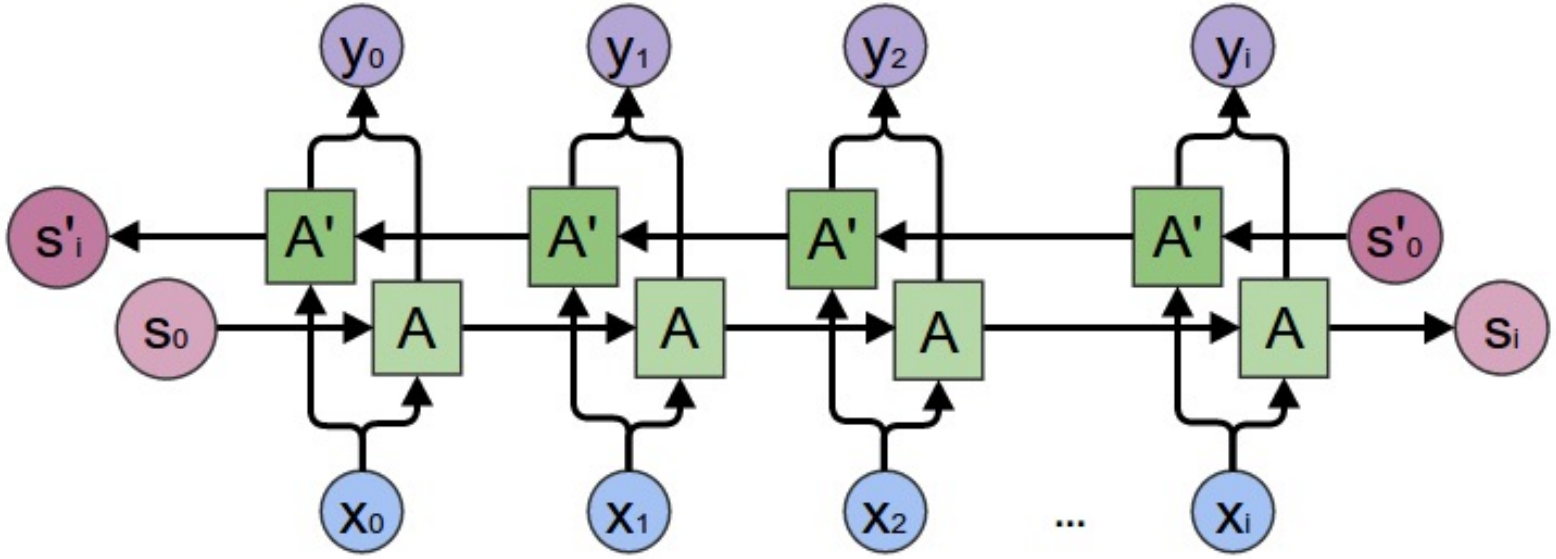


What are disadvantages of a bi-directional RNN compared to a single RNN?

https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

Many Options for How to Encode Input

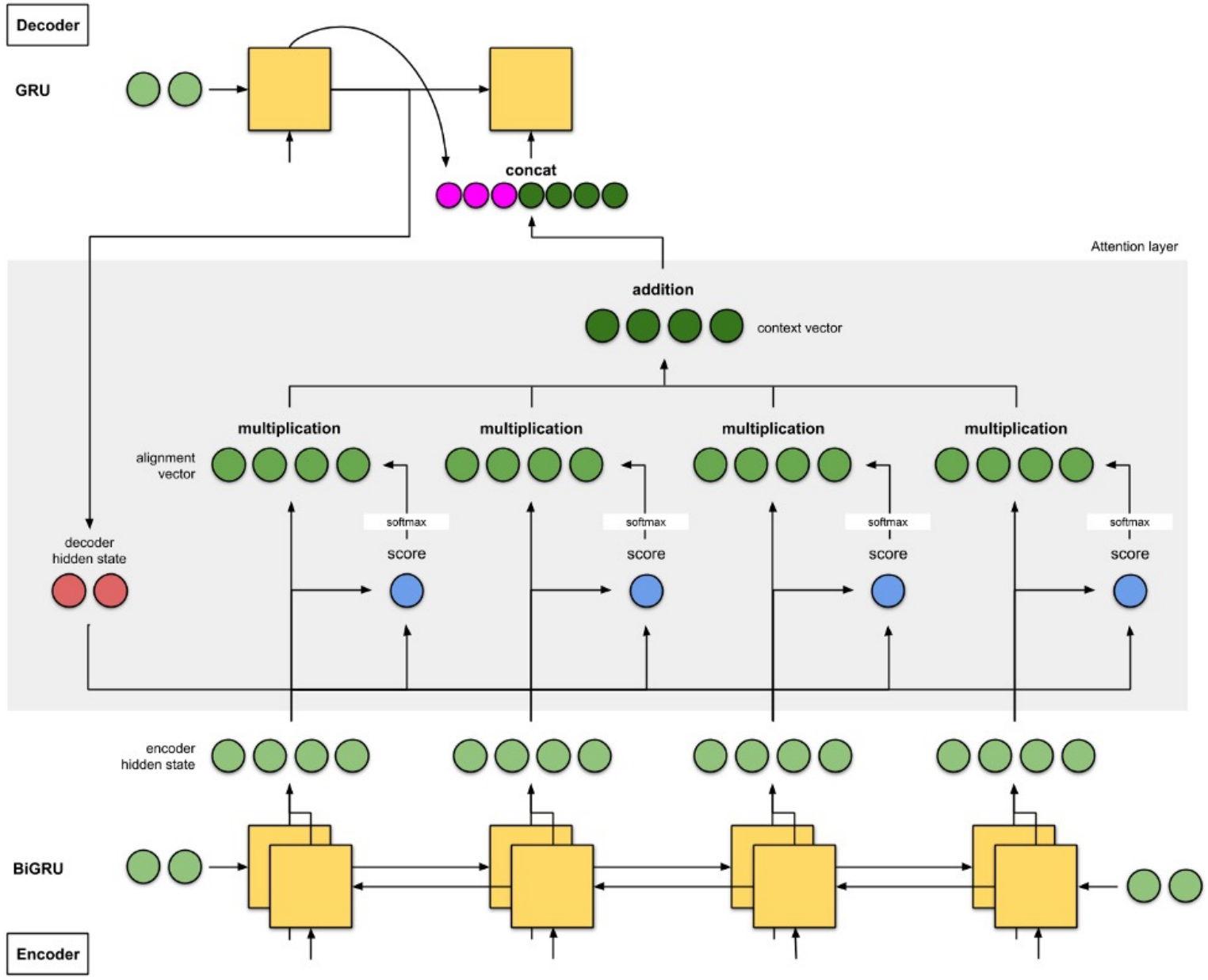
- Two RNNs where input is fed forward and backward respectively and then the hidden states (typically) are concatenated into a hidden state



Entire sequence must be observed to make a prediction (e.g., unsuitable for text prediction)

https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

Bahdanau's Neural Machine Translation



Bi-directional RNN

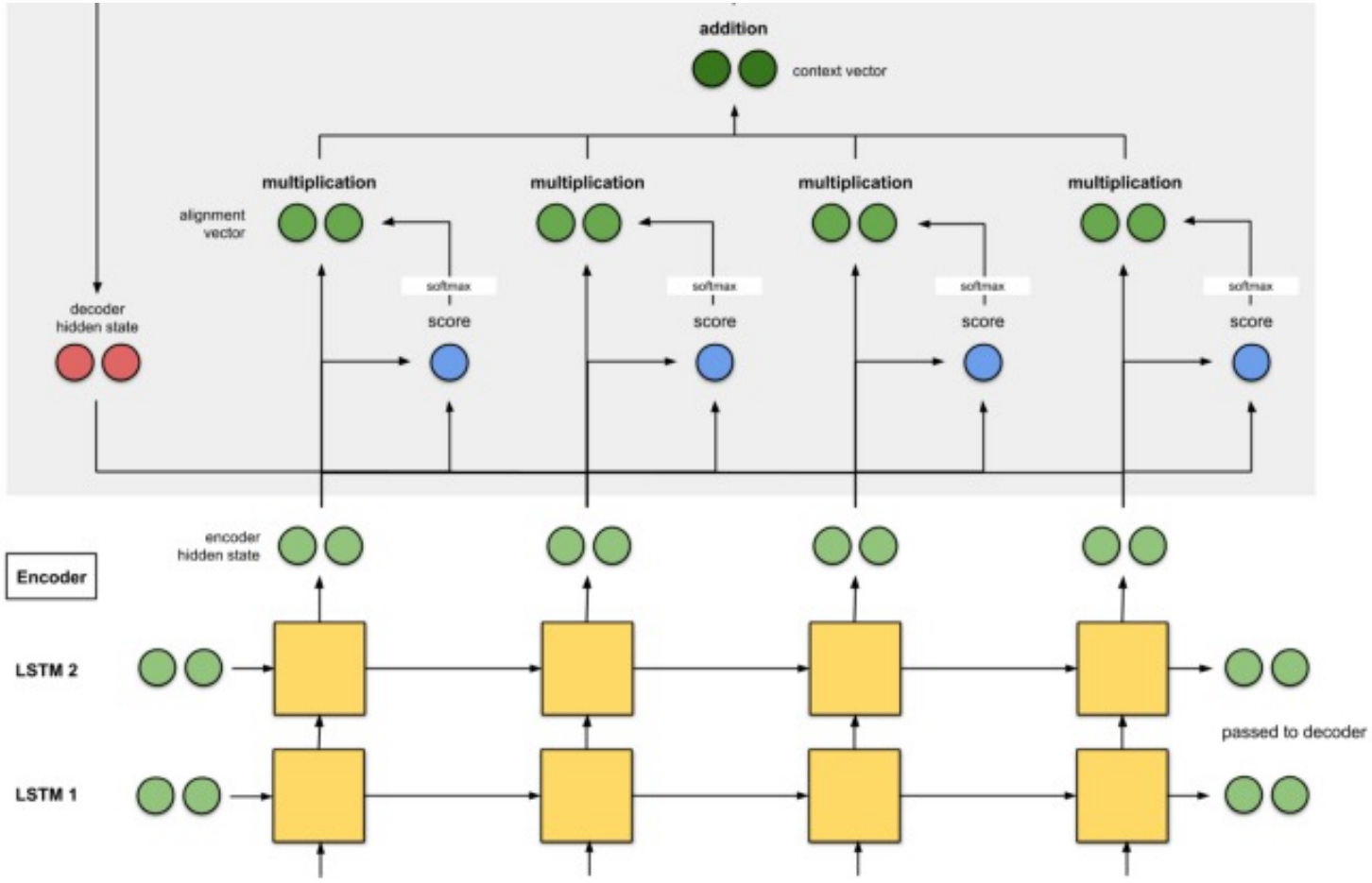
Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Popular Choices for Encoding Input

- Bi-directional RNN
- Stacked RNNs

Luong's Neural Machine Translation



2 layers

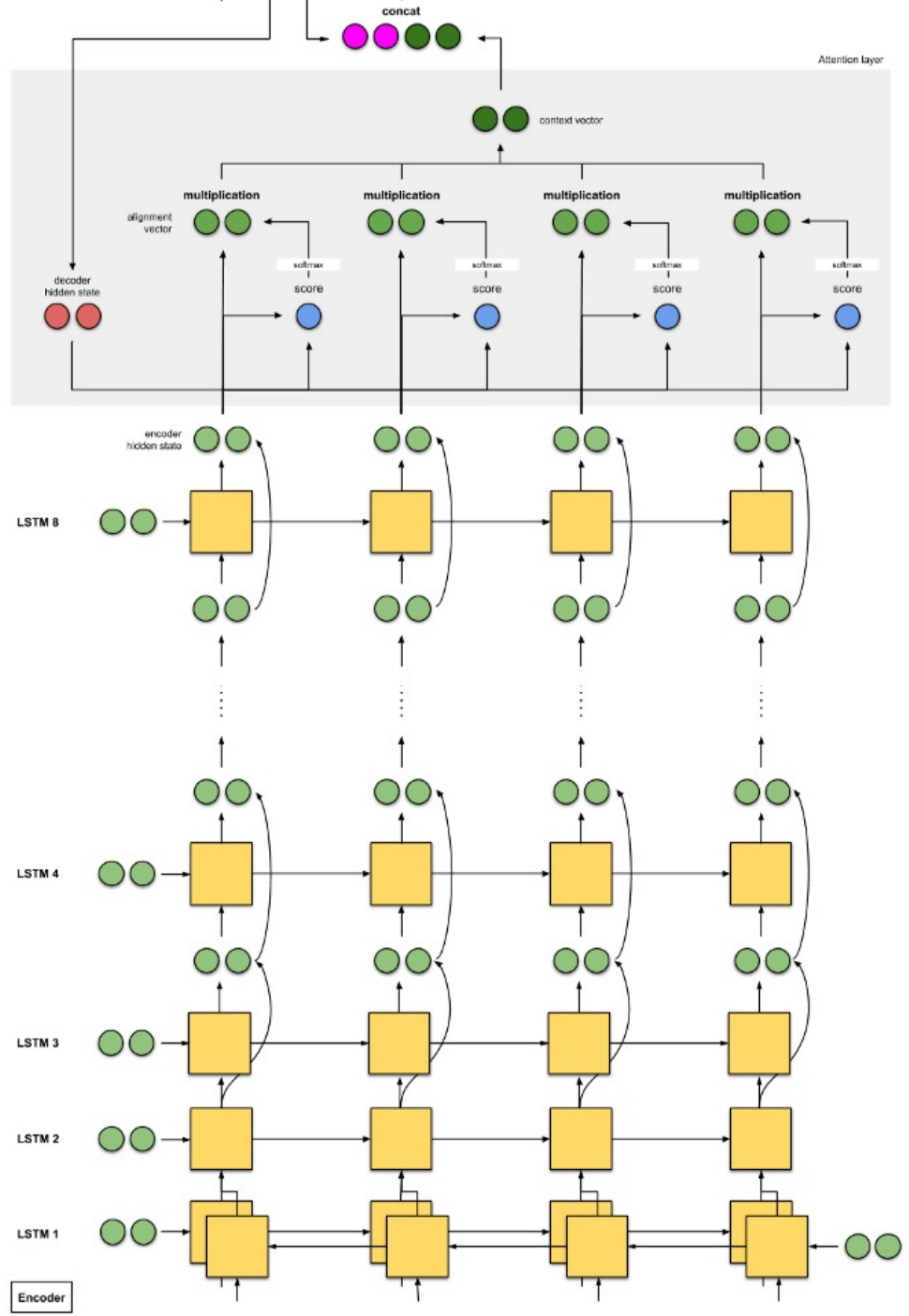
Luong et al. Effective Approaches to Attention-based Neural Machine Translation. EMNLP 2015

https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3#df28

Popular Choices for Encoding Input

- Bi-directional RNN
- Stacked RNNs

Google's Neural Machine Translation



8 layers with 1rst layer bi-directional

Wu et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv 2016.

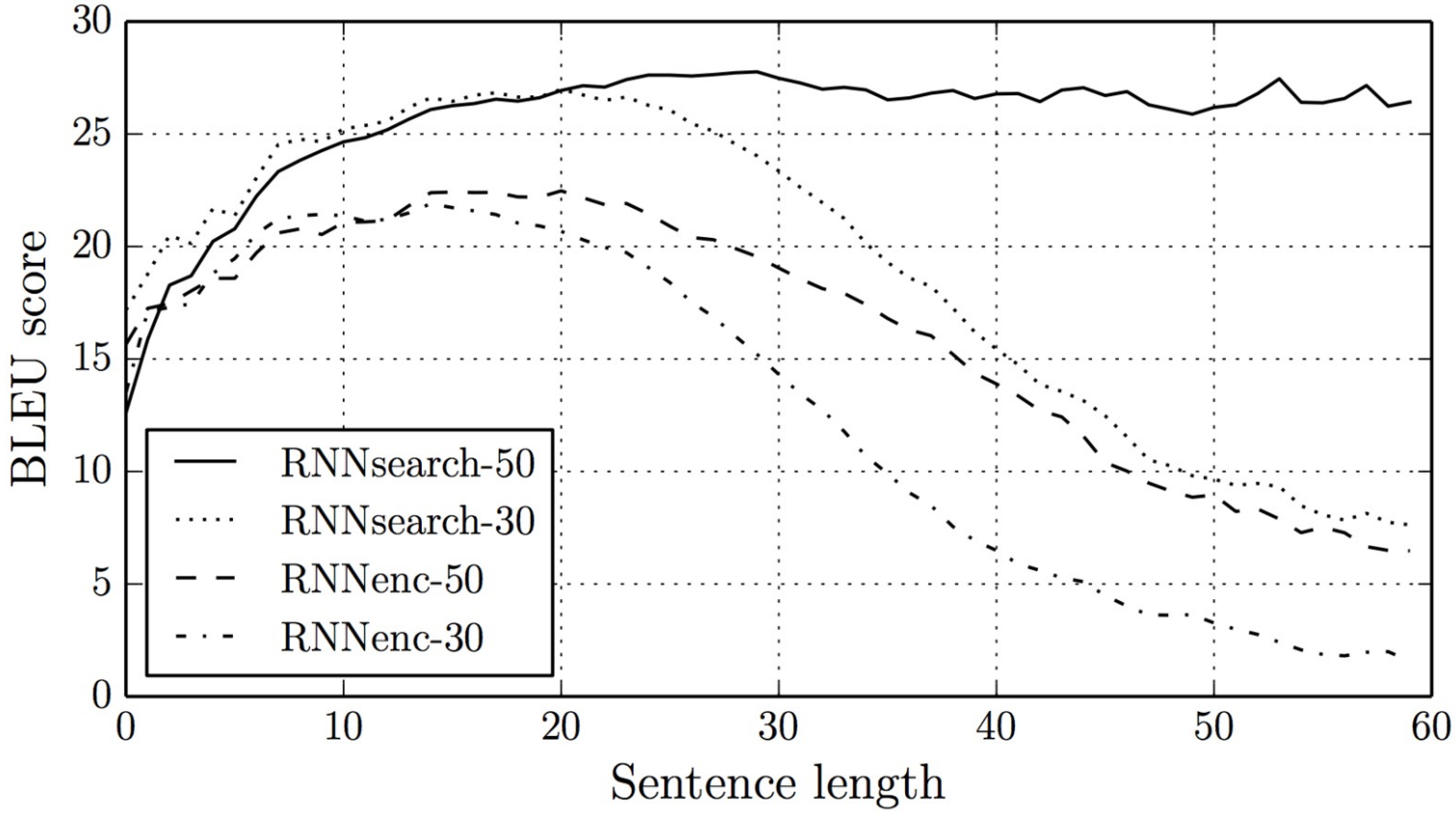
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3#df28

Popular Choices for Encoding Input

- Bi-directional RNN
- Stacked RNNs

Analysis of Attention Models

(larger scores are better)

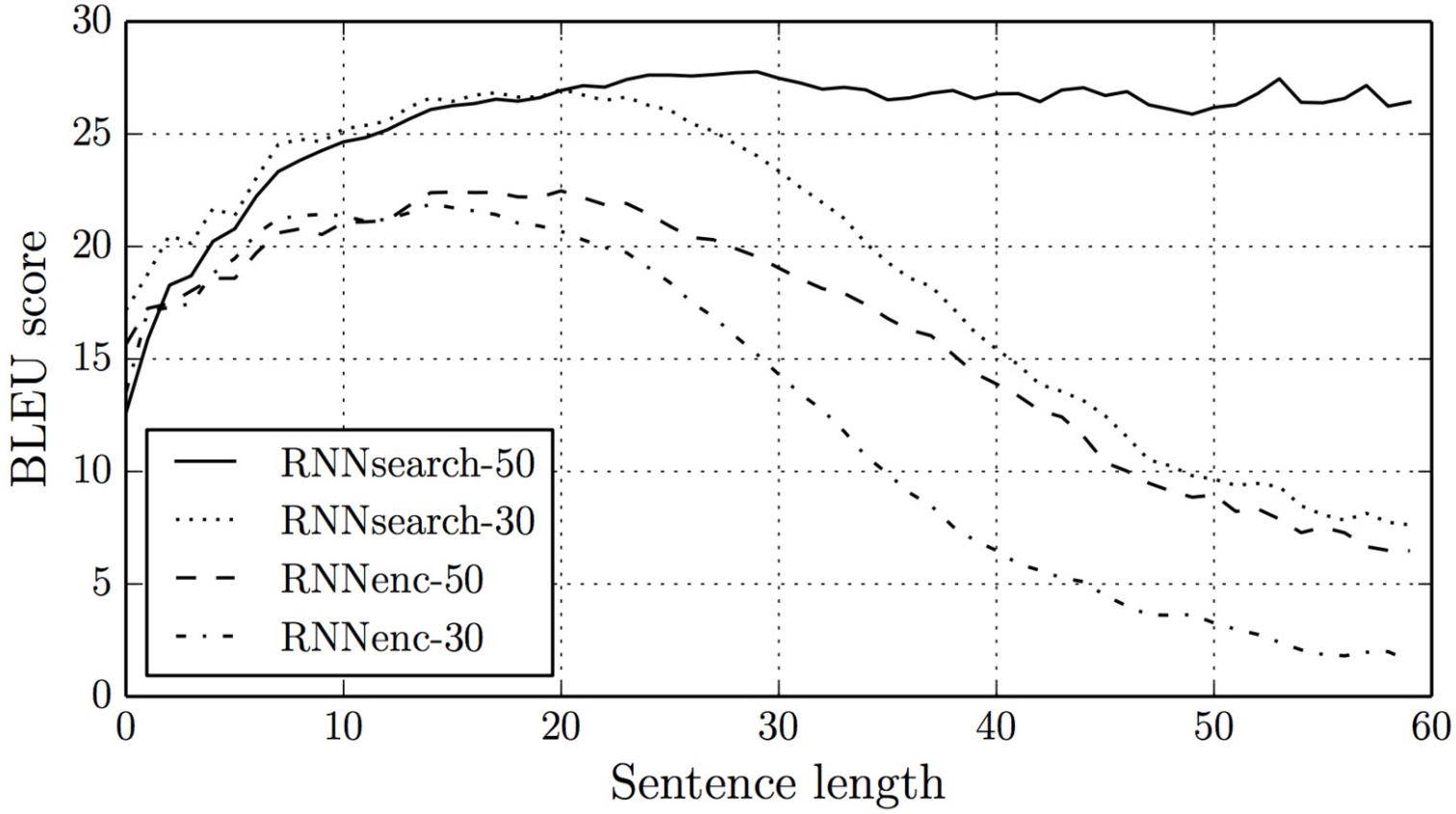


What performance trend is observed as the number of words in the input sentence grows?

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Analysis of Attention Models

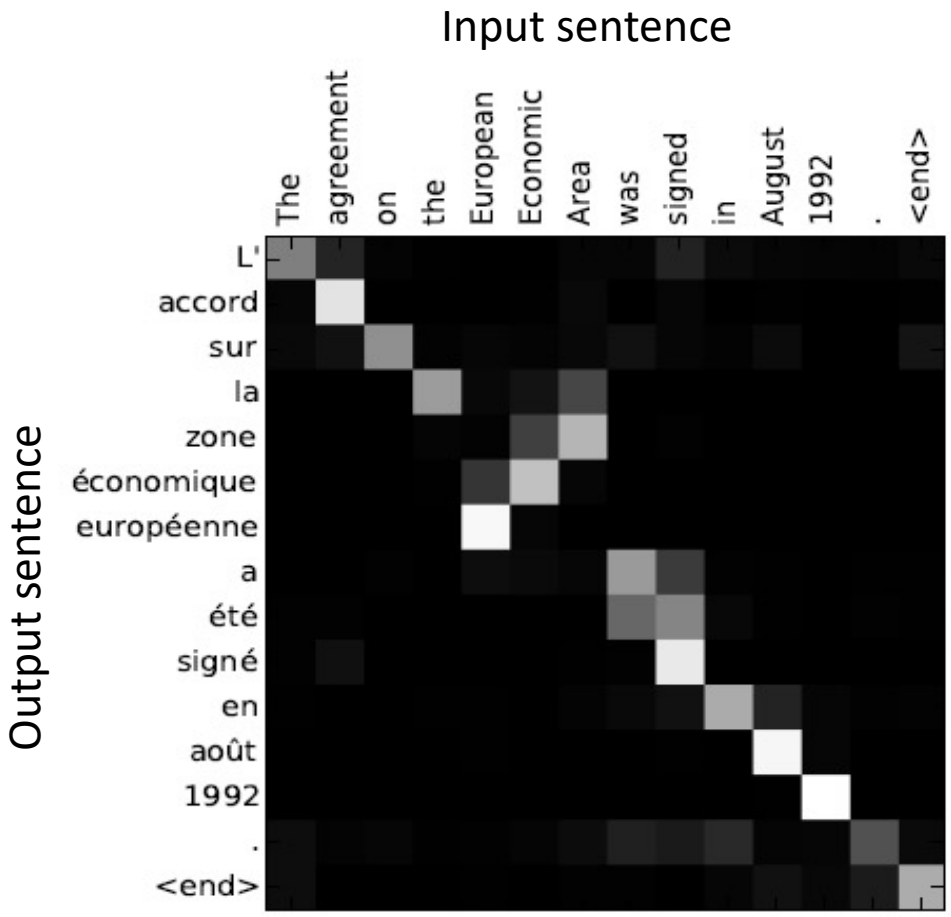
(larger scores are better)

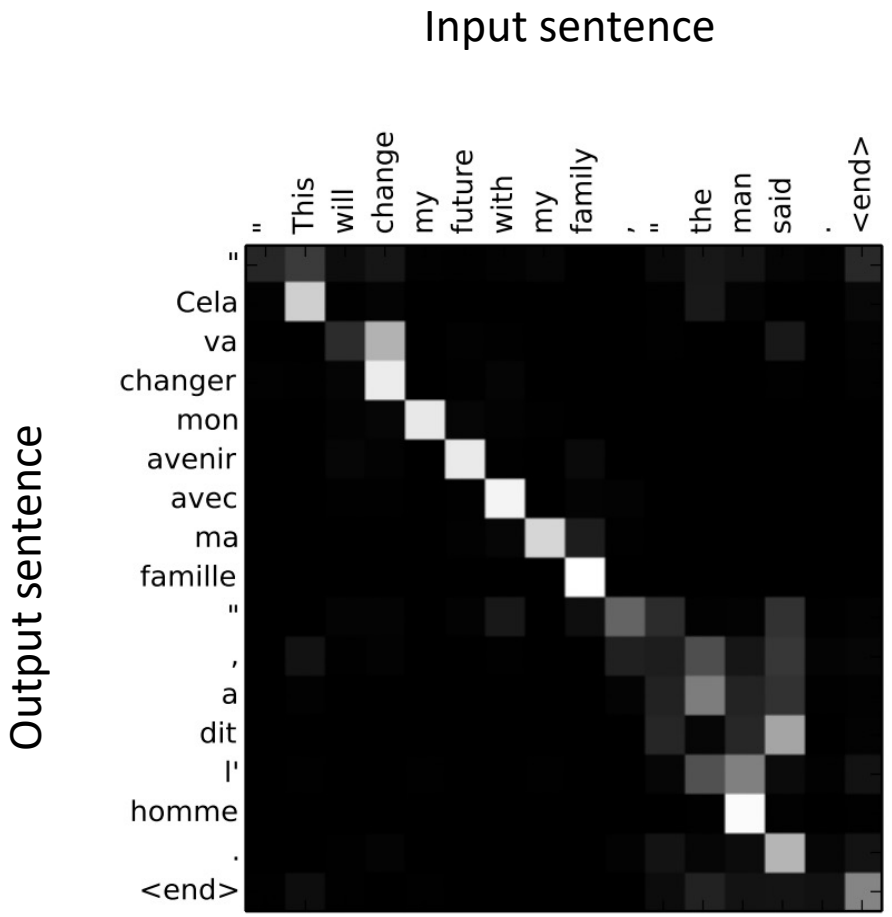


Performance no longer drops for longer sentences!

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Visualizing Attention

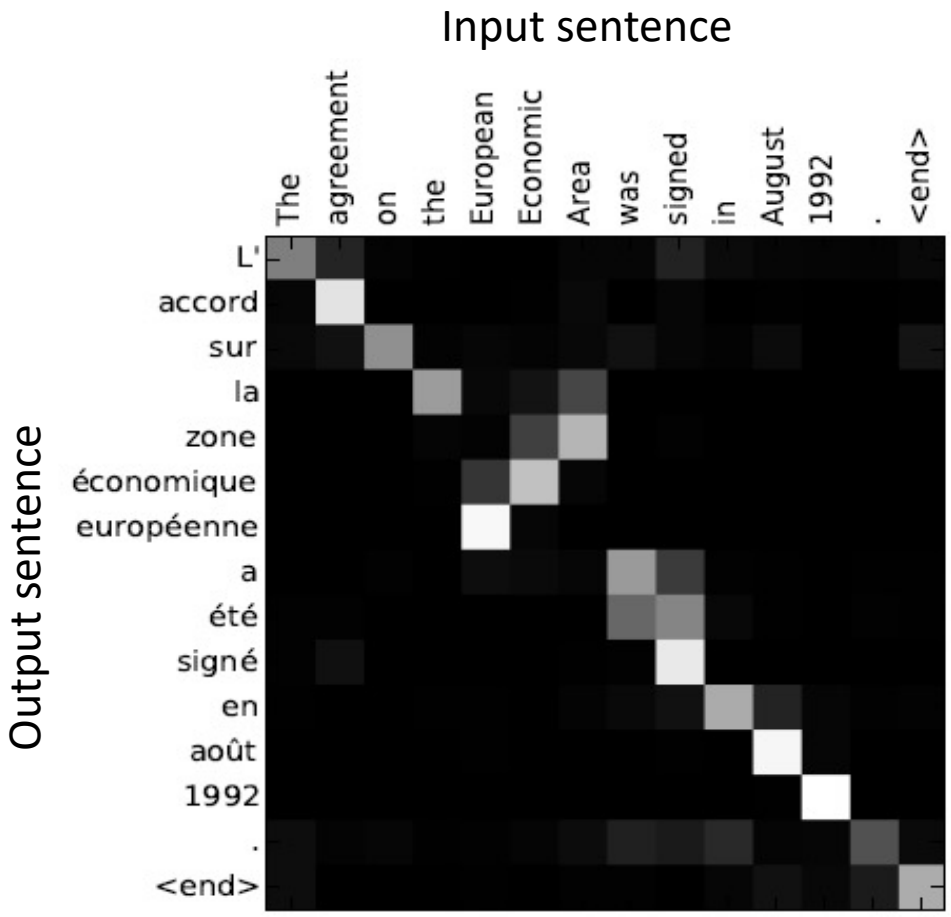


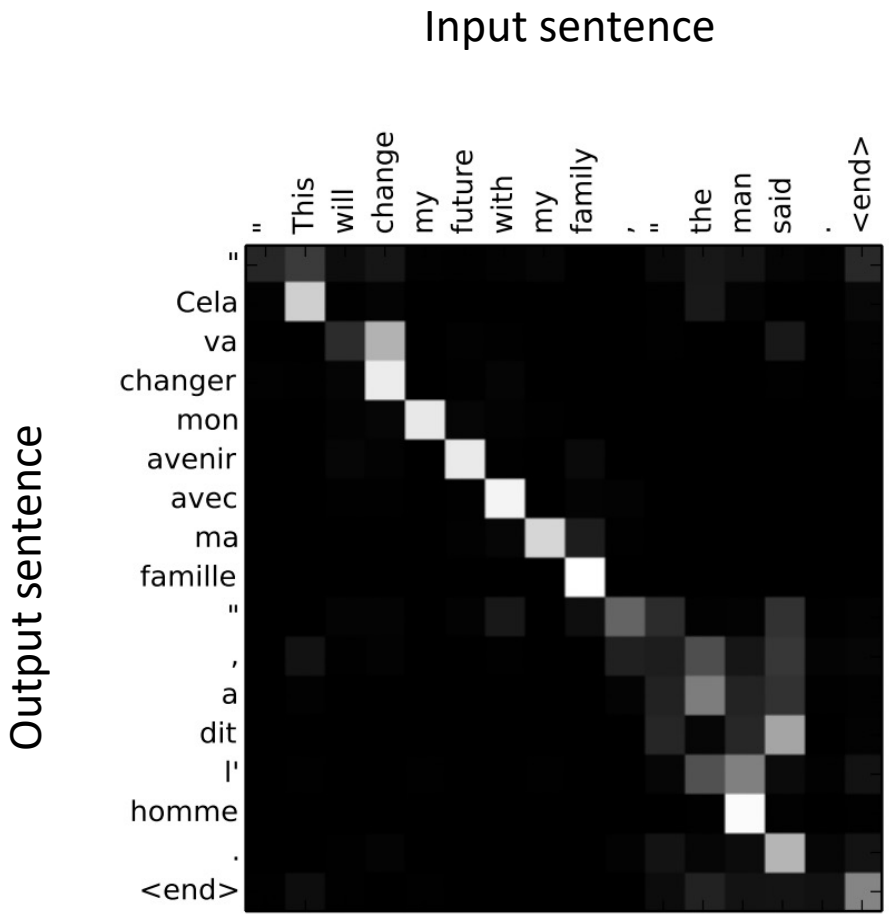


Values are 0 to 1, with whiter pixels indicating larger attention weights

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Visualizing Attention

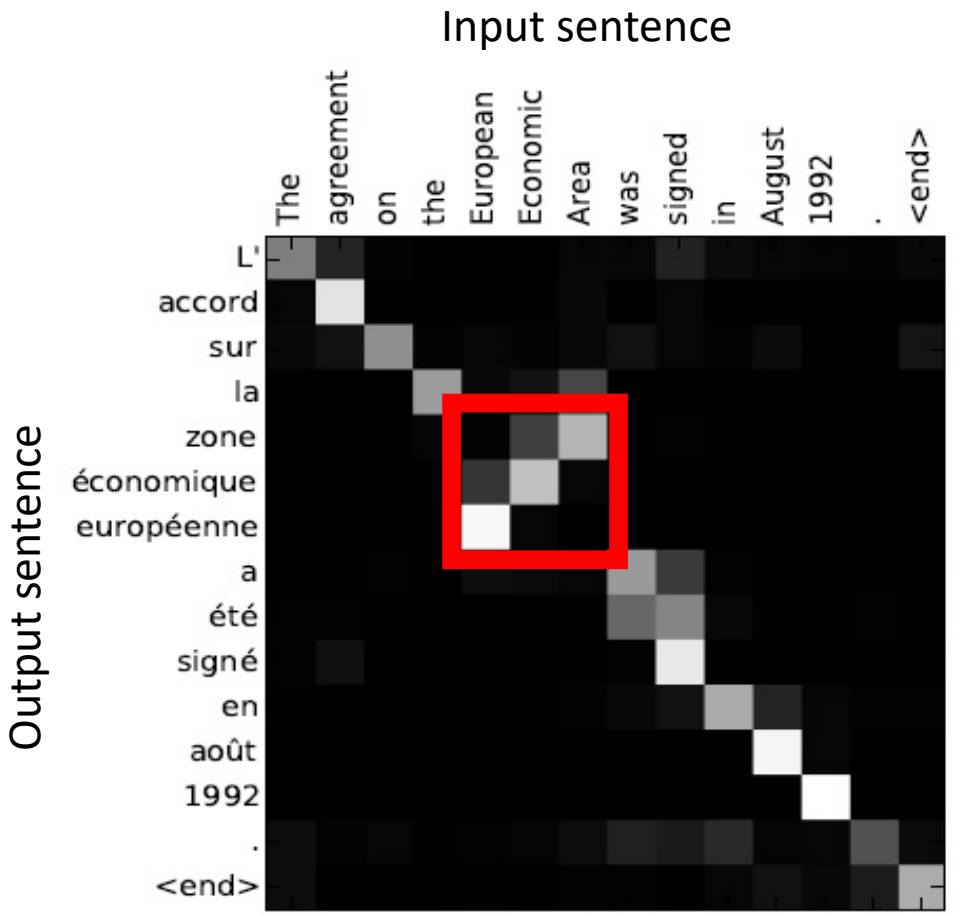


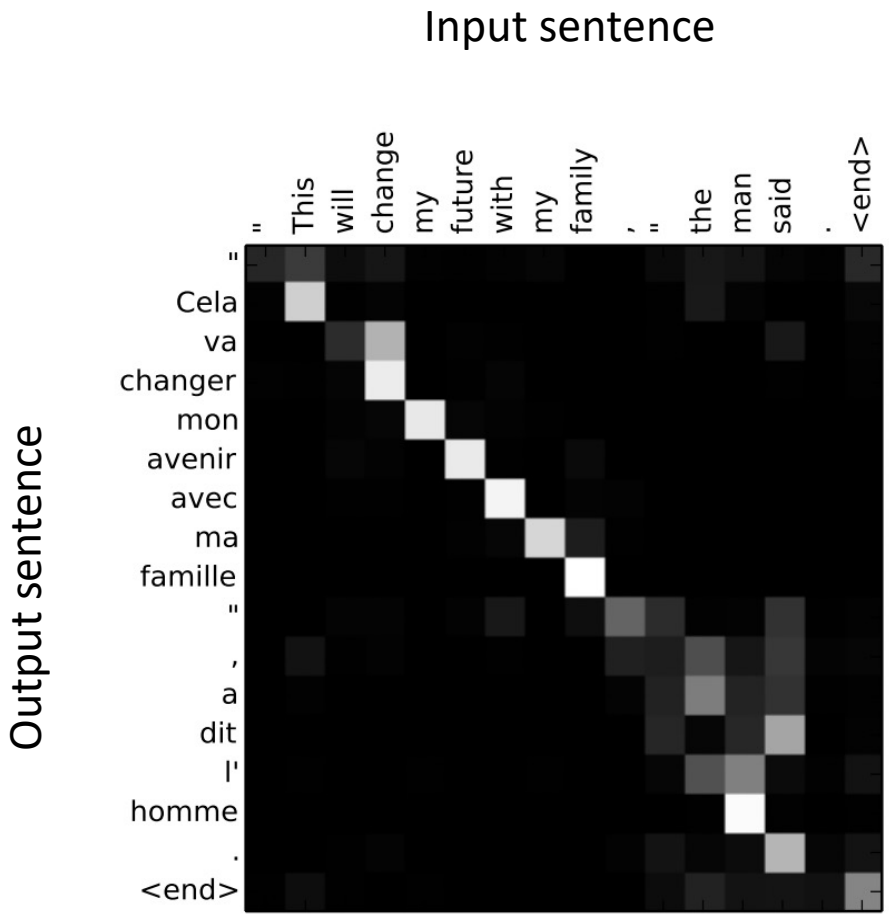


What insights can we glean from these examples?

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Visualizing Attention

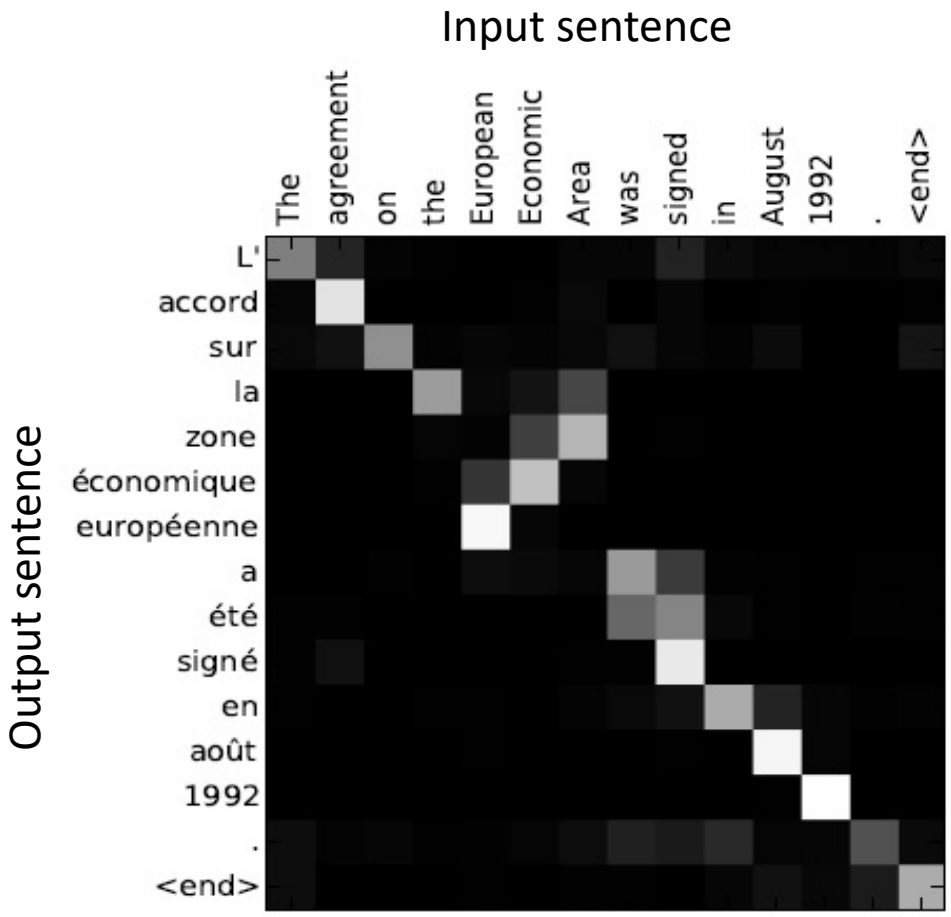


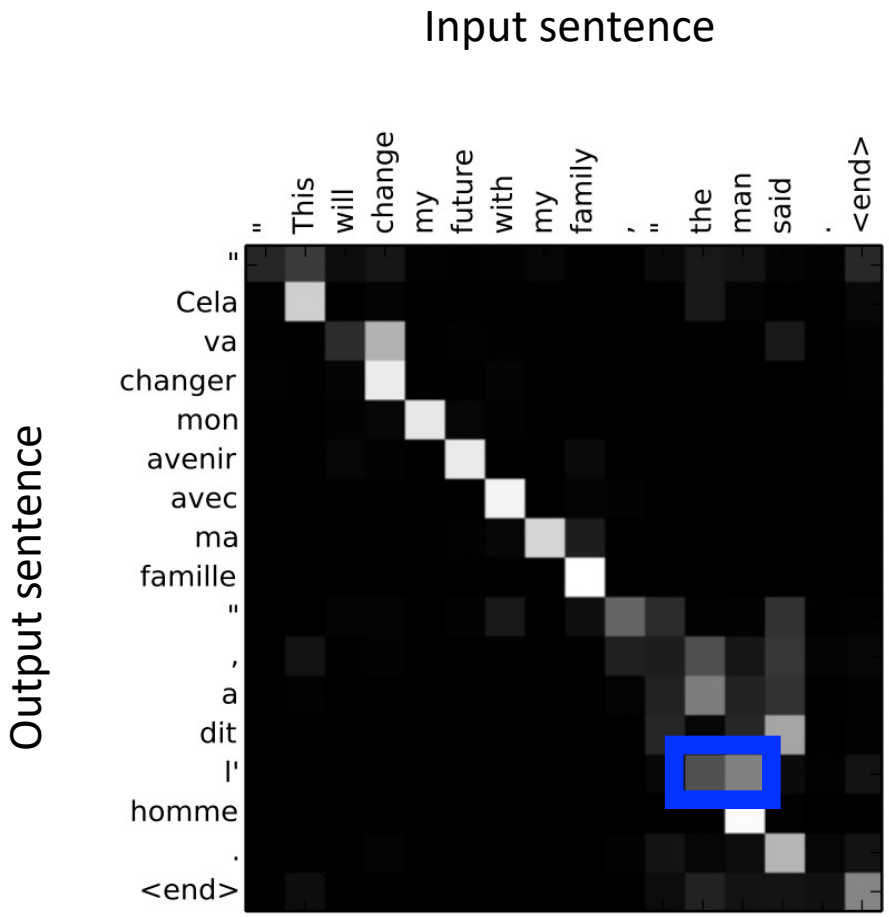


While a linear alignment between input and output sentences is common, there are exceptions (e.g., order of adjectives and nouns can differ)

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Visualizing Attention

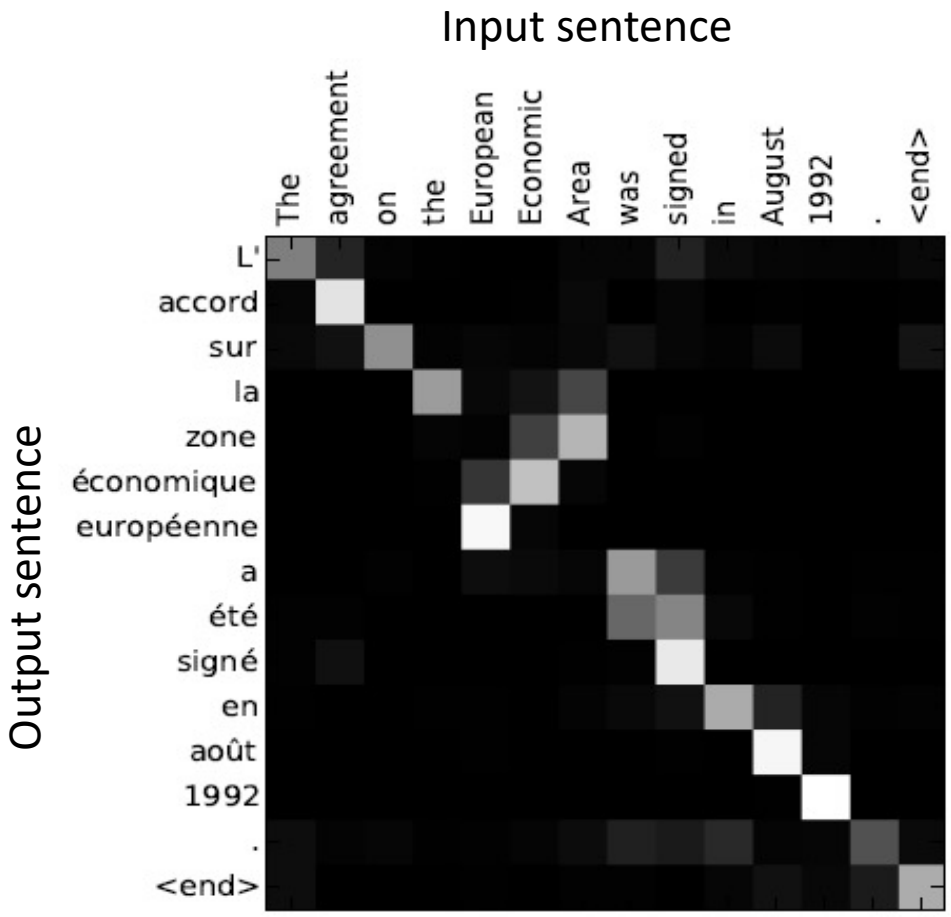


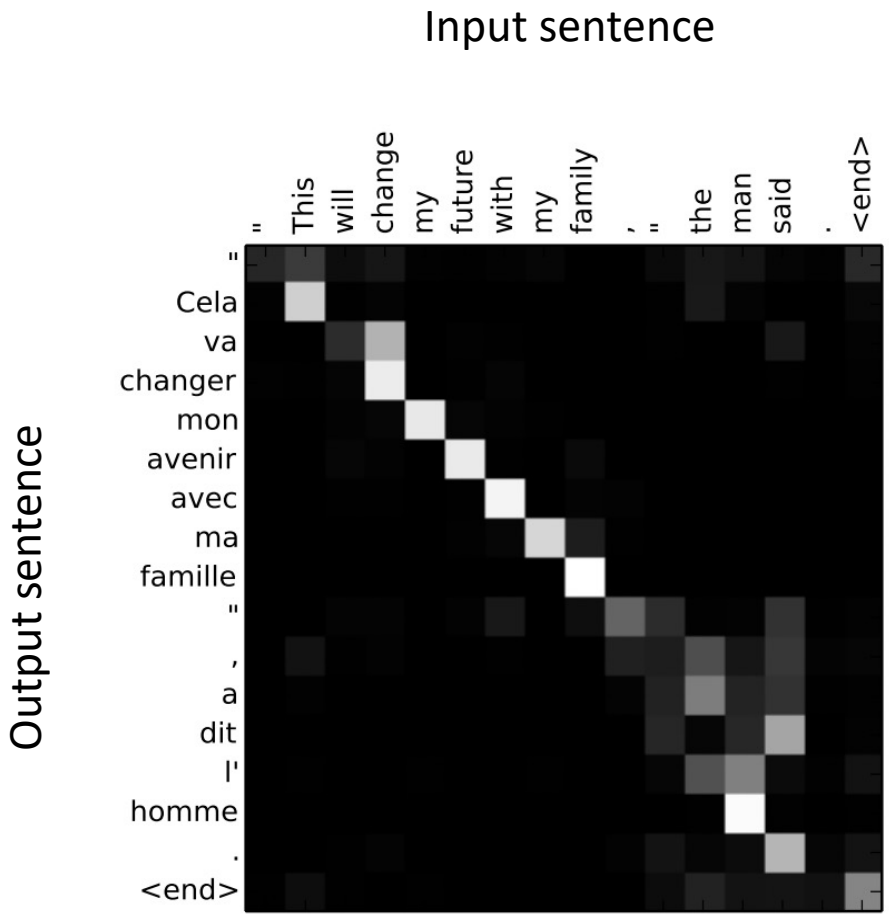


Output words are often informed by more than one input word;
e.g., “man” indicates translation of “the” to l’ instead of le, la, or les

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Visualizing Attention





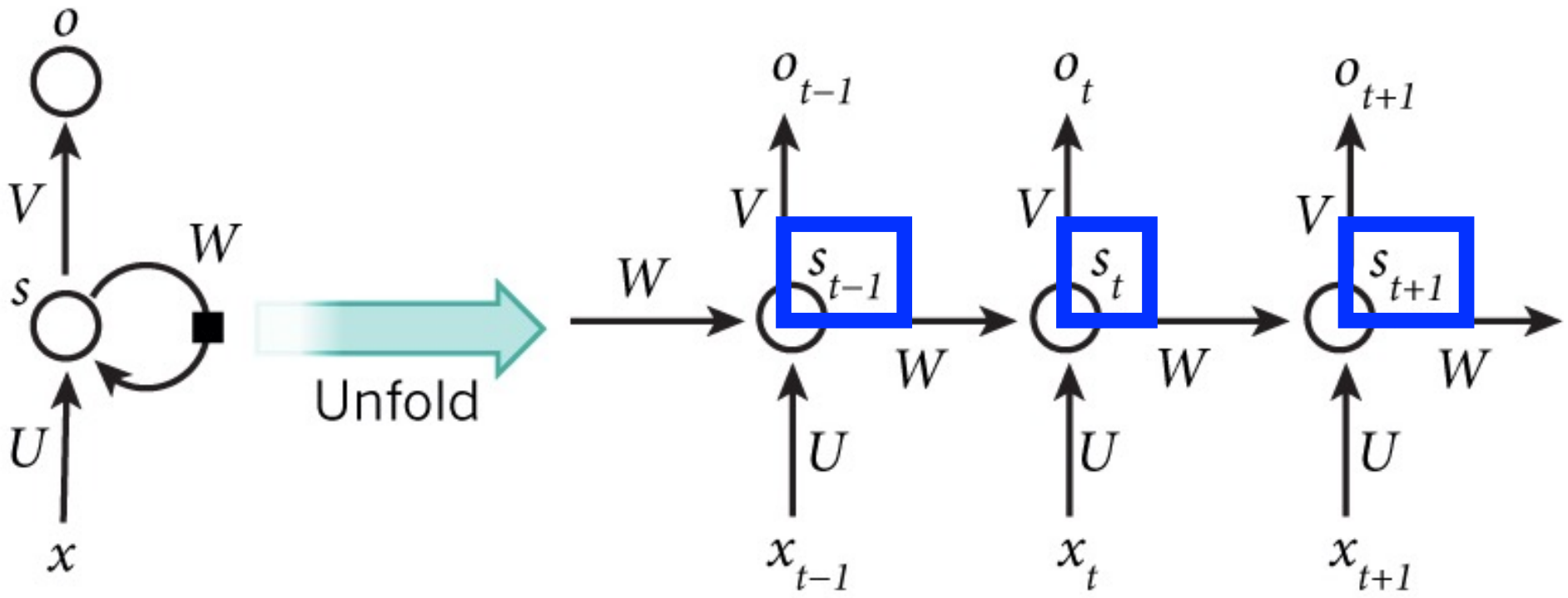
It naturally handles different input and output lengths
(e.g., 1 extra output word for both examples)

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Today's Topics

- Transformer overview
- Self-attention
- Multi-head attention
- Common transformer ingredients
- Pioneering transformer: machine translation

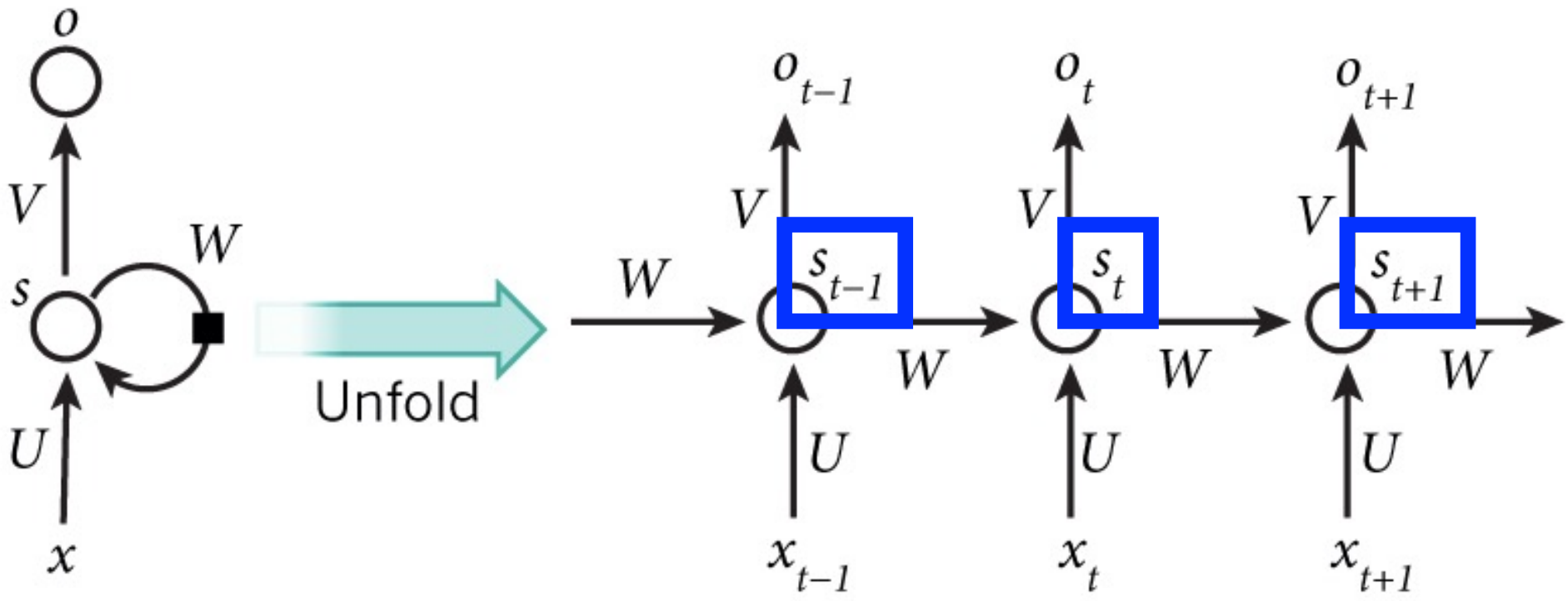
Goal: Model Sequential Data (Recall RNN)



Each hidden state is a function of the previous hidden state

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

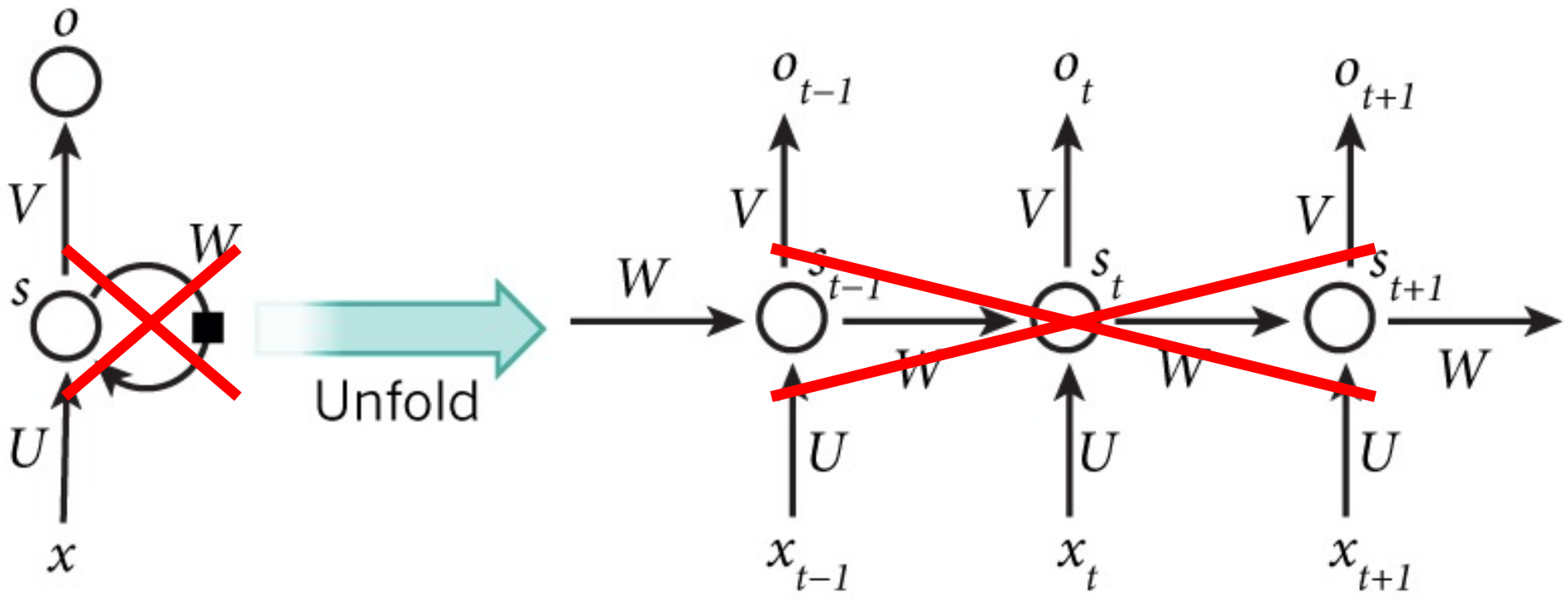
Problem: RNNs Use Sequential Computation



Seemingly hard for RNNs to carry information through hidden states across many time steps and train/testing is slow

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Idea: Model Sequential Data Without Recurrence

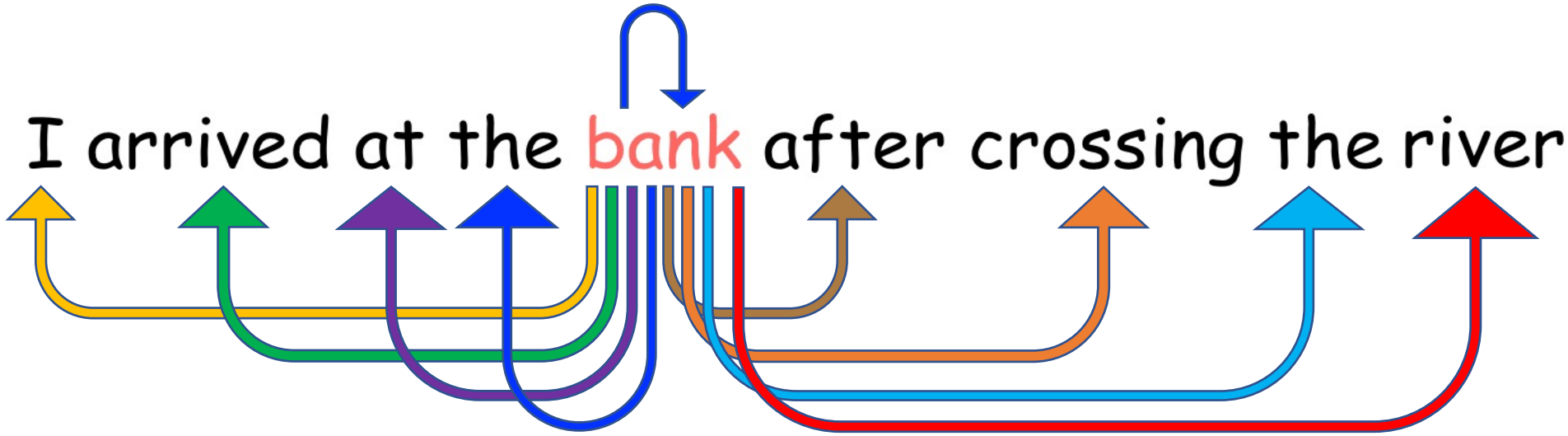


Replace sequential hidden states for capturing knowledge of other inputs with a new representation of each input that shows its relationship to all other inputs (i.e., self-attention)

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Transformer Key Idea: Self-Attention

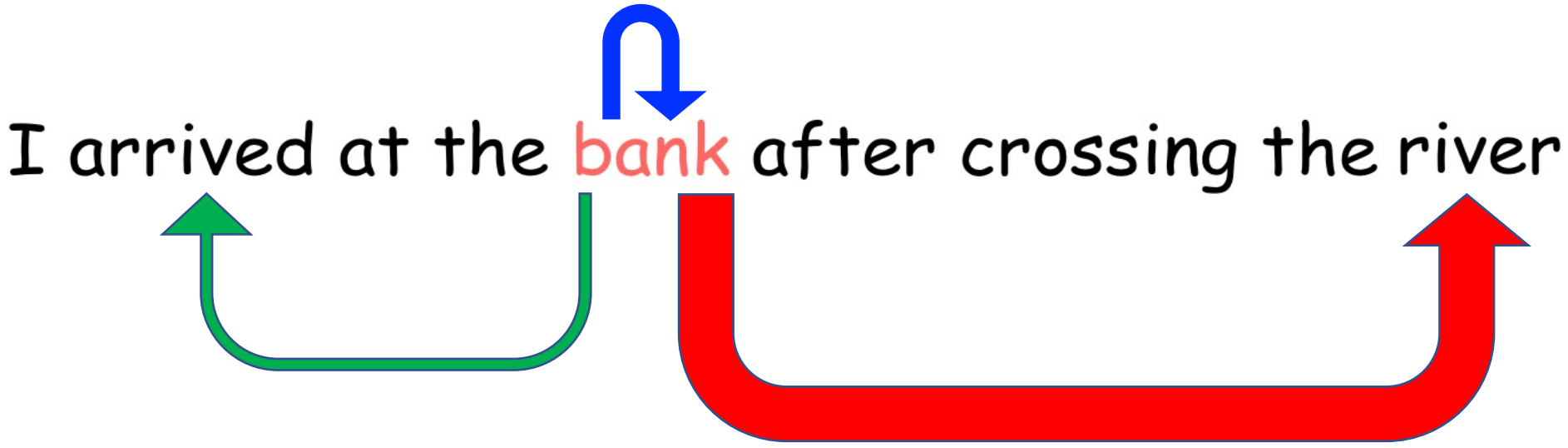
New representation of each token in a sequence showing its relationship to all tokens; e.g.,



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer Key Idea: Self-Attention

New representation of each token in a sequence showing its relationship to all tokens; e.g.,



Arrow thickness is indicative of attention weight

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer Key Idea: Self-Attention

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

A large attention score means the other word will strongly inform the new representation of the word

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer Intuition

What does bank mean in this sentence?

I arrived at the bank after crossing the …

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer Intuition

What does bank mean in this sentence?
- the new representation of the word disambiguates the meaning by identifying other relevant words (e.g., high attention score with “river”)

I arrived at the bank after crossing the river

vs

I arrived at the bank after crossing the street

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer vs RNN (Intuition)

I arrived at the bank after crossing the …

…street? …river?

What does **bank** mean in this sentence? Meaning depends on other input words

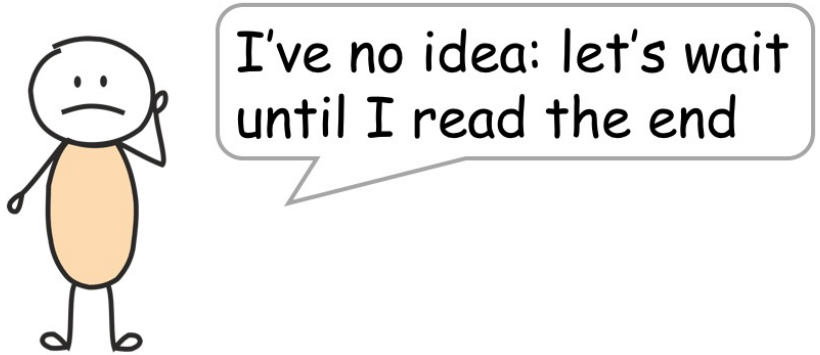
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer vs RNN (Intuition)

I arrived at the bank after crossing the …

…street? …river?

What does **bank** mean in this sentence? Meaning depends on other input words



RNNs

O(N) steps to process a sentence with length N

Transformer

Constant number of steps to process any sentence

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer: A Suggested Definition

"Any architecture designed to process a connected set of units—such as the tokens in a sequence or the pixels in an image—where the only interaction between units is through self-attention."

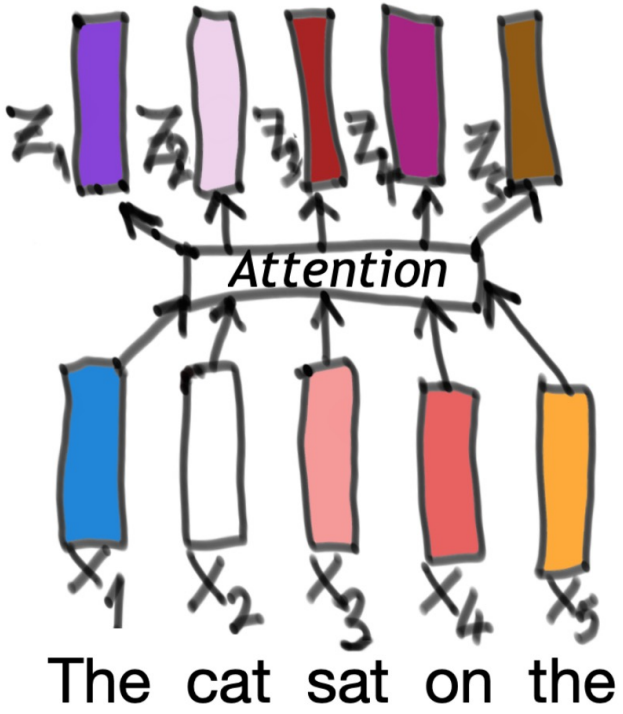
http://peterbloem.nl/blog/transformers

Today's Topics

- Transformer overview
- Self-attention
- Multi-head attention
- Common transformer ingredients
- Pioneering transformer: machine translation

Self-Attention: Outcome

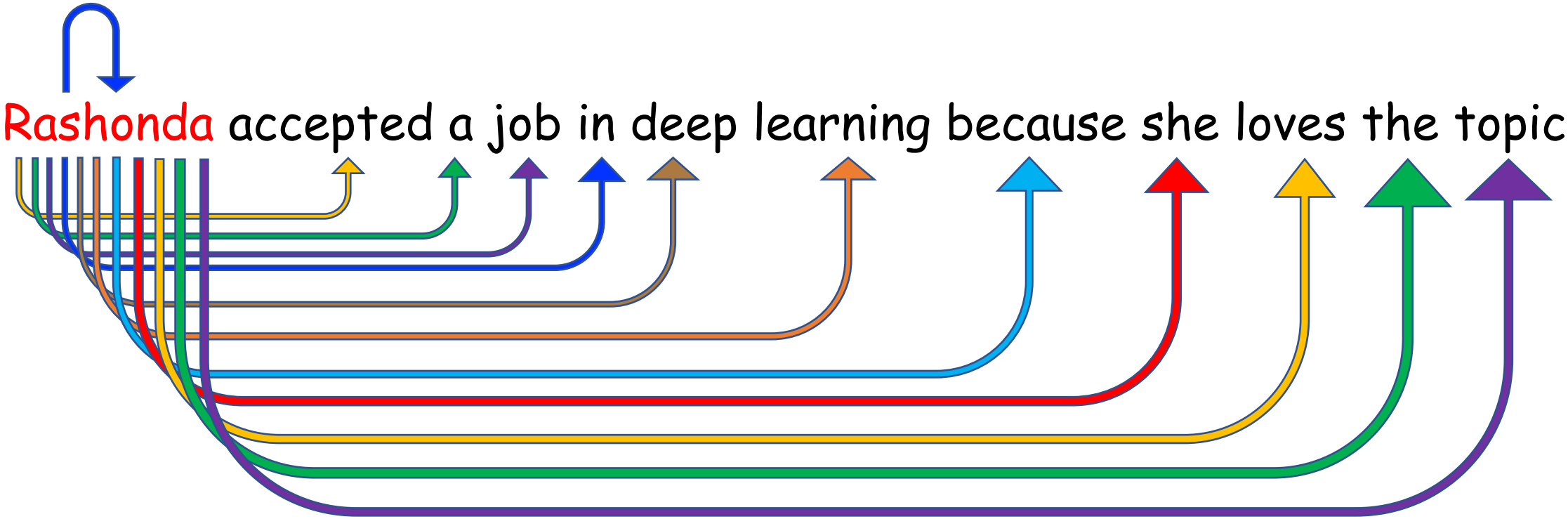
New representation of each token in a sequence showing its relationship to all tokens



https://towardsdatascience.com/self-attention-5b95ea164f61

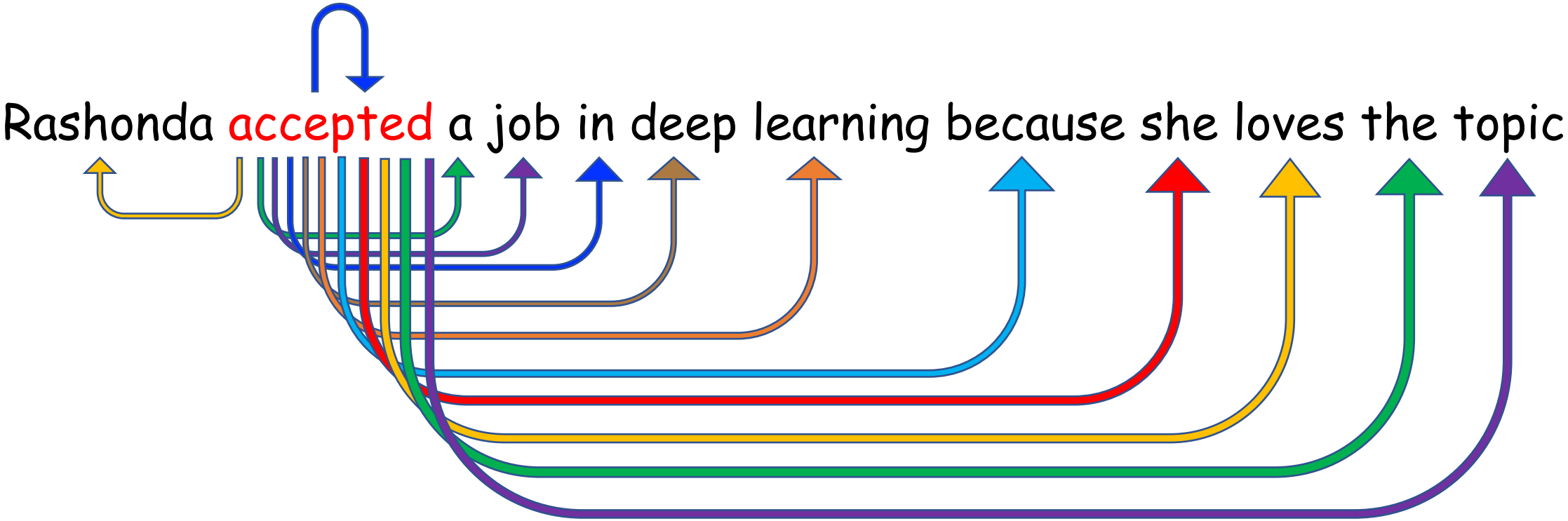
Self-Attention: Outcome

New representation of each token in a sequence showing its relationship to all tokens; e.g.,



Self-Attention: Outcome

New representation of each token in a sequence showing its relationship to all tokens; e.g.,



Self-Attention: Outcome

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

Rashonda accepted a job in deep learning because she loves the topic

And so on for remaining words…

Self-Attention: Disambiguates Word Meanings

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

Rashonda accepted a job in deep learning because she loves the topic

A better representation of “she” would
encode information about “Rashonda”

Self-Attention: Disambiguates Word Meanings

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

I arrived at the bank across the river

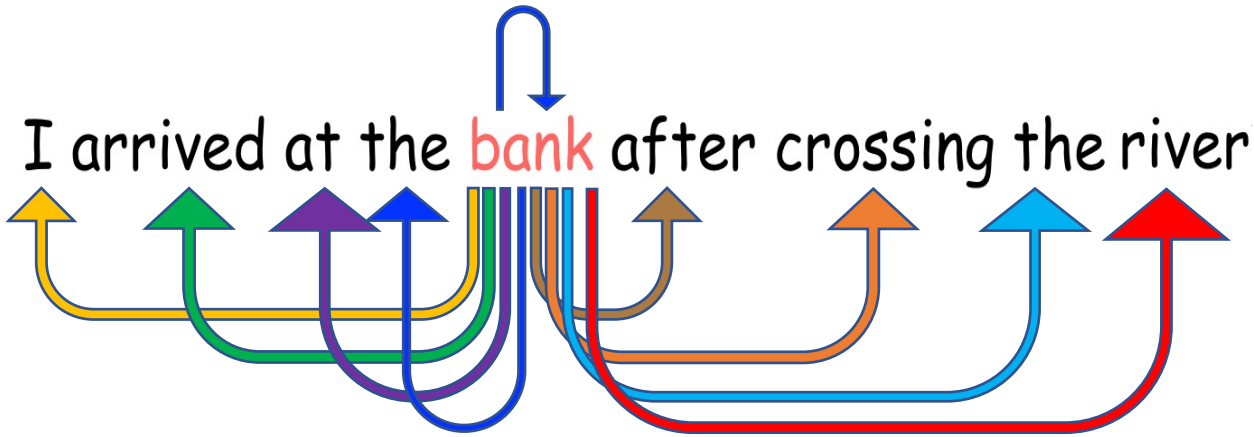
A better representation of “bank” would encode information about “river”

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Self-Attention vs General Attention

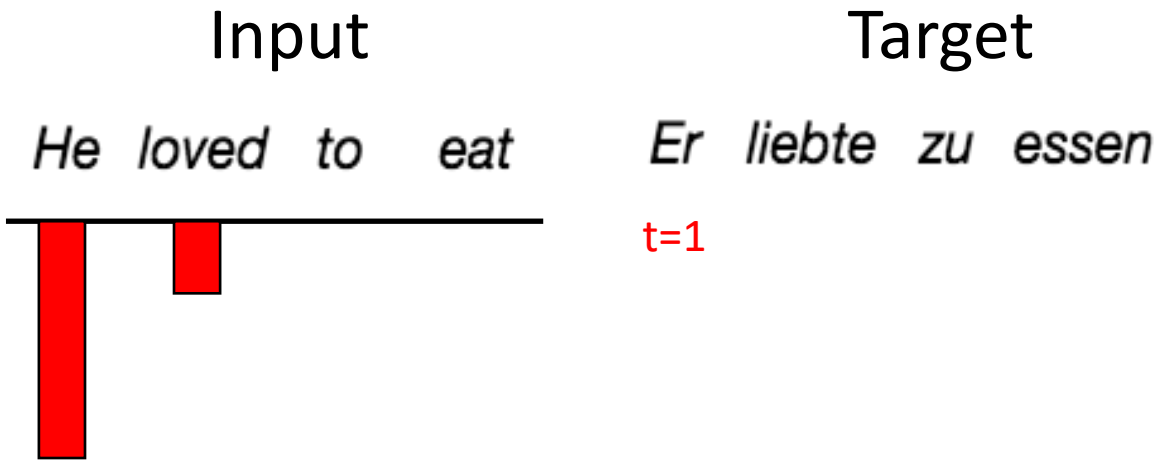
Self-attention

Relates tokens from the same source

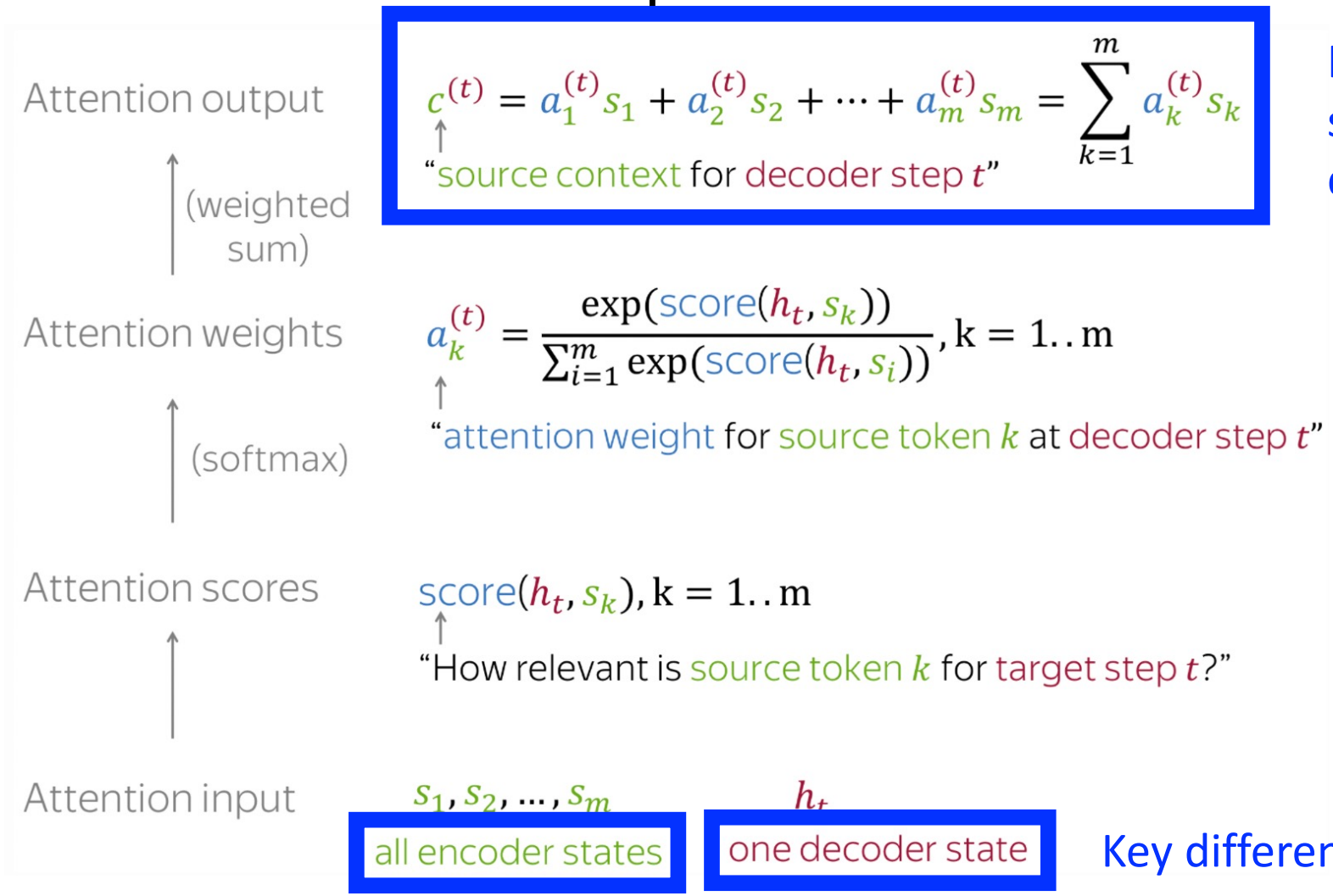


General attention

Relates tokens from different sources



Computing Self-Attention: Similar Approach to How We Compute General Attention



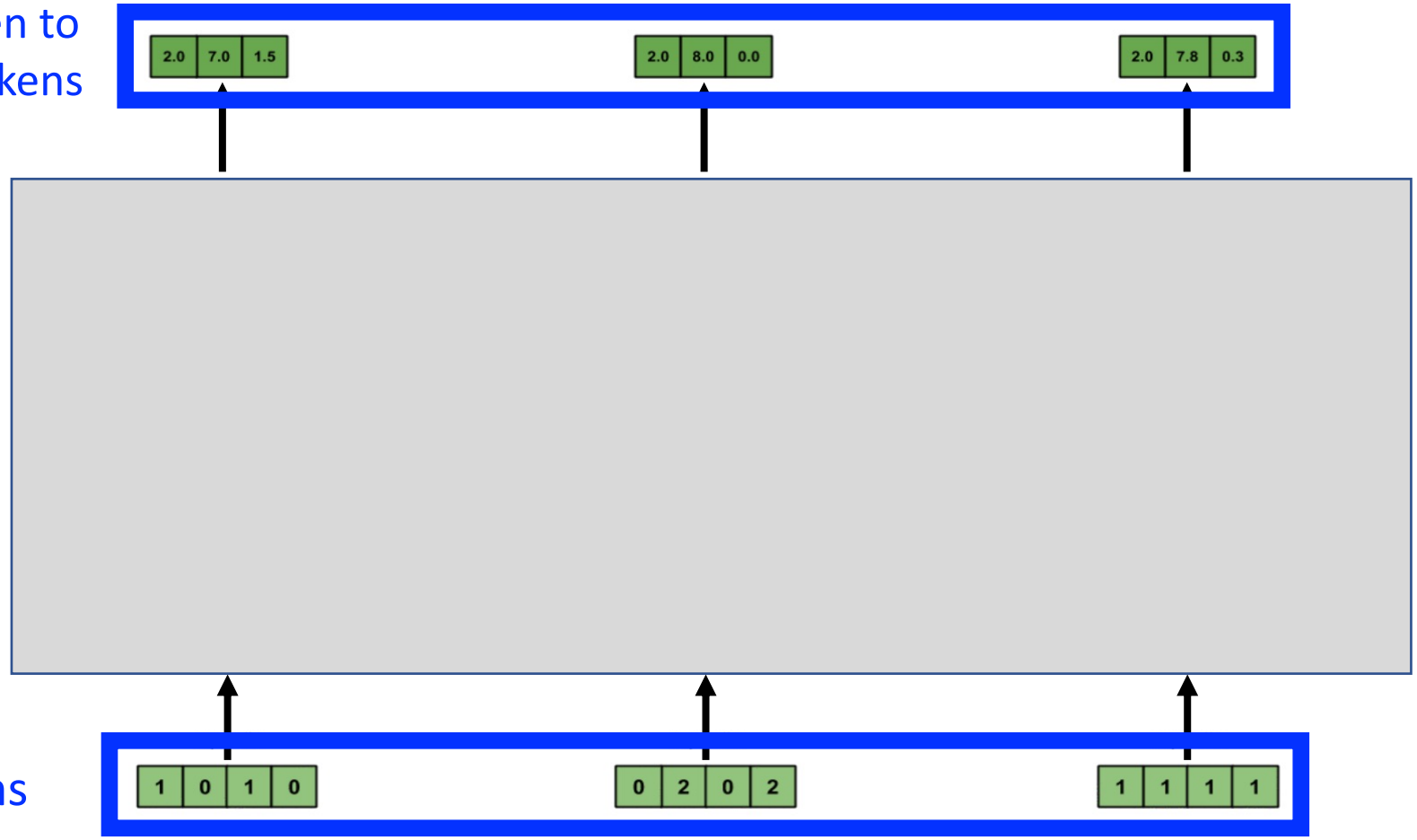
Key difference 2: attention score multiplied with a value derived from the input

Key difference 1: input for self-attention

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Computing Self-Attention: Example

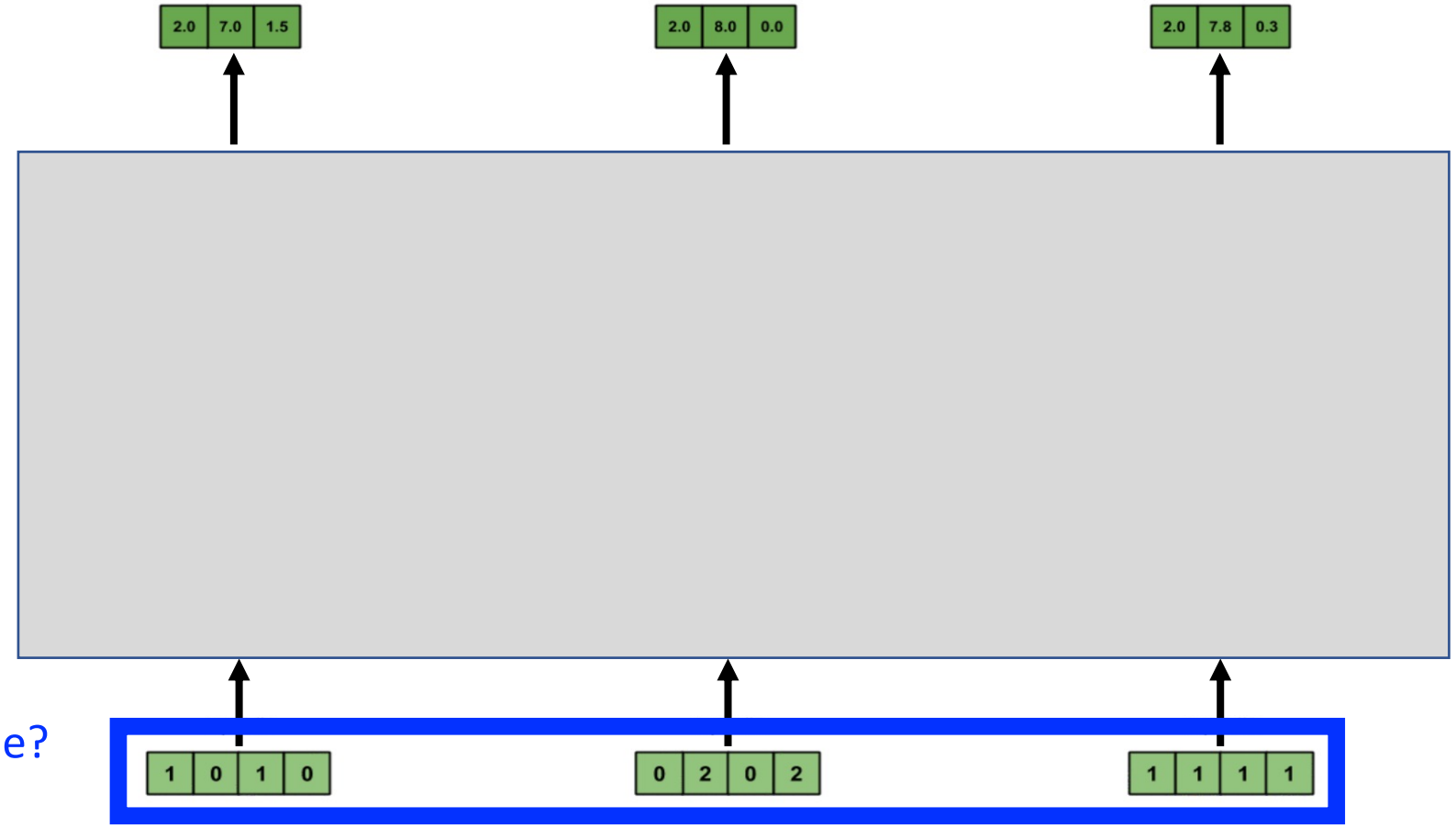
New representation of each input token to reflect each one's relationship to all tokens



Input tokens

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

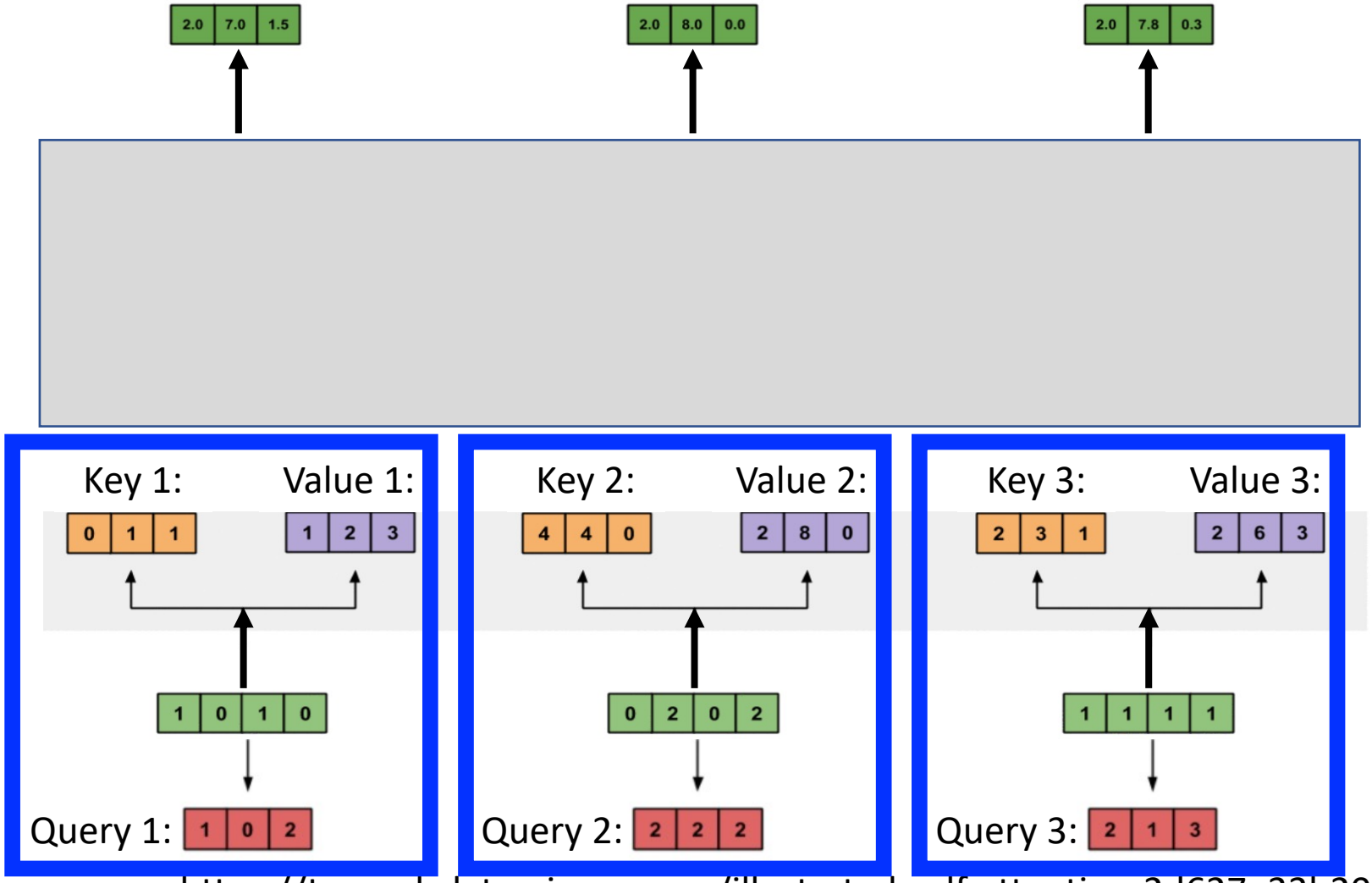


- How many inputs are in this example?
- What is each one's dimensionality?

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Three vectors are derived for each input by multiplying with three weight matrices (learned during training): query, key, and value

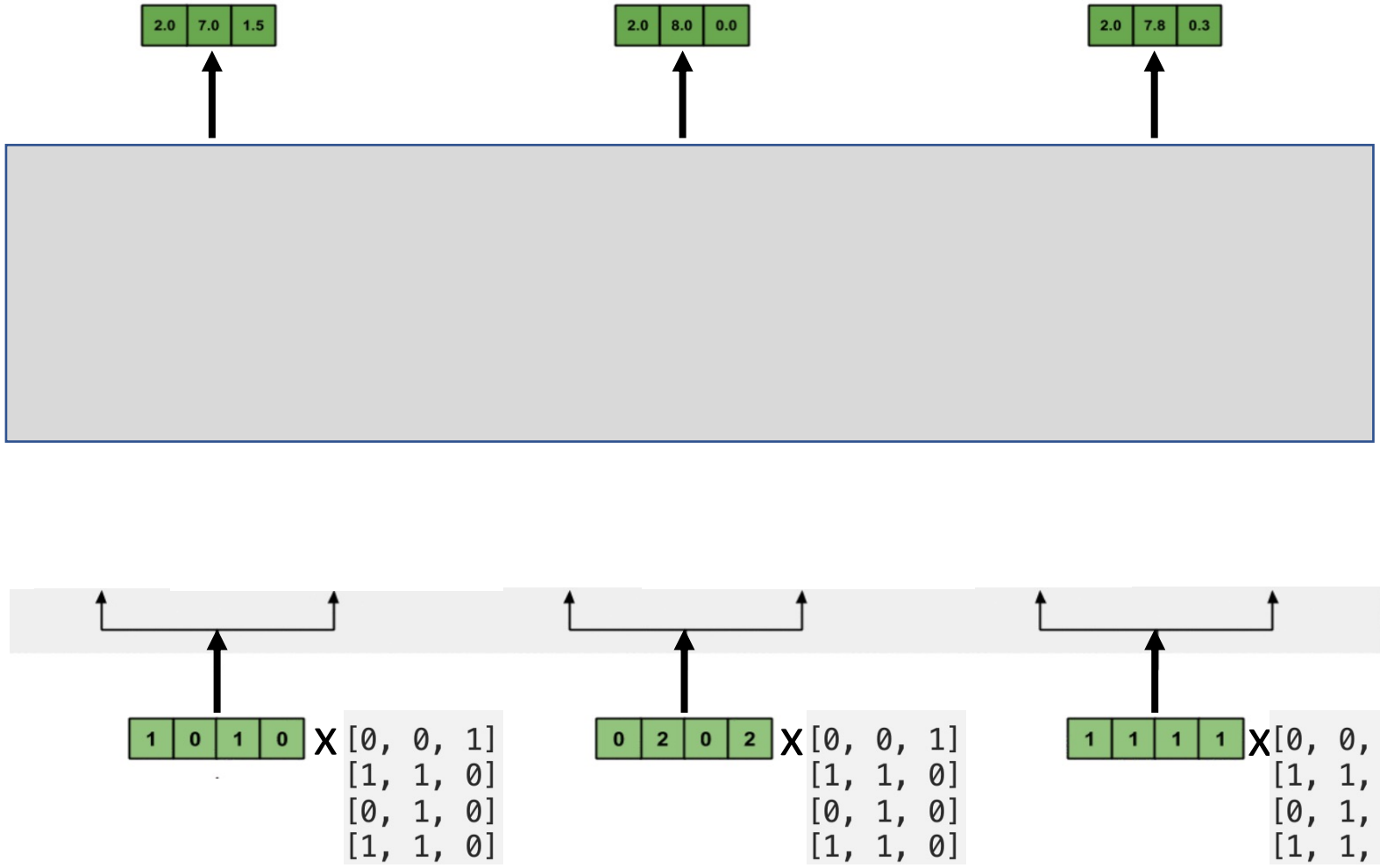


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

e.g., key weights

```
[0, 0, 1]
[1, 1, 0]
[0, 1, 0]
[1, 1, 0]
```



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

e.g., value weights

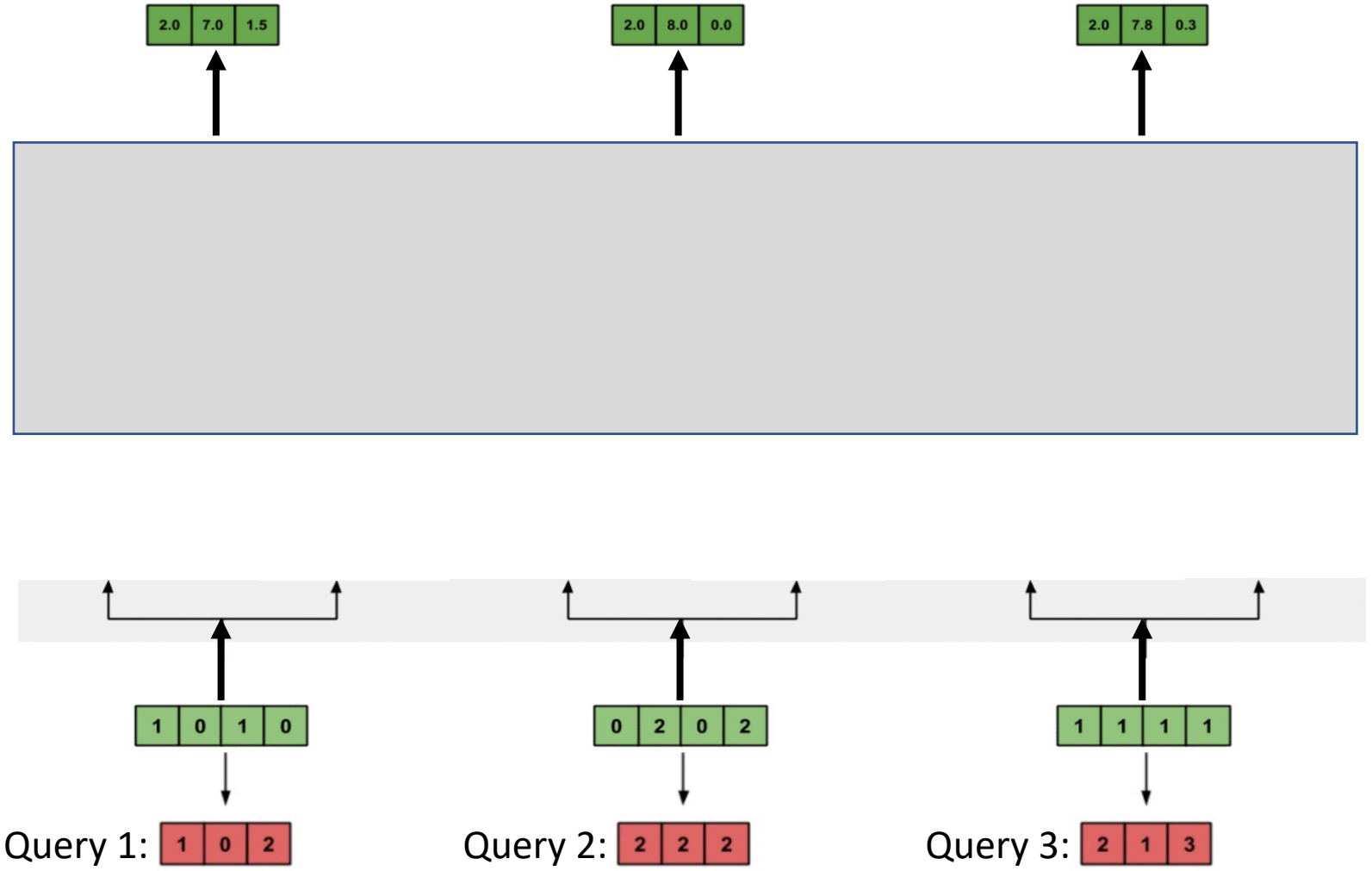
```
[0, 2, 0]
[0, 3, 0]
[1, 0, 3]
[1, 1, 0]
```

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

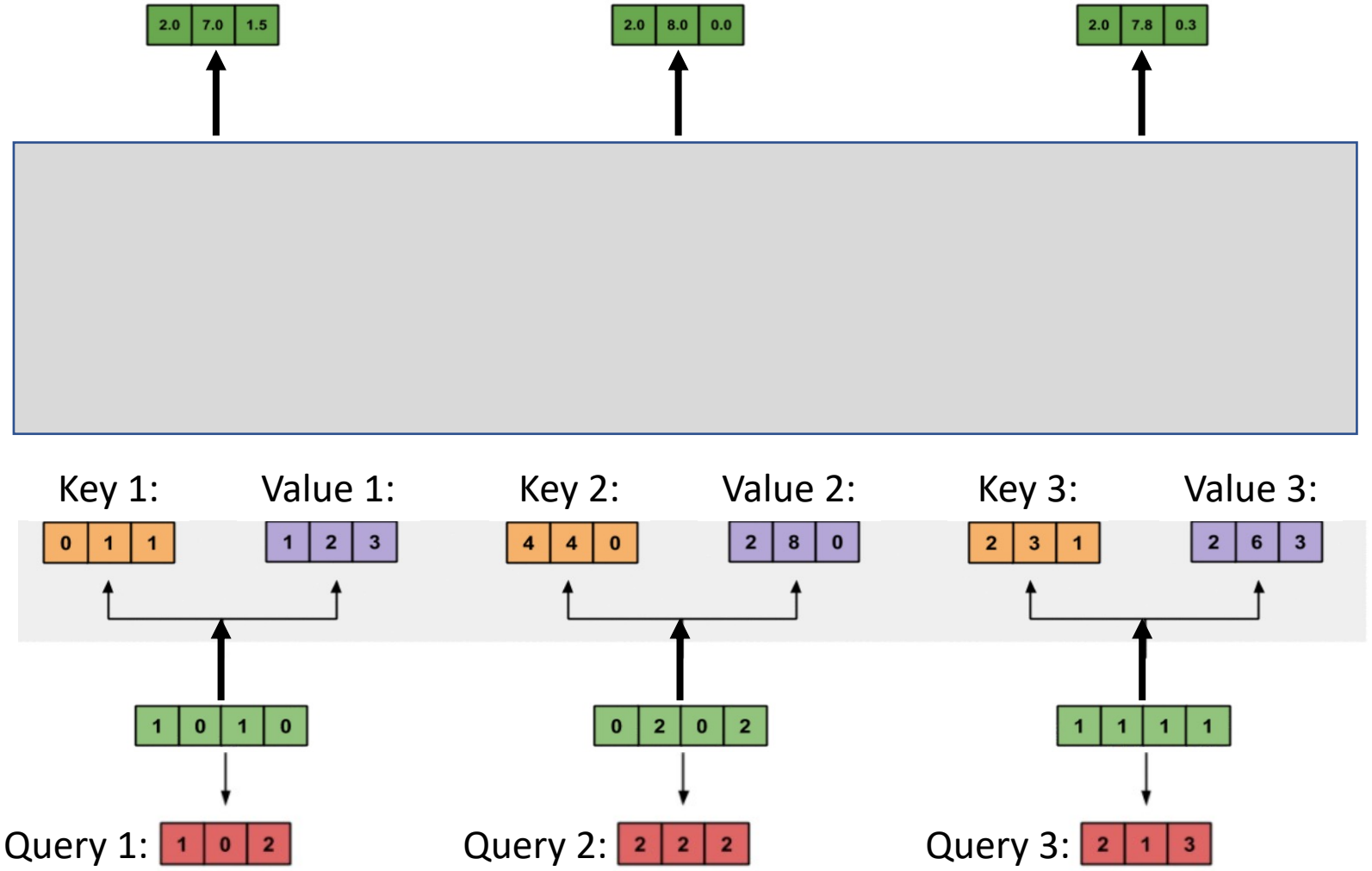
e.g., query weights

```
[1, 0, 1]
[1, 0, 0]
[0, 0, 1]
[0, 1, 1]
```



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example



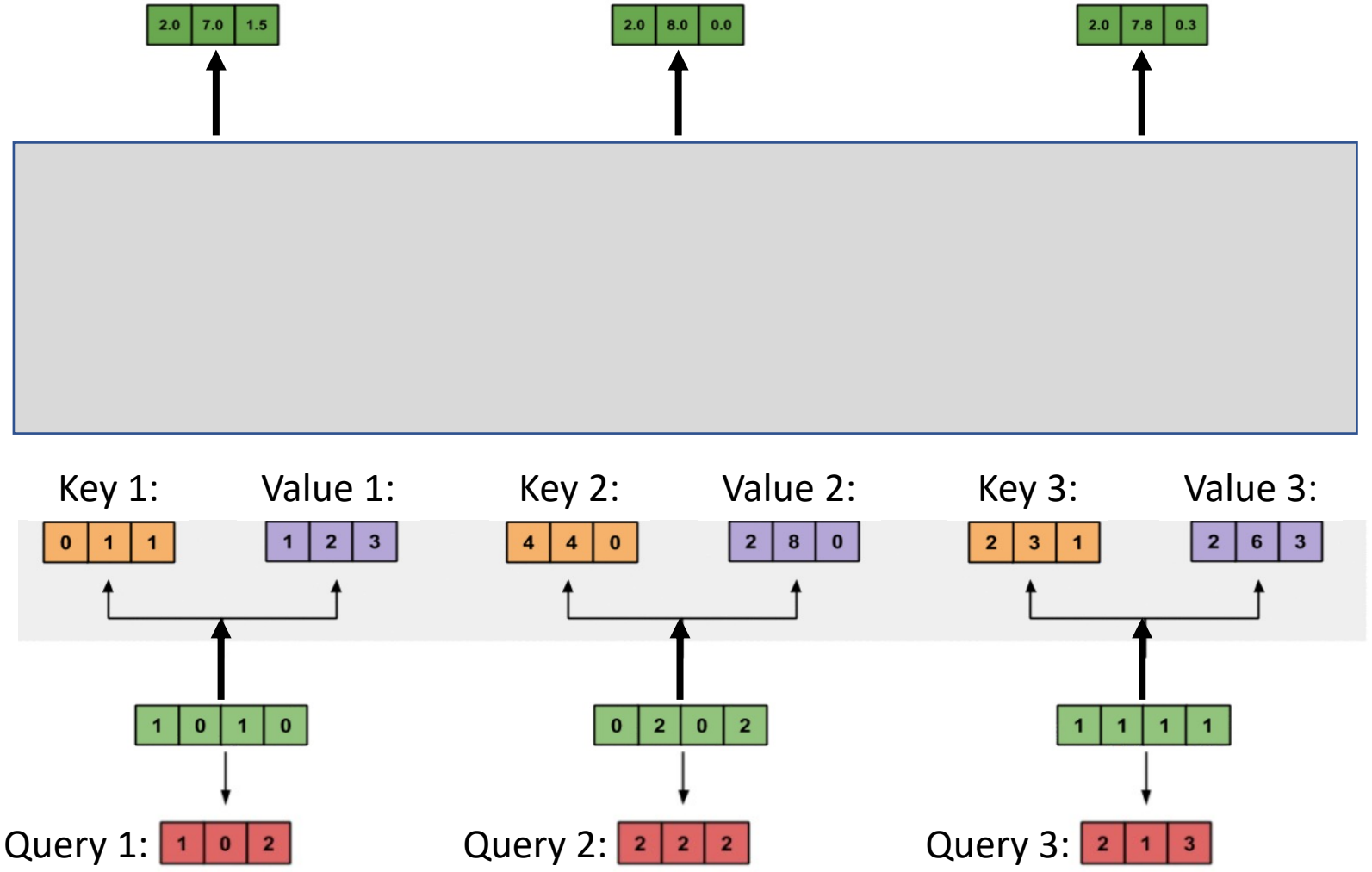
How many weight matrices are learned in this example?

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

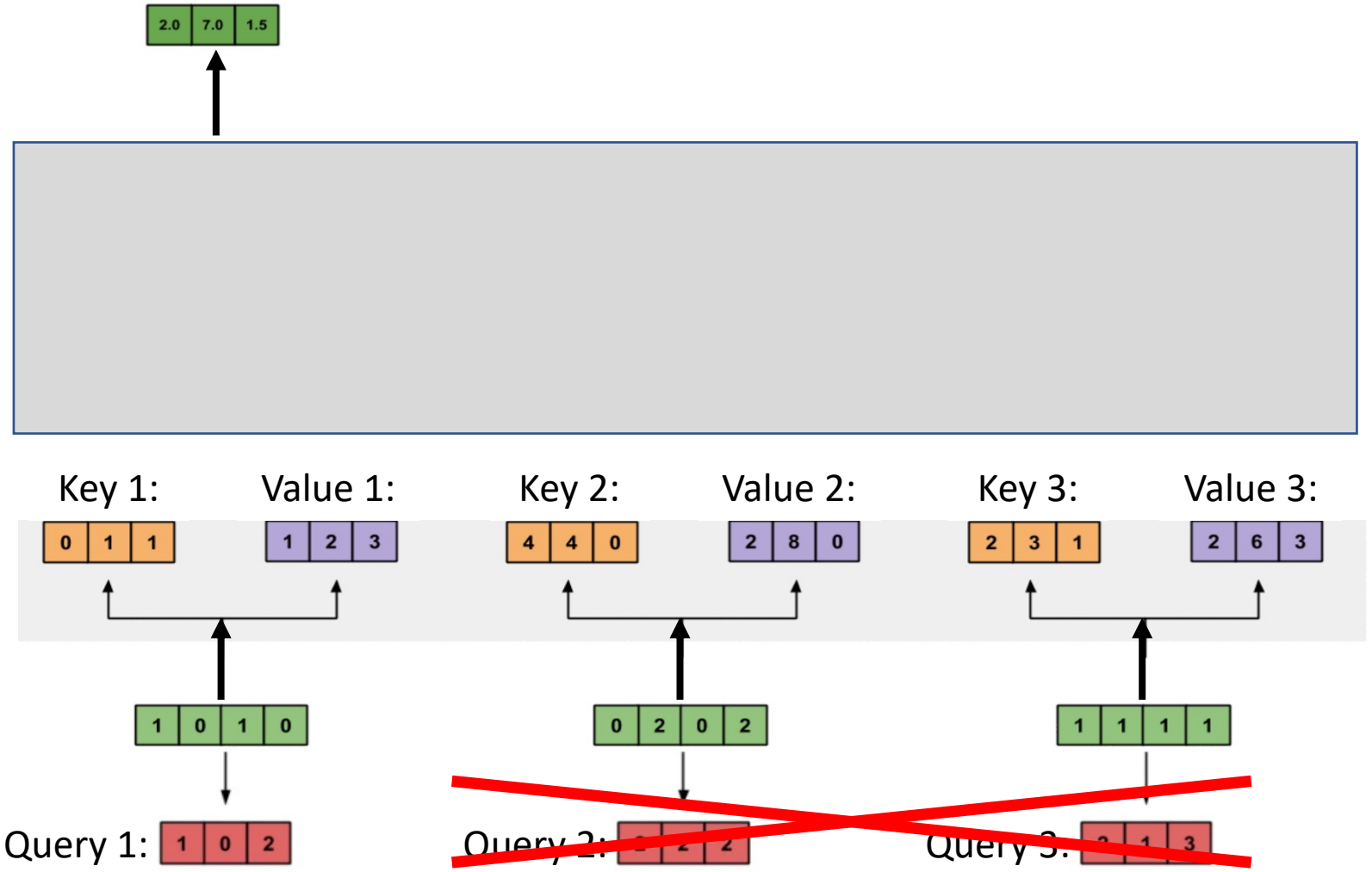
Why do we learn the three weight matrices?

For each input, 2 of the derived vectors are used to compute **attention weights** (query and key) and the 3rd is **information** passed on for the new representation (value)



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example



We now will examine how to find the new representation for the first input.

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

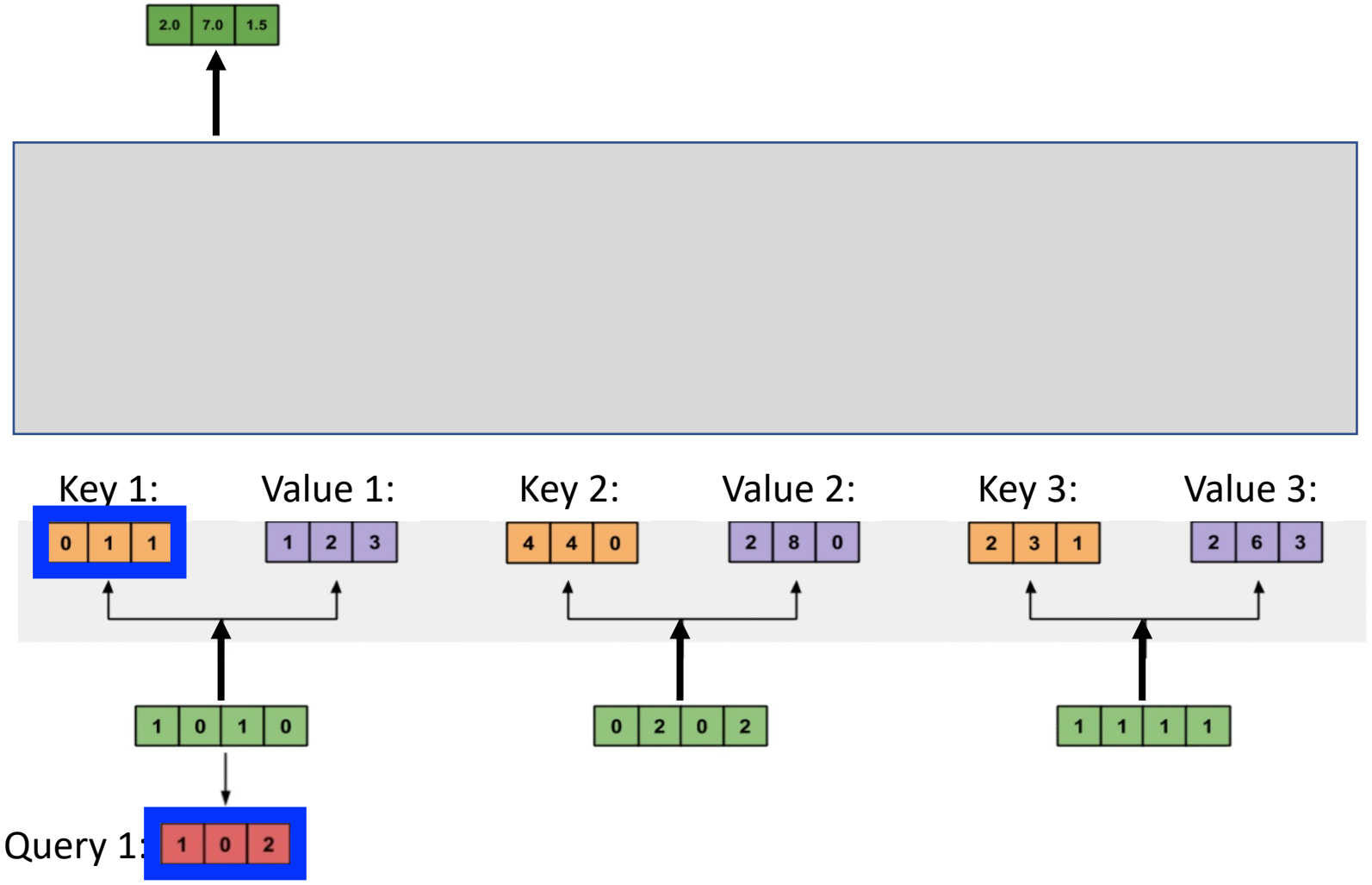
Attention score: dot product of query with all keys to identify relevant tokens; e.g.,

1	0	2

x

0
1
1

= ?



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

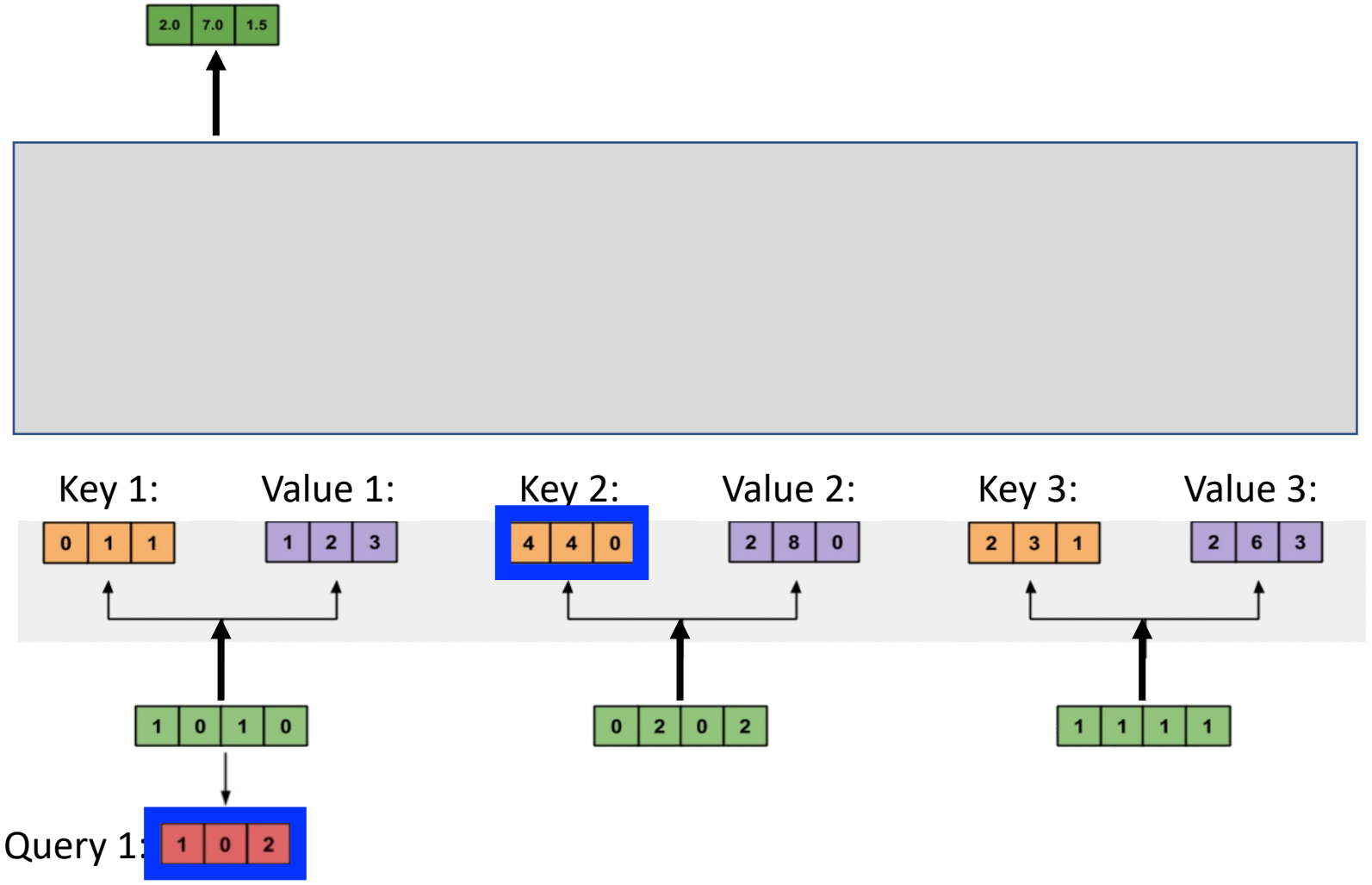
Attention score: dot product of query with all keys to identify relevant tokens; e.g.,

1	0	2

x

4
4
0

= ?



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

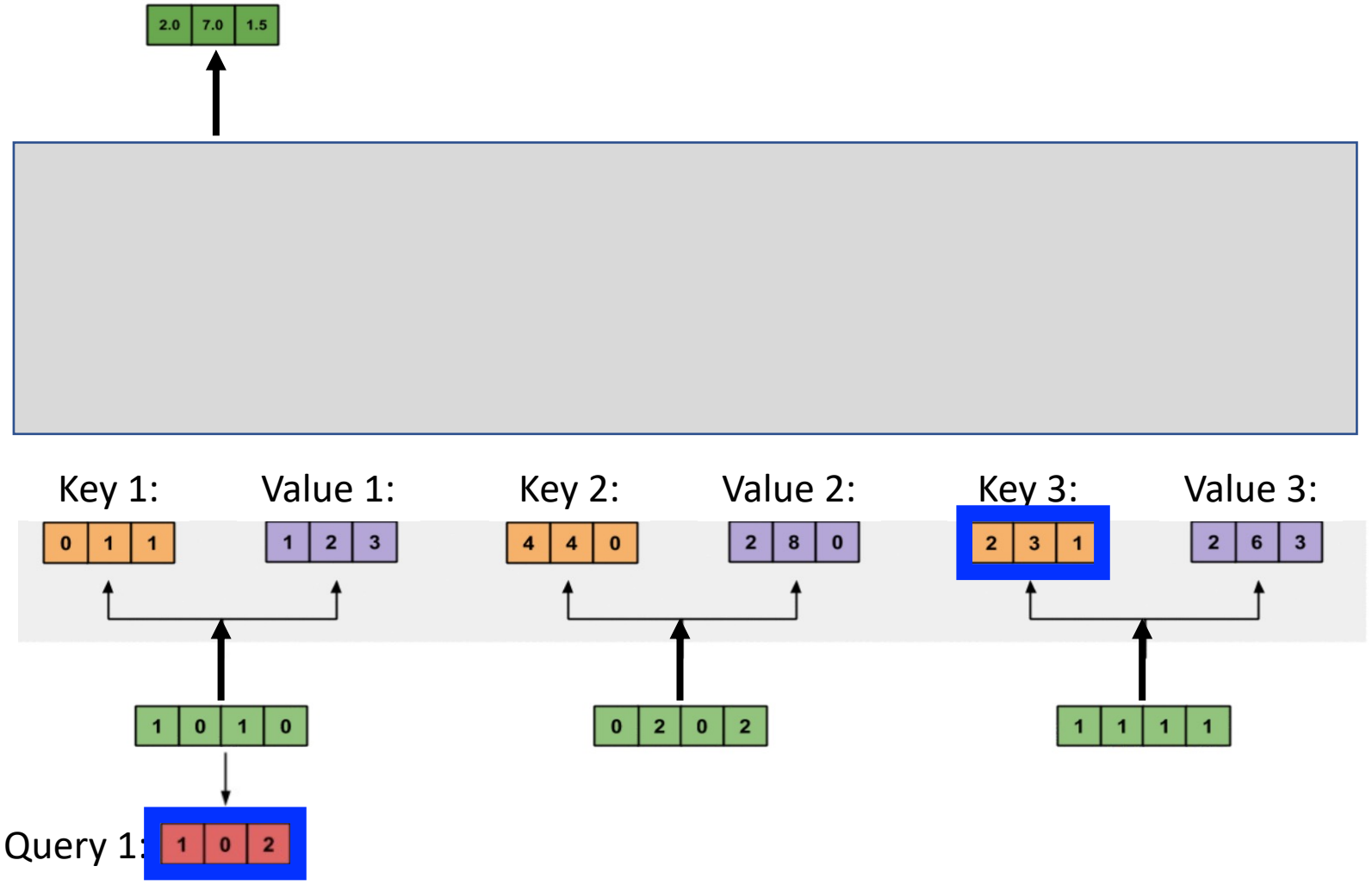
Attention score: dot product of query with all keys to identify relevant tokens; e.g.,

1	0	2

x

2
3
1

= ?

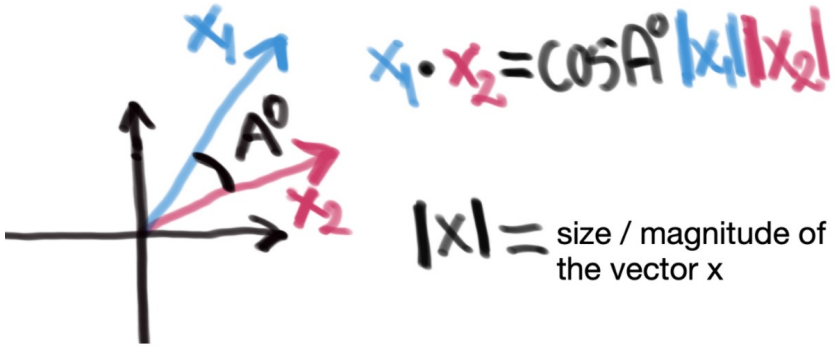


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

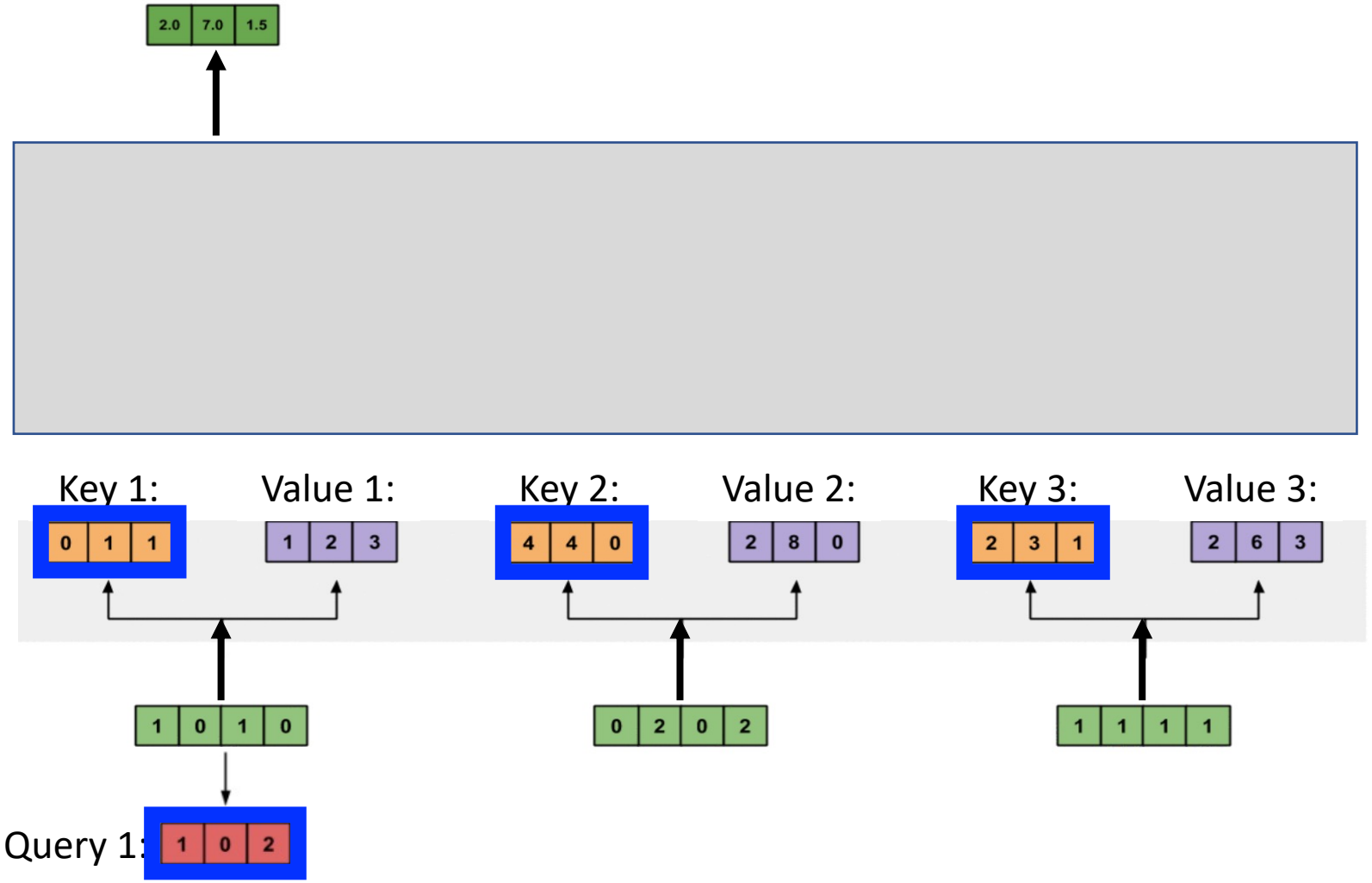
Computing Self-Attention: Example

Why dot product? Indicates similarity of two vectors

- Match = 1 (i.e., cos(0))
- Opposites = -1 (i.e., cos(180))



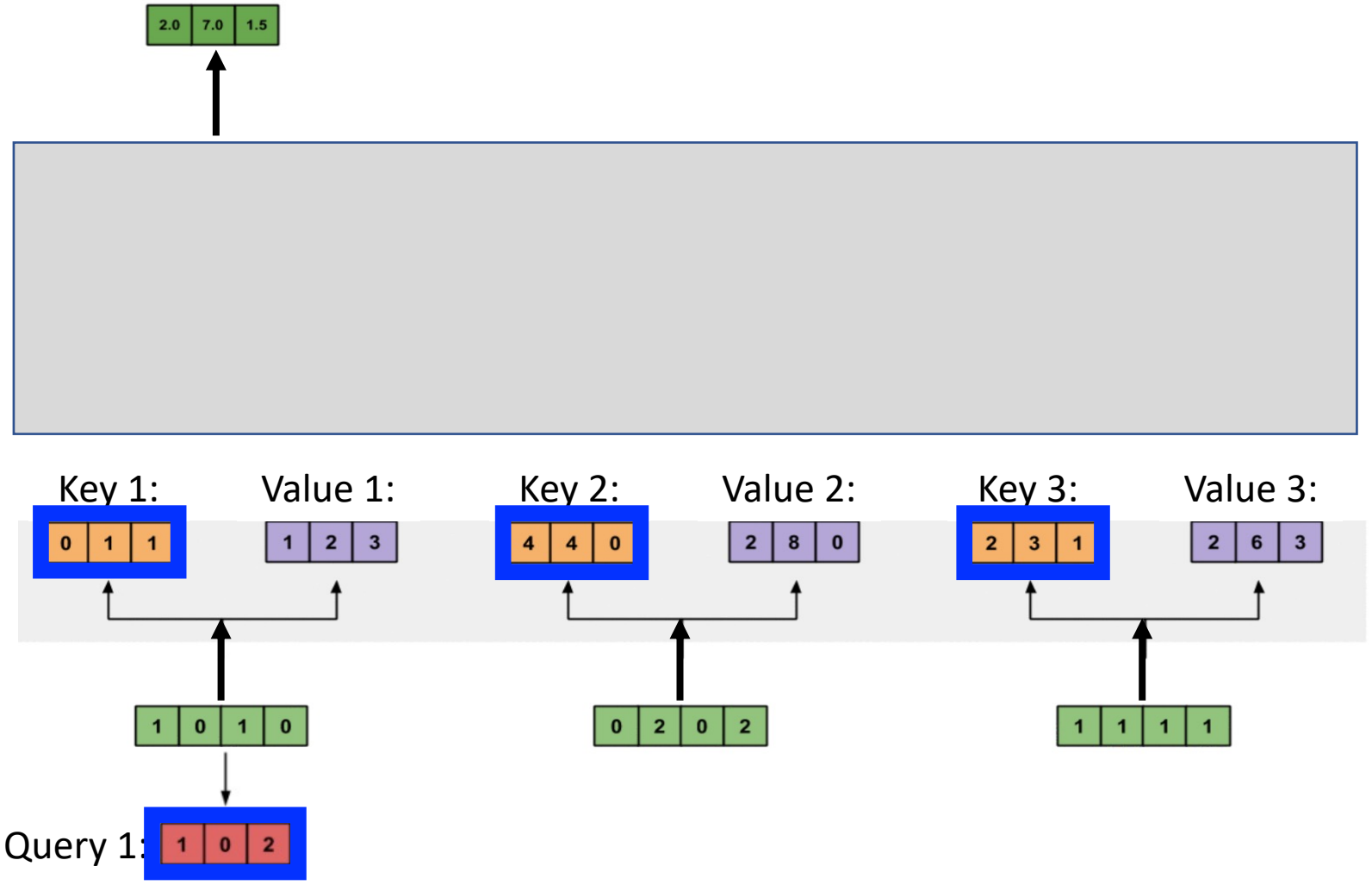
https://towardsdatascience.com/self-attention-5b95ea164f61



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Can also use similarity measures other than the dot product



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

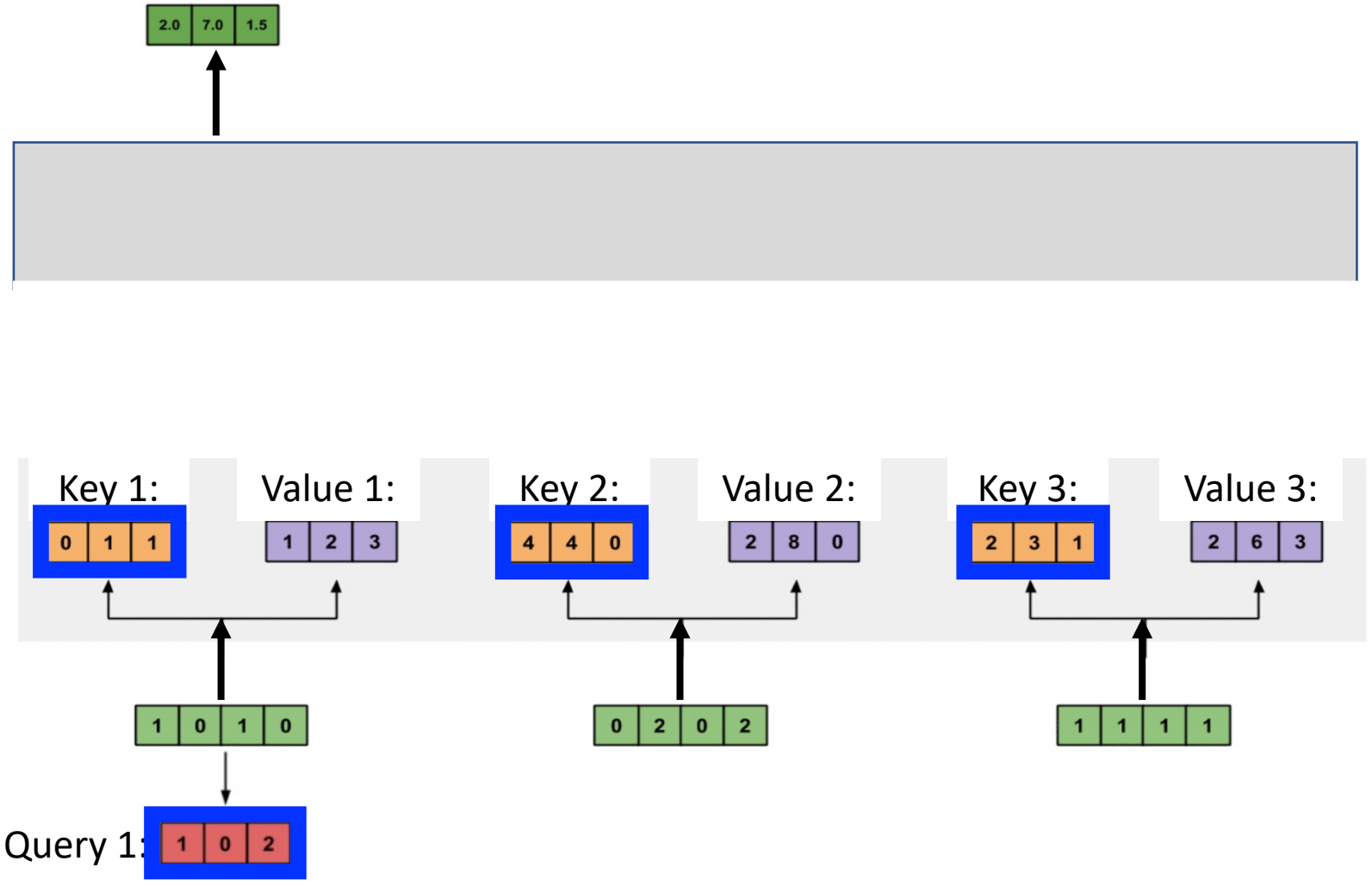
Computing Self-Attention: Example

Attention weights: softmax scores for all inputs to quantify each token's relevance; e.g.,

= softmax([2, 4, 4])

= [0.0, 0.5, 0.5])

To which input(s) is input 1 most related?

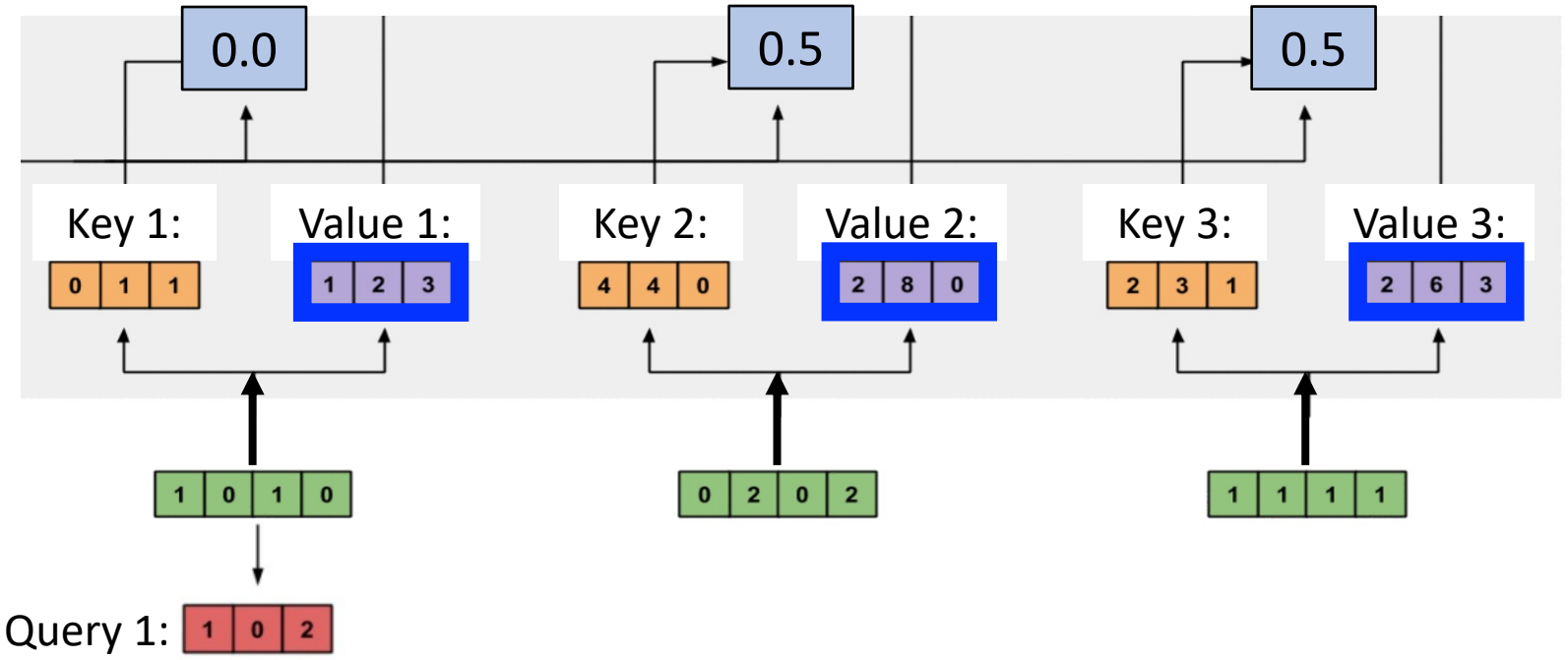


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Compute new representation of input token that reflects entire input:

1. Attention weights x Values



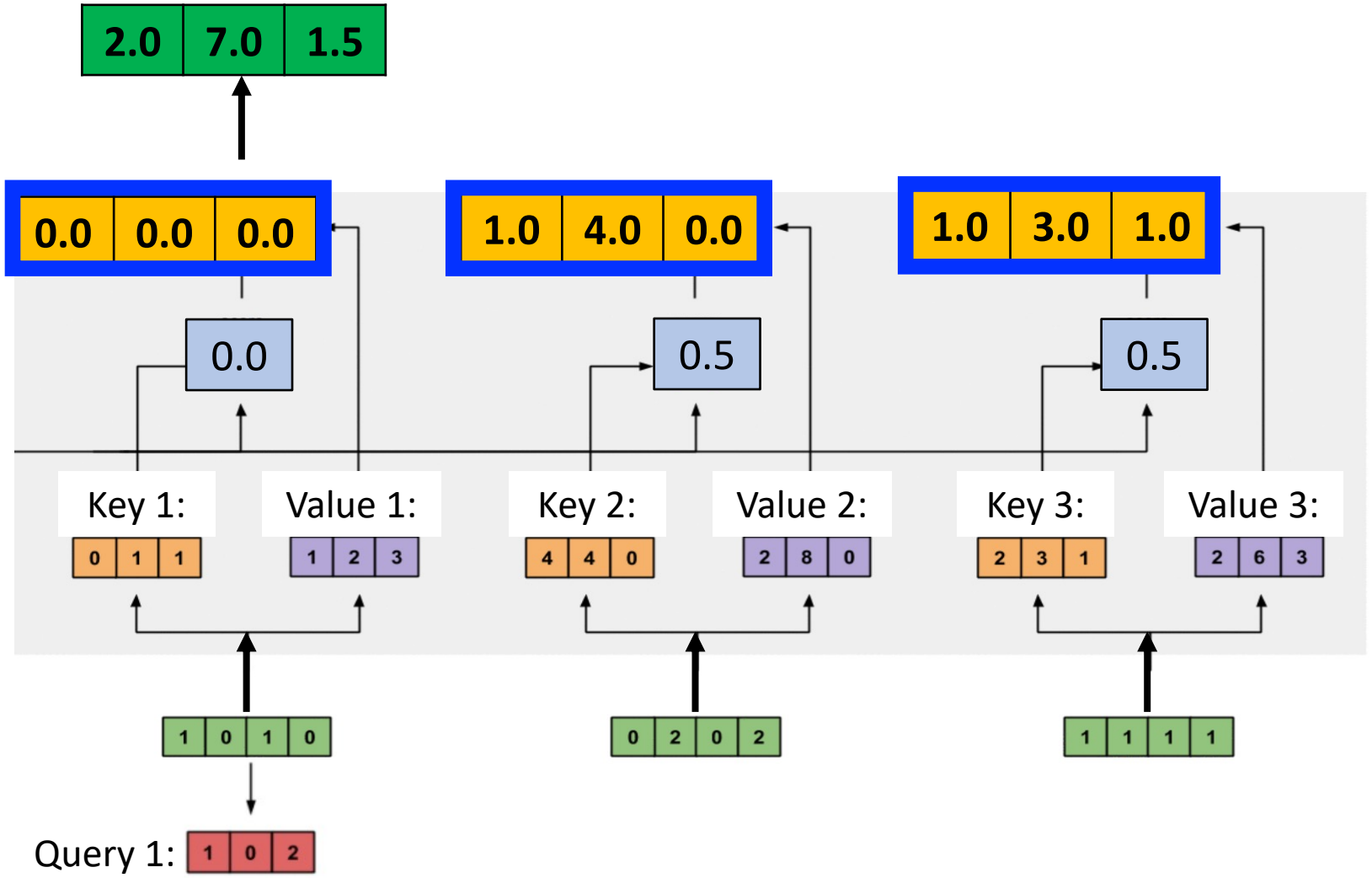
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Compute new representation of input token that reflects entire input:

1. Attention weights x Values

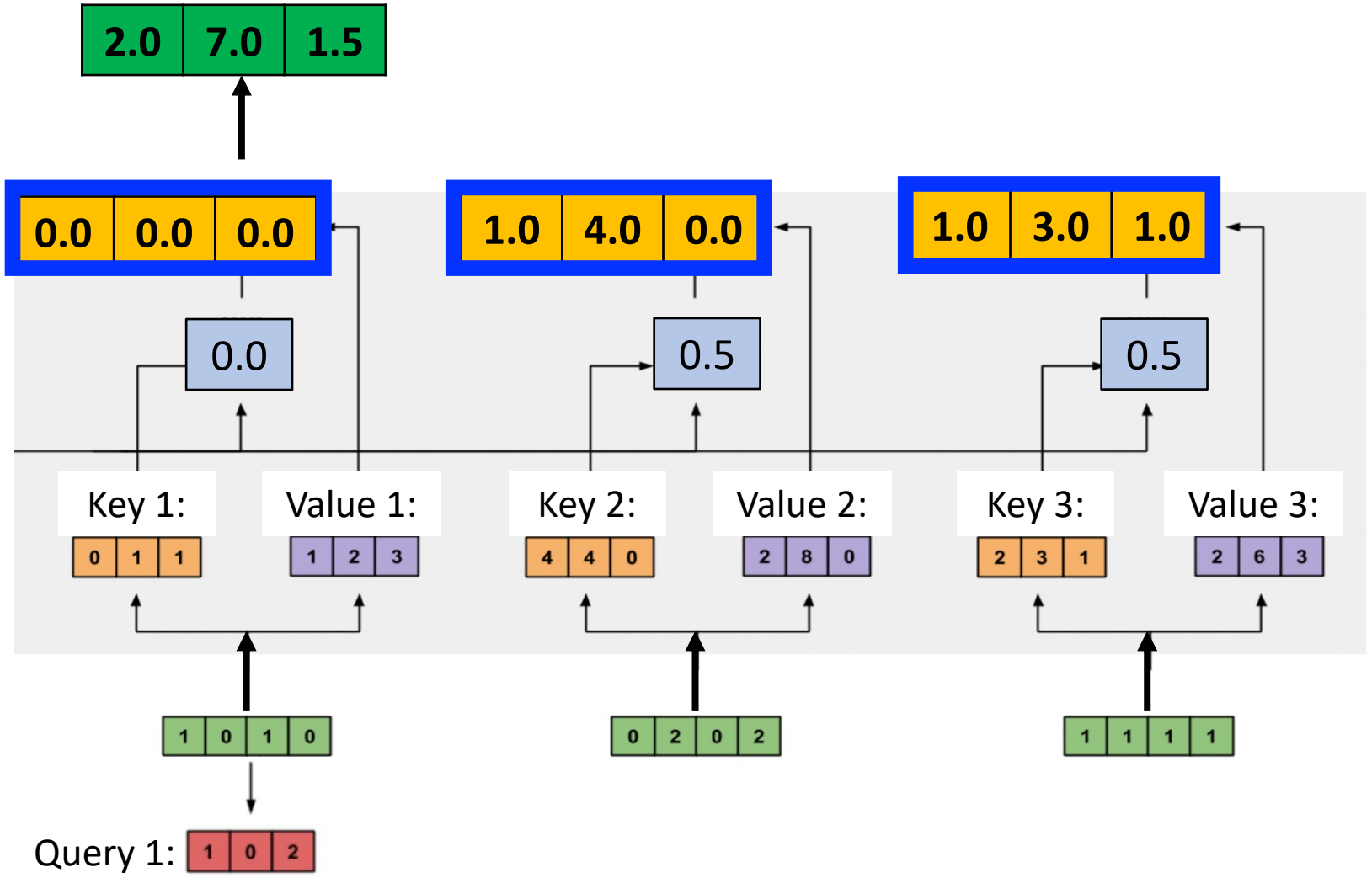
2. Sum all weighted vectors



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

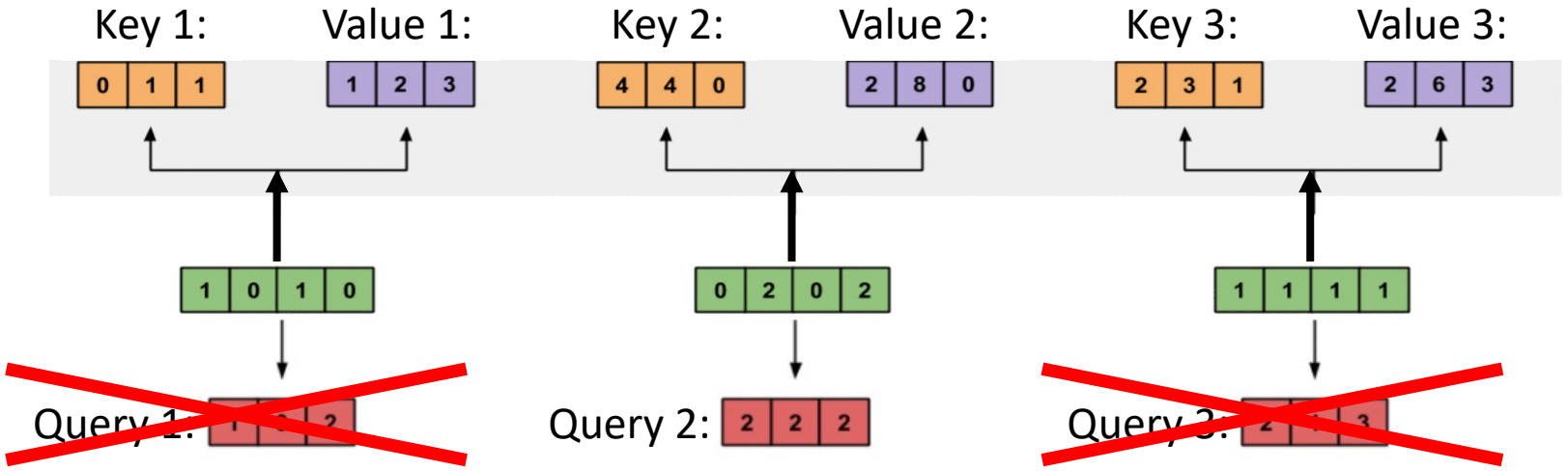
Attention weights amplify input representations (values) that we want to pay attention to and repress the rest



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Repeat the same process for each remaining input token

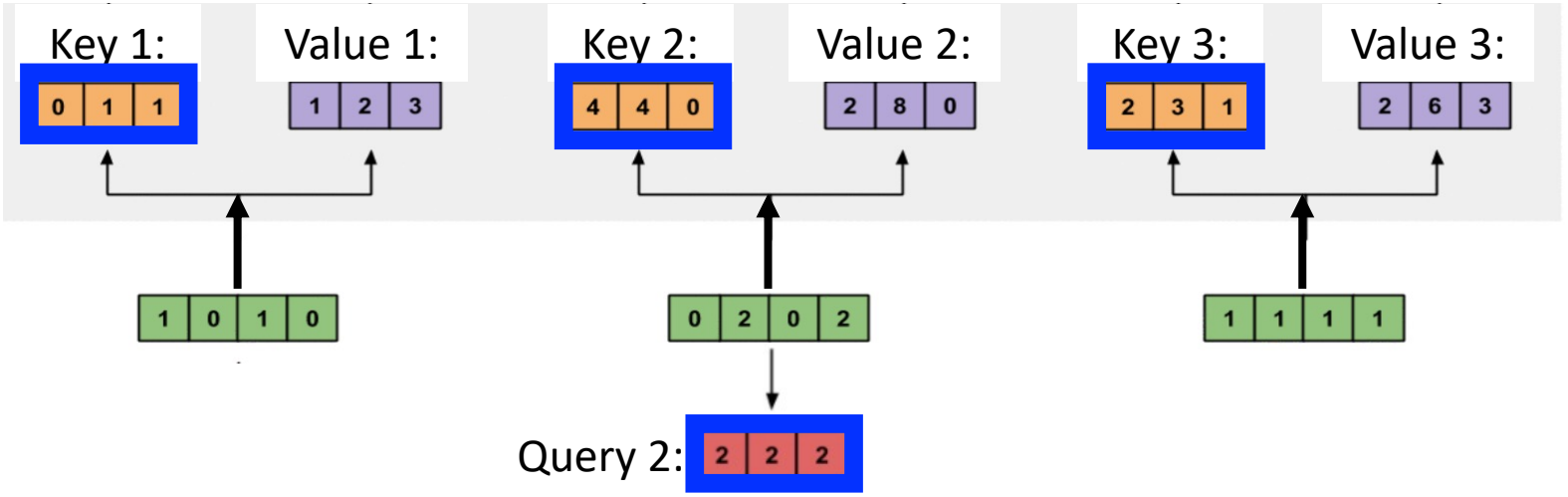


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

To which input(s) is input 2 most related?`

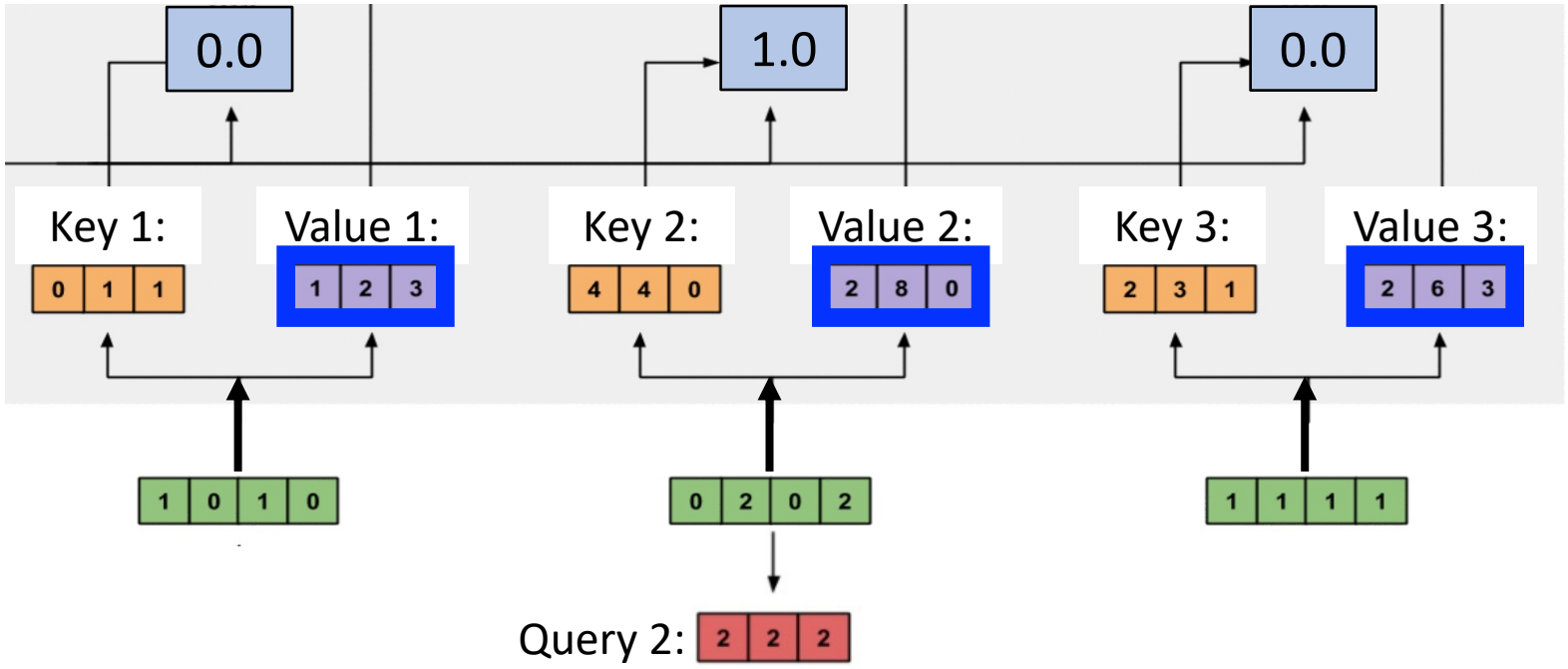


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

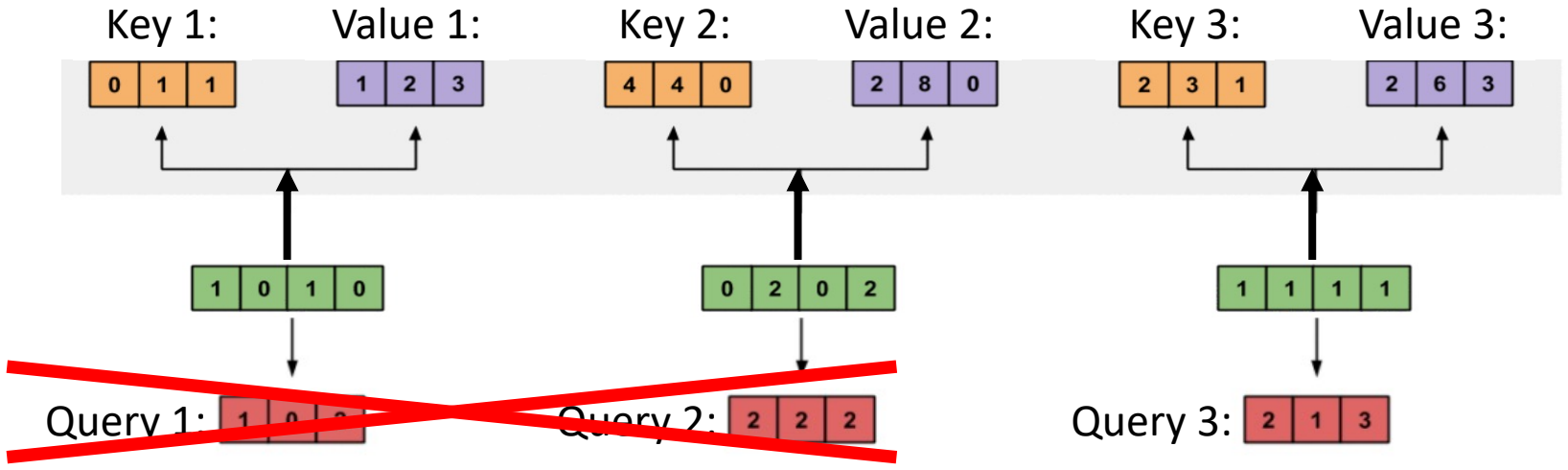
2. Compute weighted sum of values using attention scores



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Repeat the same process for each remaining input token

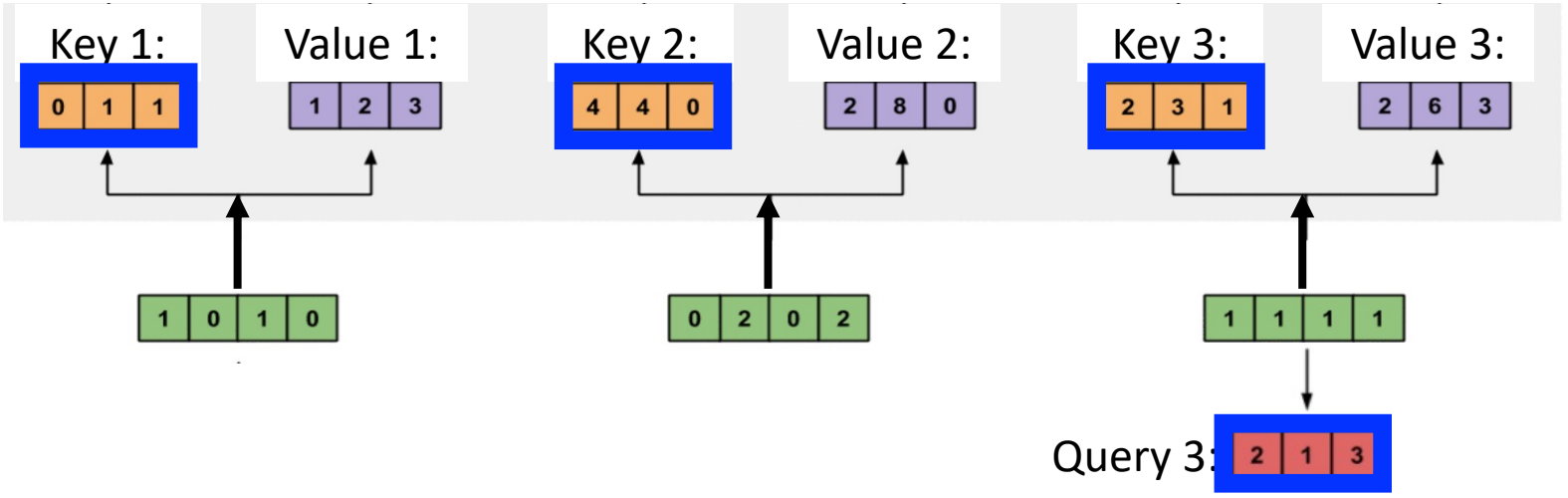


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

To which input(s) is input 3 most related?

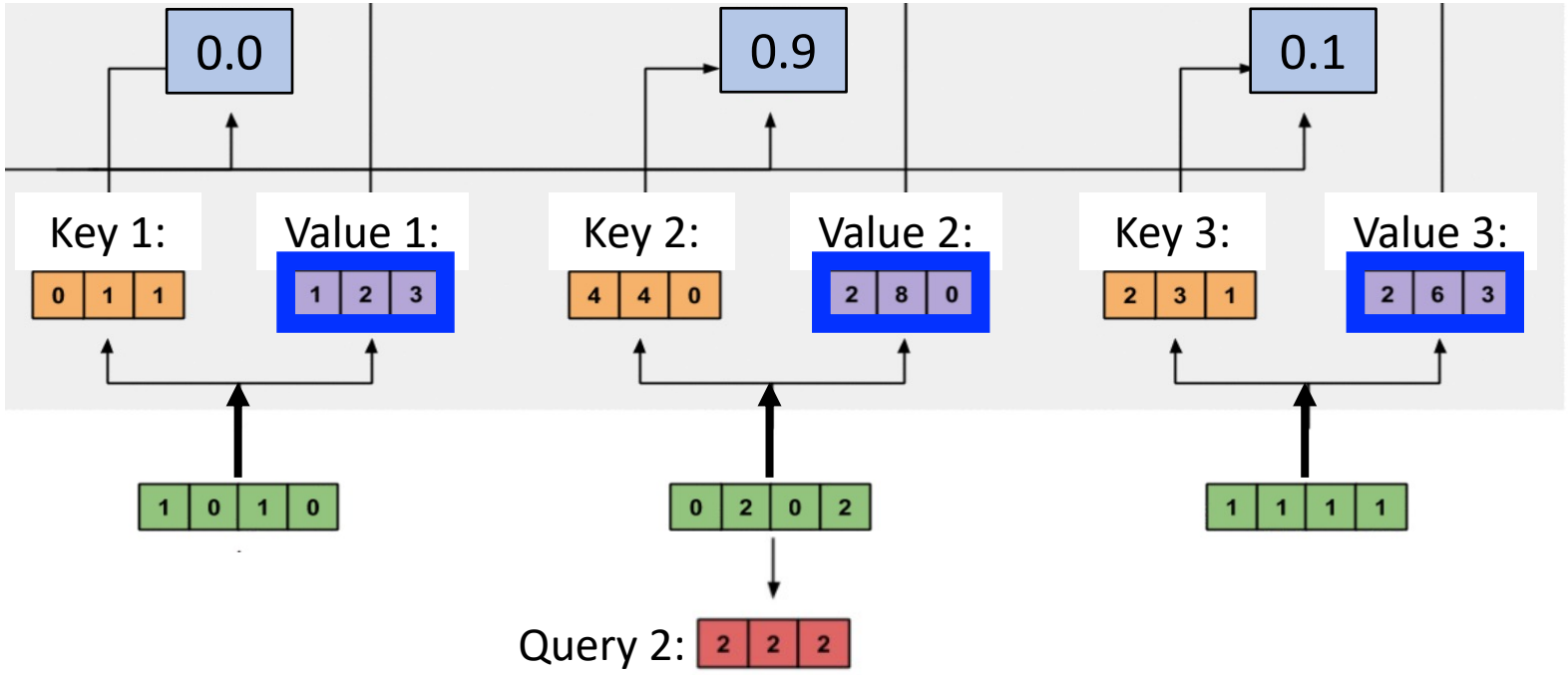


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

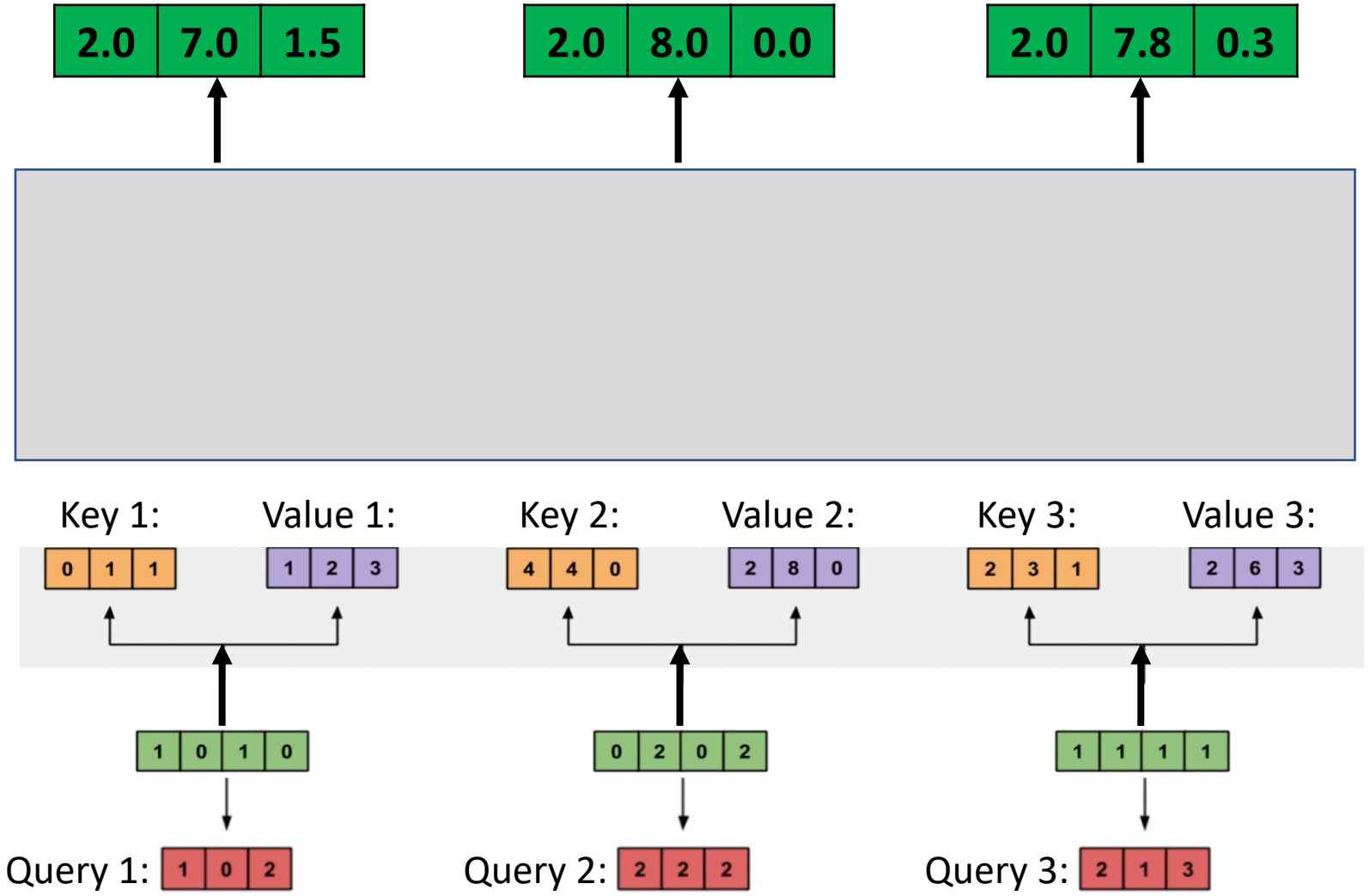
1. Compute attention weights
- Softmax resulting 3 scores from query x keys

2. Compute weighted sum of values using attention scores



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

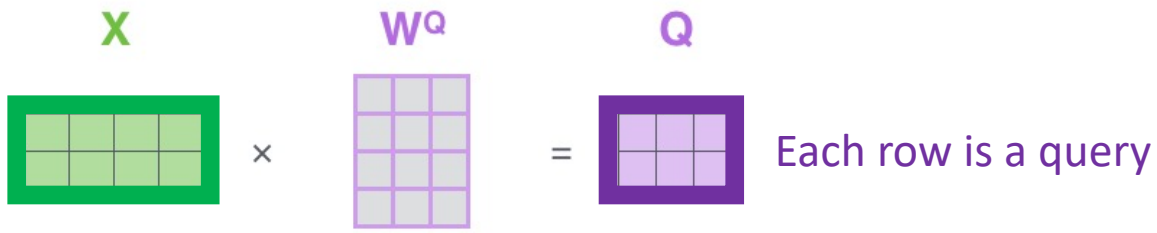


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

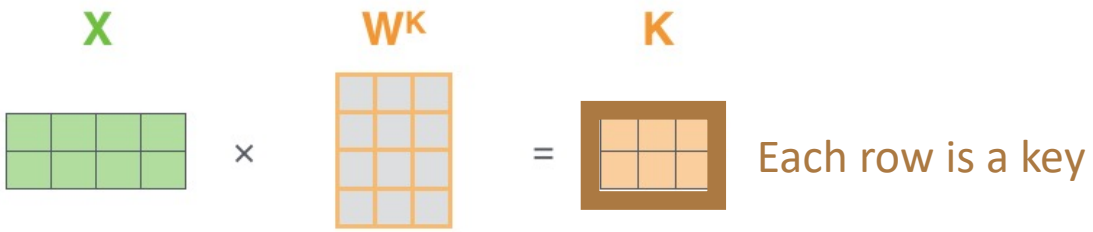
Efficient Computation for Self-Attention

Step 1

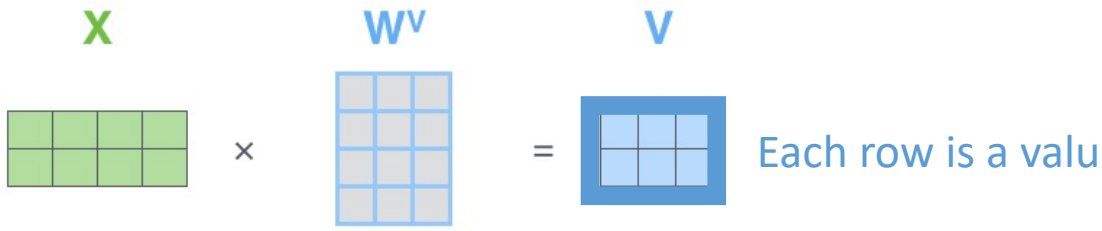
Each row is an input token:



Each row is a query



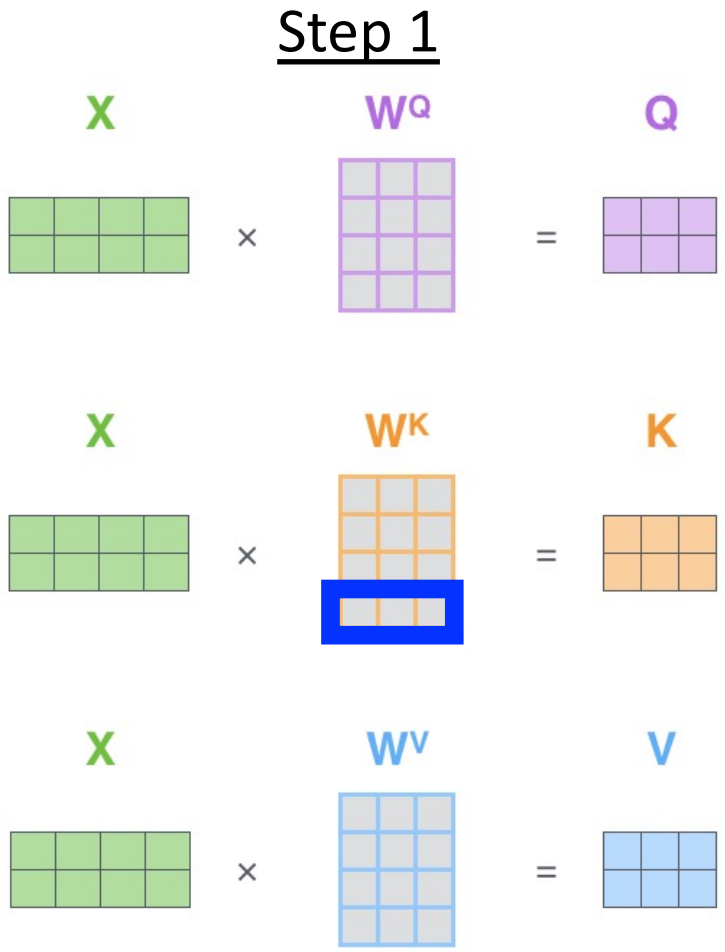
Each row is a key

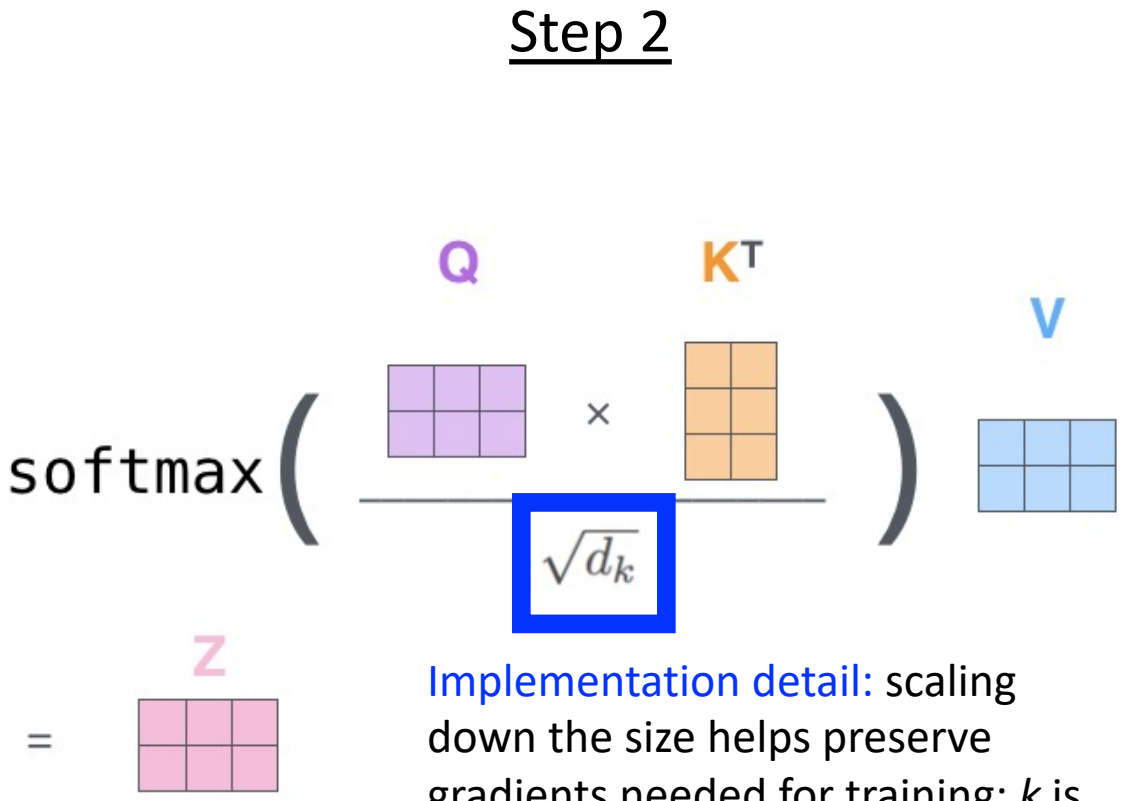


Each row is a value

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

Efficient Computation for Self-Attention

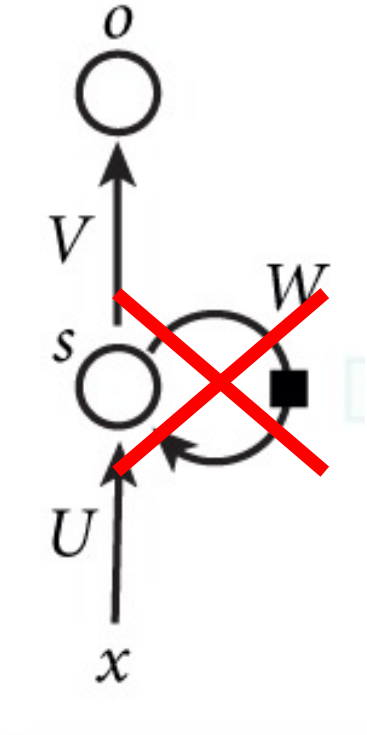




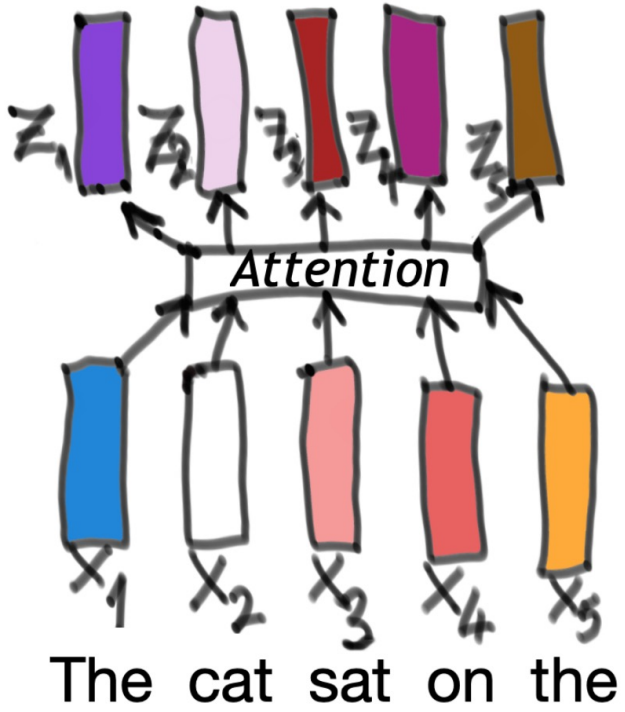
Implementation detail: scaling down the size helps preserve gradients needed for training; *k* is dimensionality of the key vector

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

Self-Attention vs RNN: Propagates Information About Other Inputs Without Recurrent Units



http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/



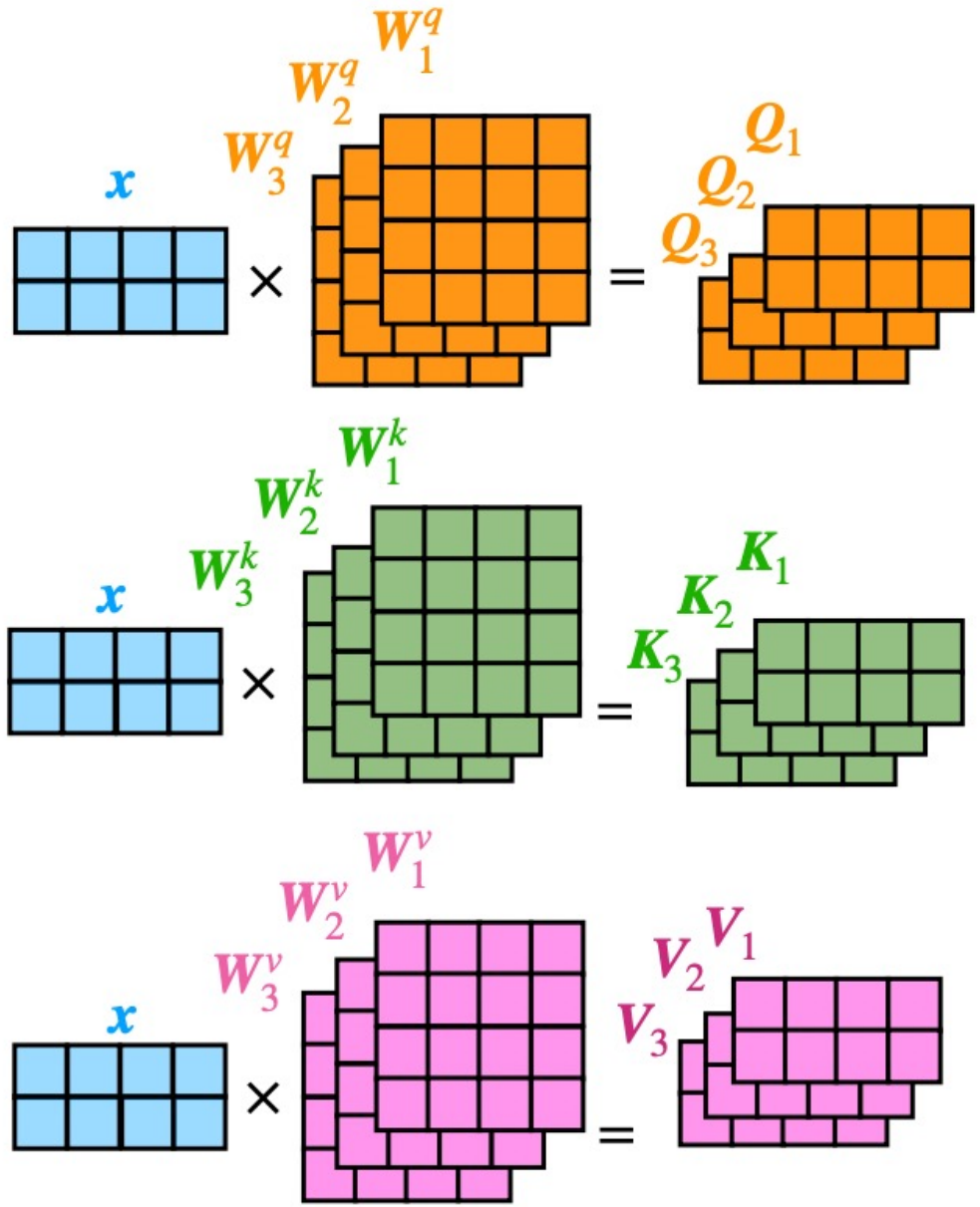
https://towardsdatascience.com/self-attention-5b95ea164f61

Today's Topics

- Transformer overview
- Self-attention
- Multi-head attention
- Common transformer ingredients
- Pioneering transformer: machine translation

Multi-head Attention

- **Goal**: enable each token to relate to other tokens in multiple ways
- **Key idea**: multiple self-attention mechanisms, each with their own key, value and query matrices



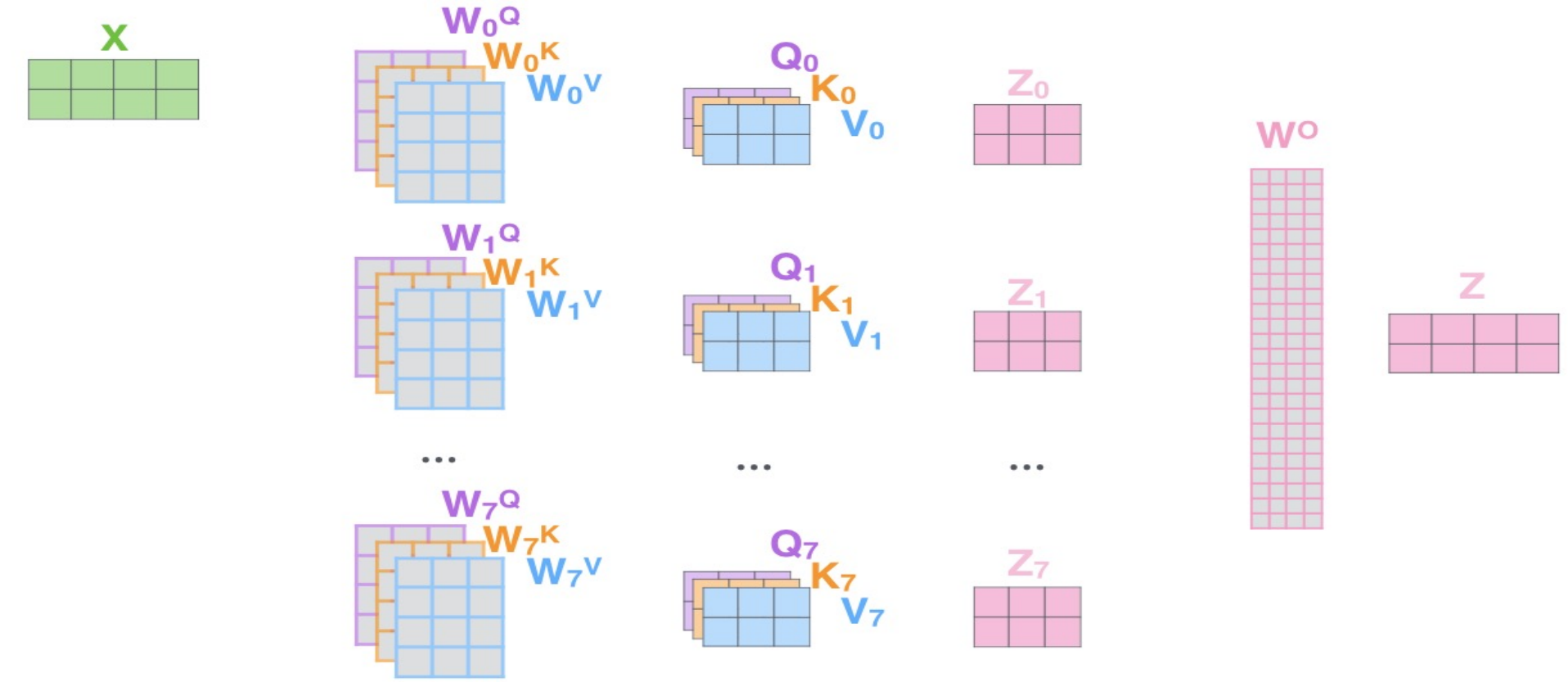
https://sebastianraschka.com/pdf/lecture-notes/stat453ss21/L19_seq2seq_rnn-transformers__slides.pdf

Multi-head Attention

1) Create query, key, and value vectors for all attentions heads

2) Compute new input representations

3) Condense all representations into a single representation by concatenating **z**-s and multiplying by a weight matrix



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

Trained Multi-head Attention Examples

Figure shows two columns of attention weights for the first two attention heads

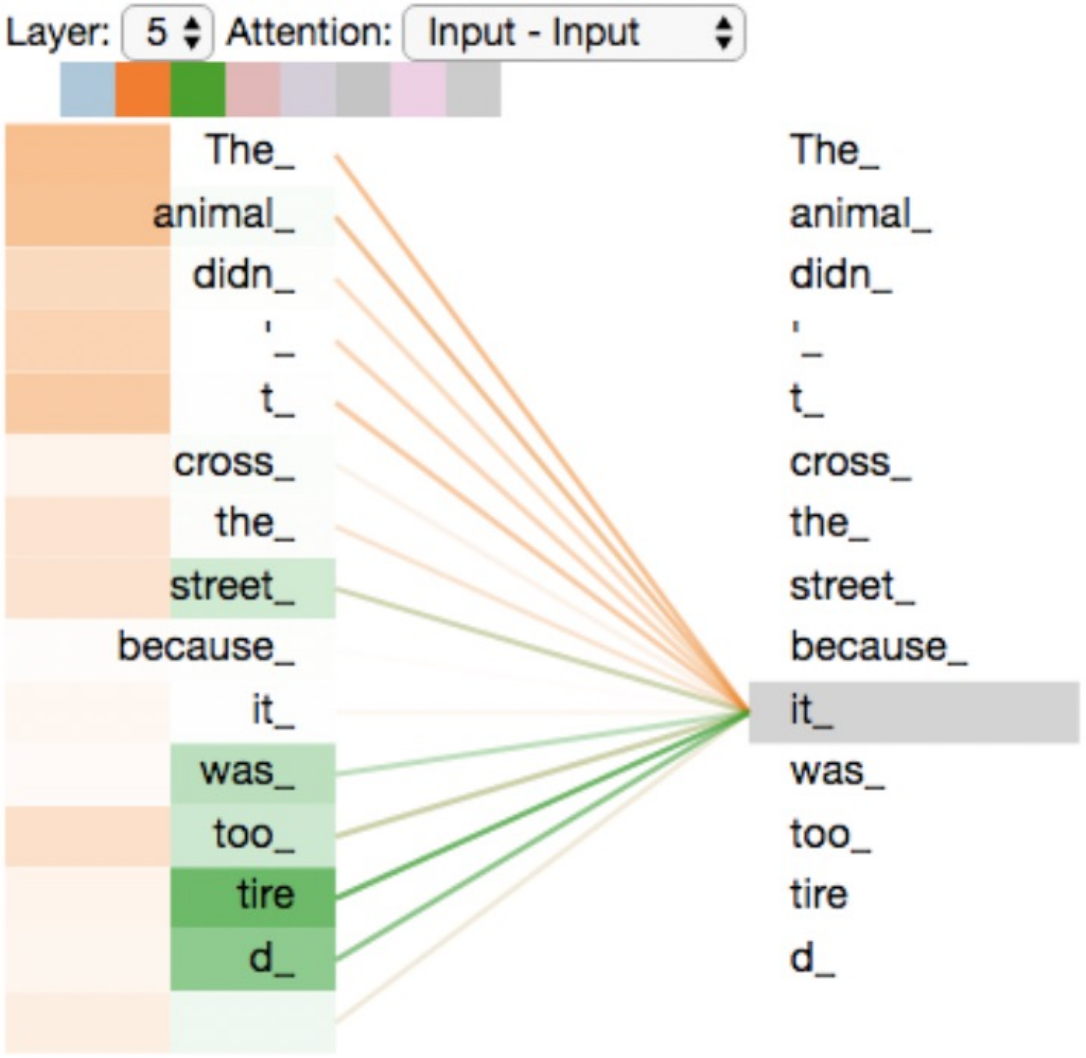
- Darker values signify larger attention scores

What does “it” focus on most in the first attention head?

- The animal (e.g., represents what is “it”)

What does “it” focus on most in the second attention head?

- tired (e.g., represents how “it” feels)



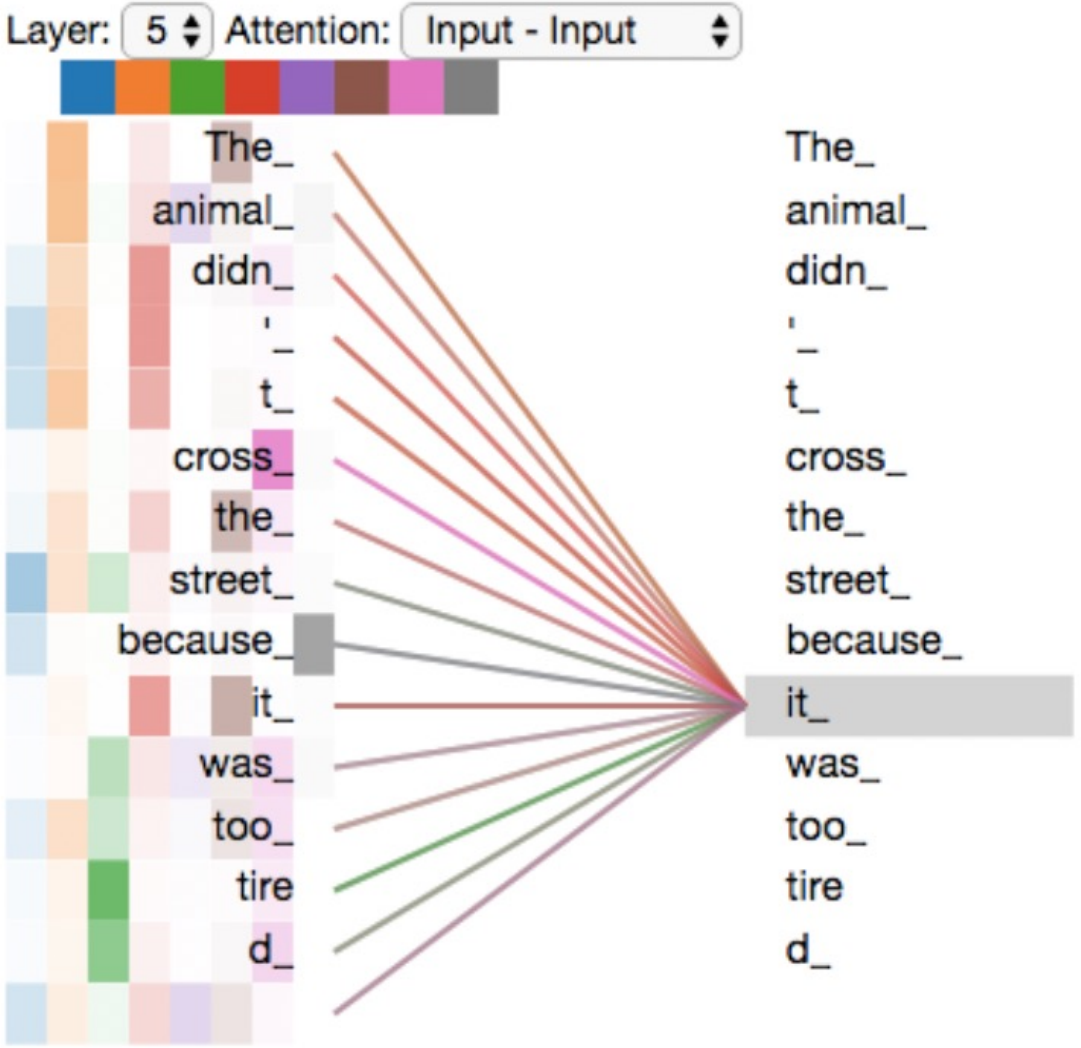
http://jalammar.github.io/illustrated-transformer/

Trained Multi-head Attention Examples

Figure shows five columns of attention weights for five attention heads

- Darker values signify larger attention scores

Attention weights may be hard to interpret

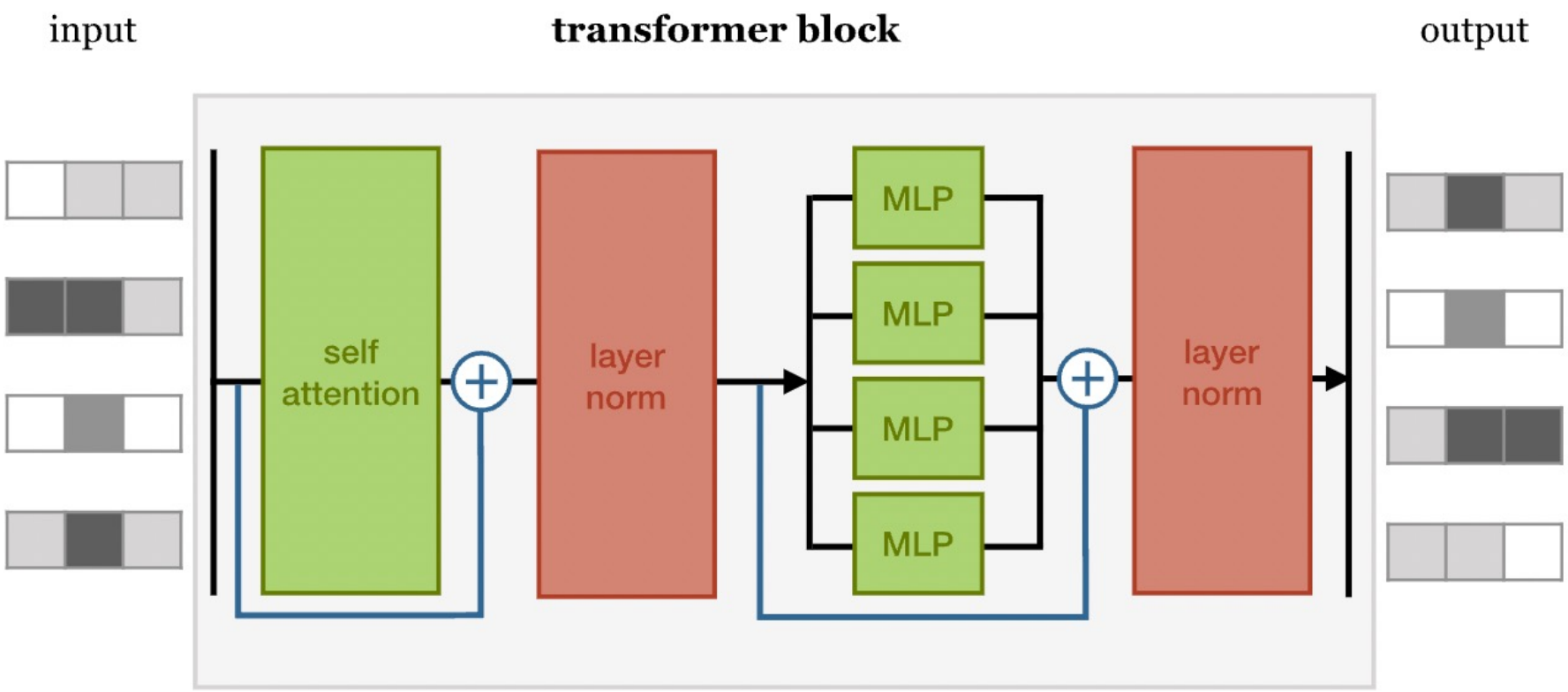


http://jalammar.github.io/illustrated-transformer/

Today's Topics

- Transformer overview
- Self-attention
- Multi-head attention
- Common transformer ingredients
- Pioneering transformer: machine translation

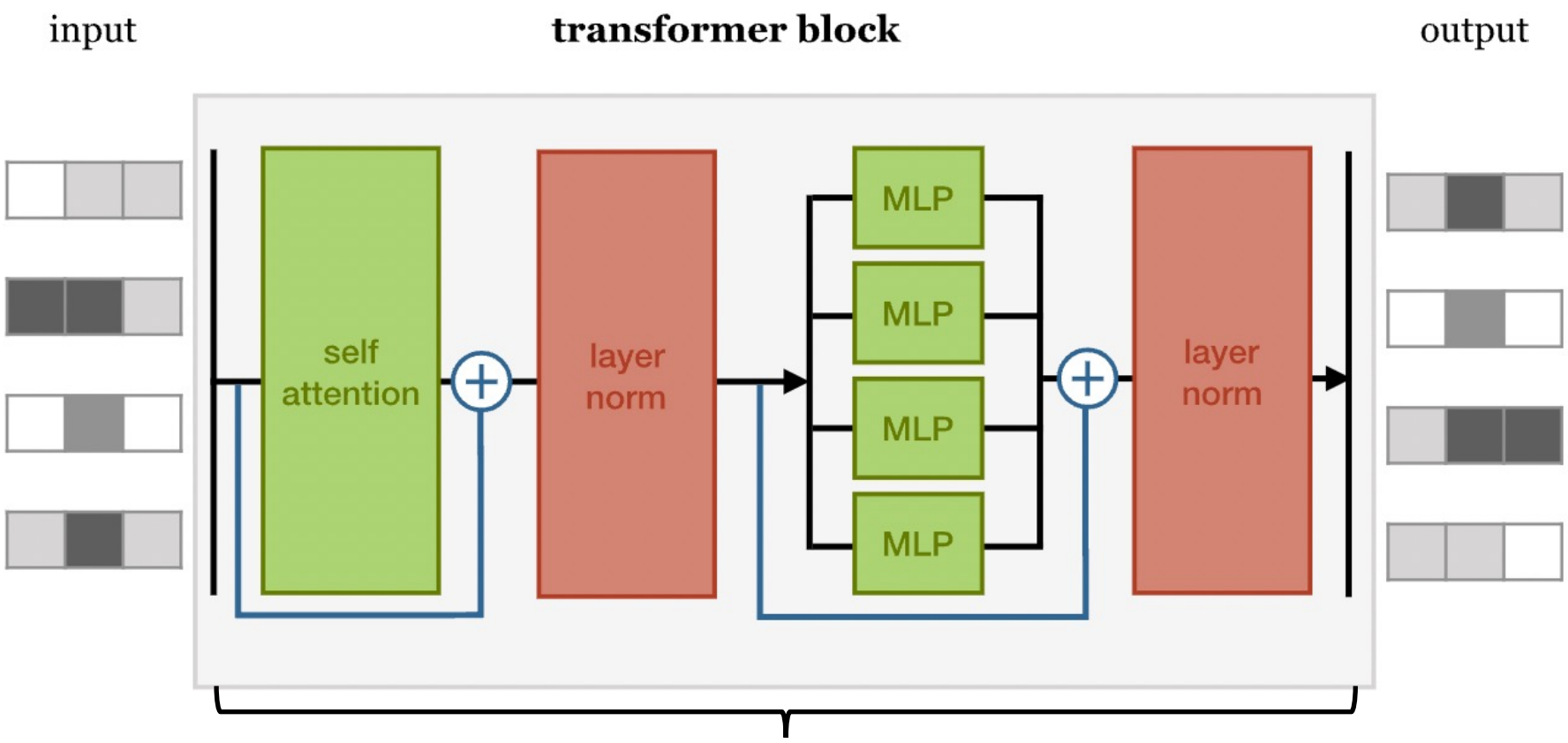
Typical Transformer Block



Architectures often chain together multiple transformer blocks, like that shown here

http://peterbloem.nl/blog/transformers

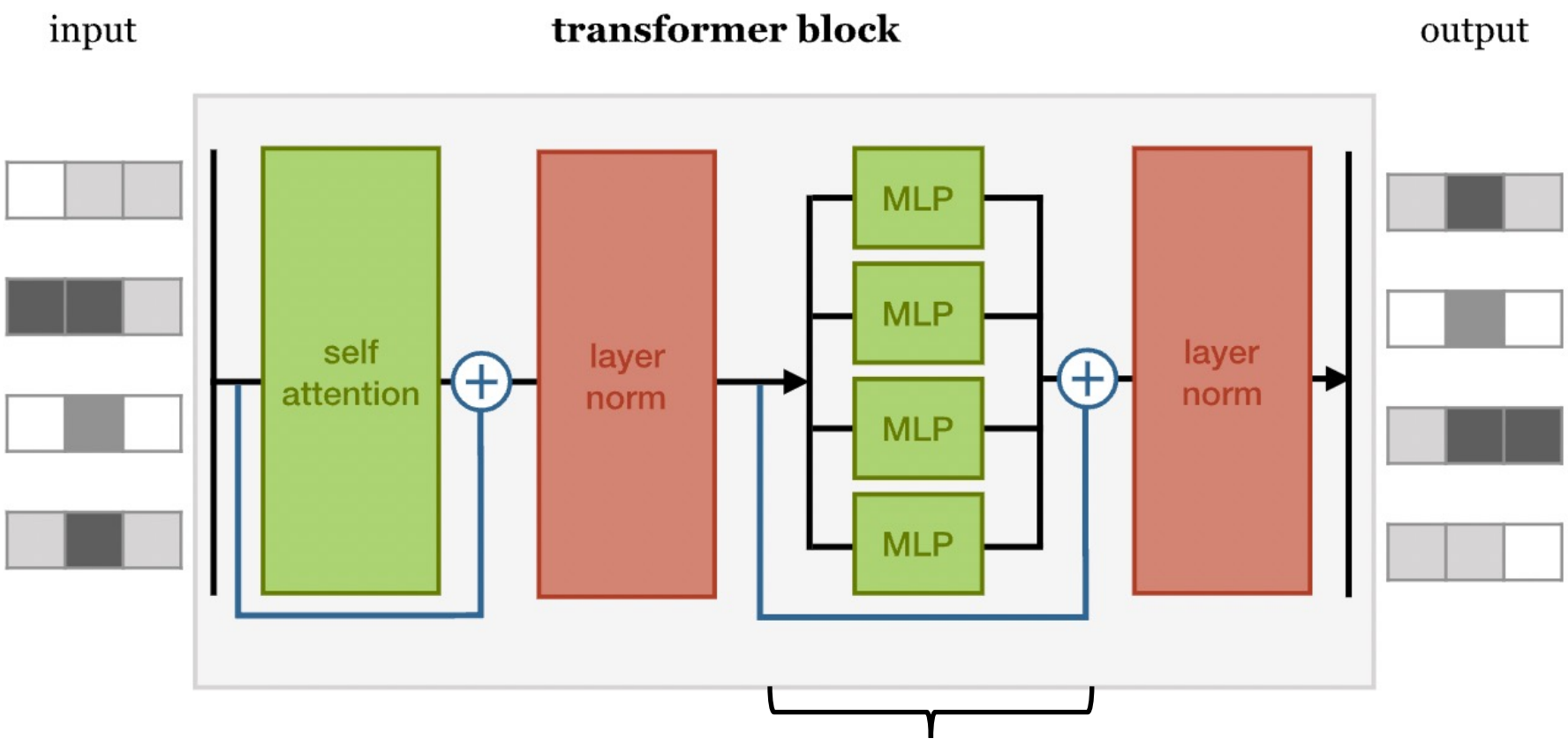
Typical Transformer Block



Layer normalization and residual connections improve training (i.e., faster and better results)

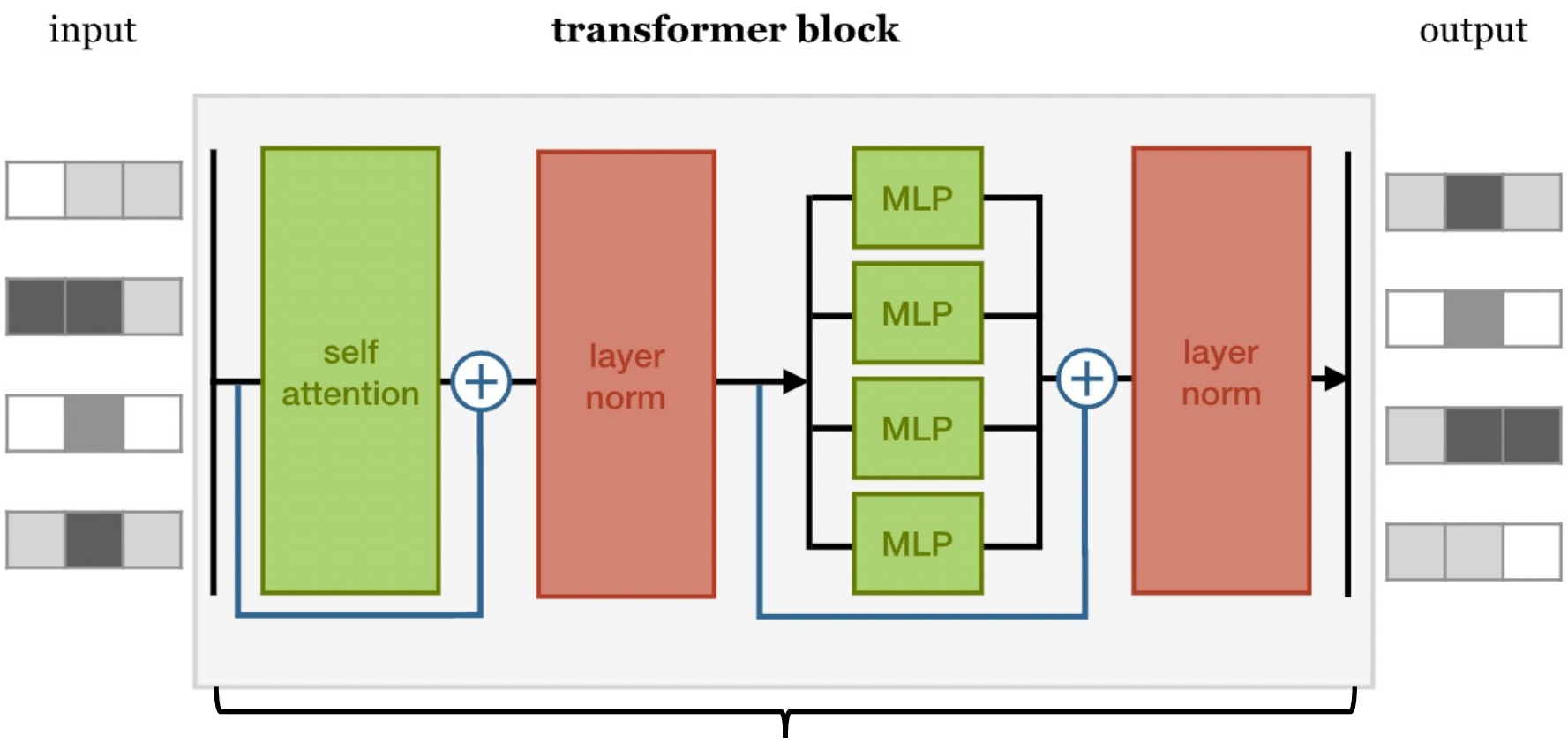
http://peterbloem.nl/blog/transformers

Typical Transformer Block



http://peterbloem.nl/blog/transformers

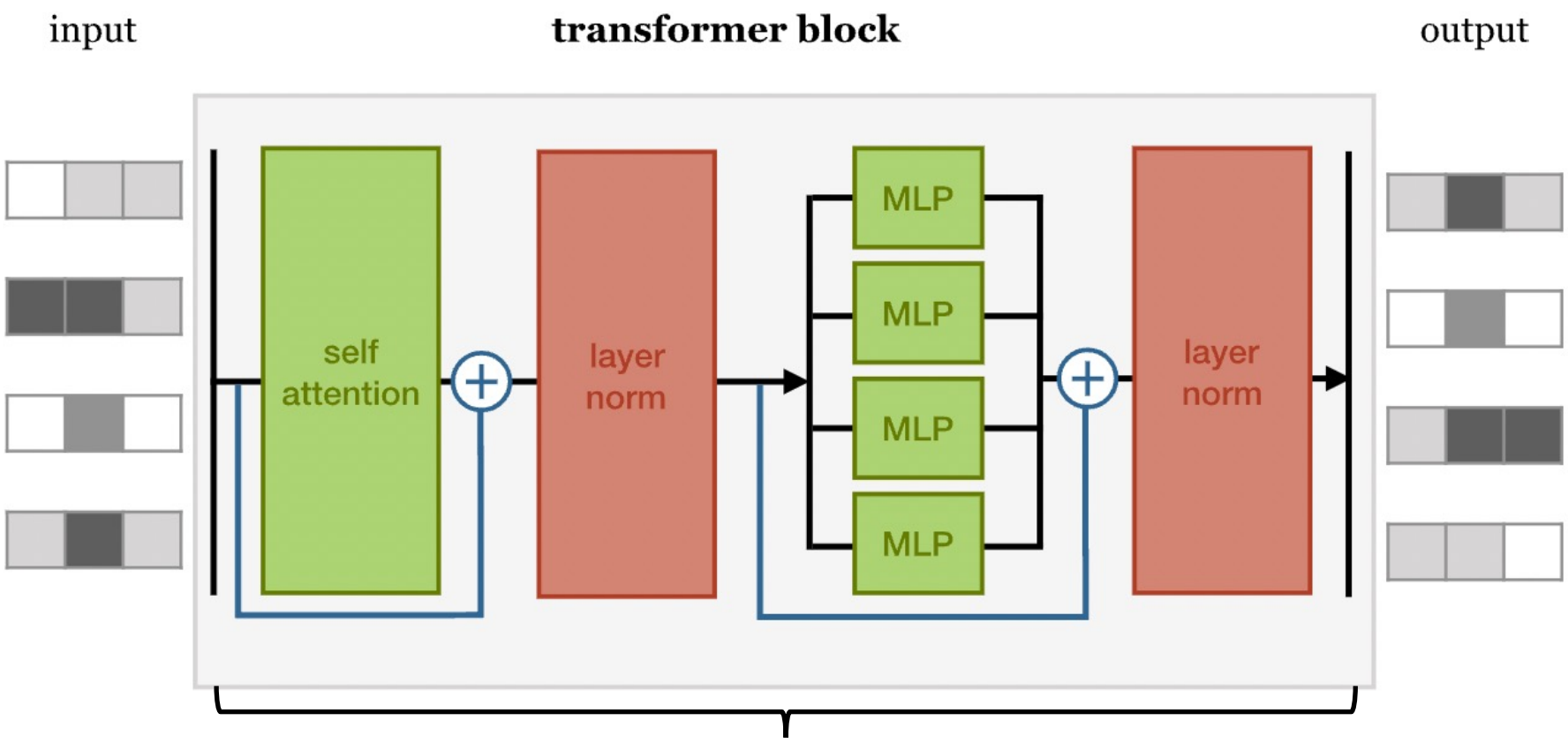
Typical Transformer Block



Where are non-linearities introduced in this block?

http://peterbloem.nl/blog/transformers

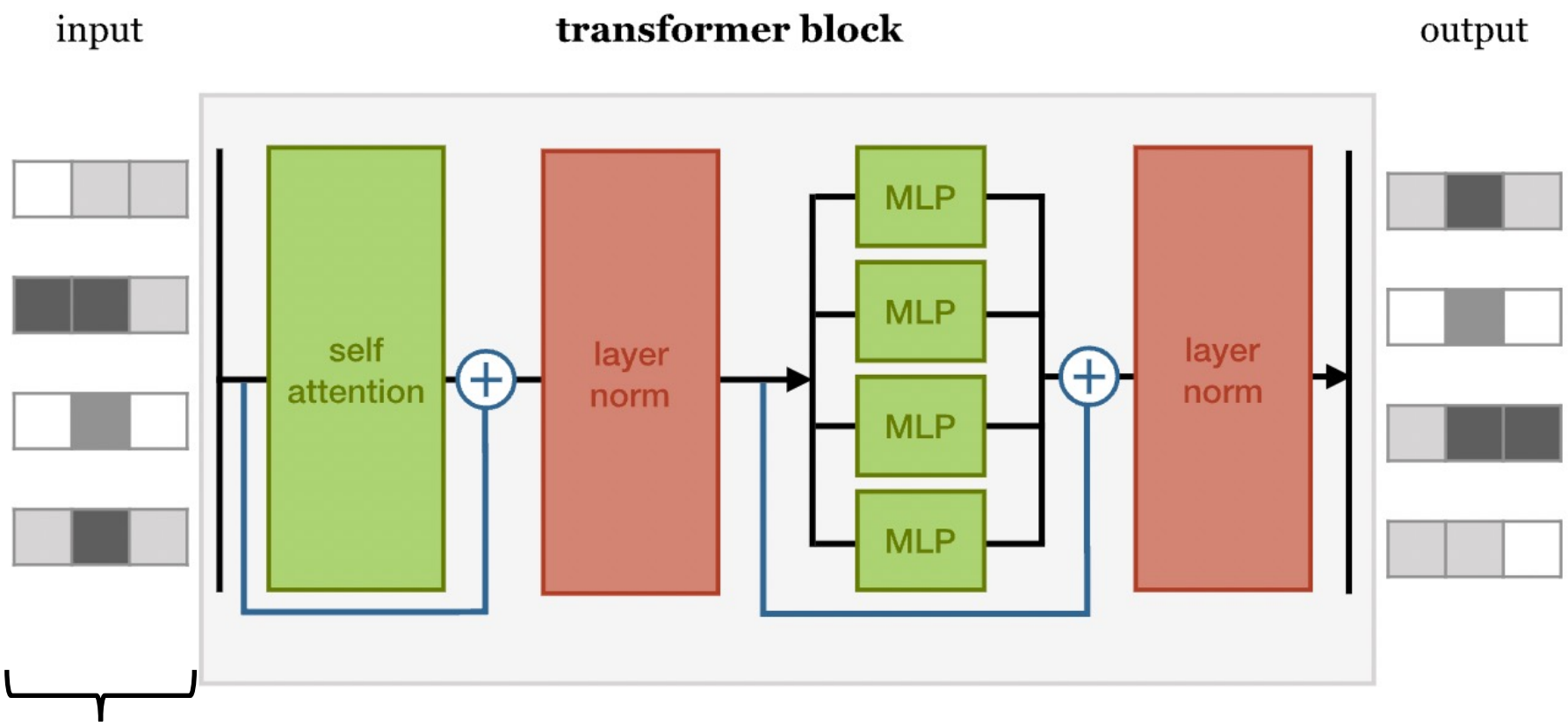
Typical Transformer Block



Non-linearities introduced in the softmax of self-attention, activation functions in MLP, and layer norms

http://peterbloem.nl/blog/transformers

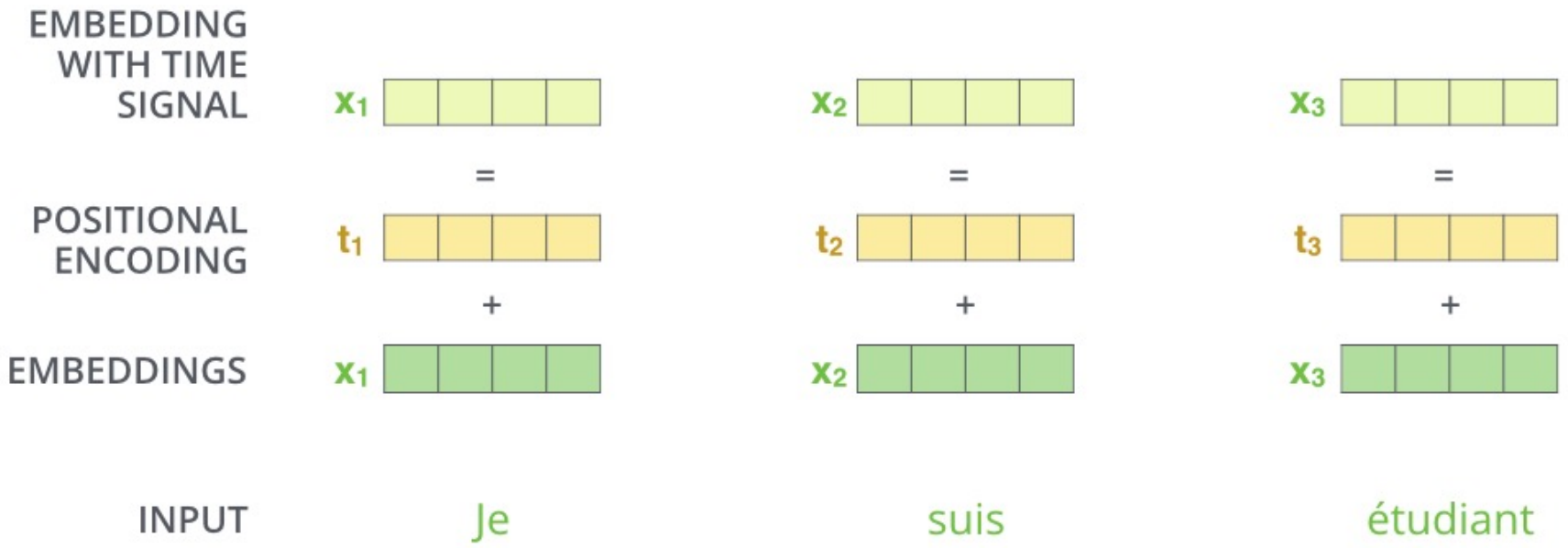
Challenge: Transformers Lack Sensitivity to the Order of the Input Tokens



Input observed as a *set* and so shuffling the order of input tokens results in the same outputs except in the same shuffled order (i.e. self-attention is *permutation equivariant*)

http://peterbloem.nl/blog/transformers

Solution: Add Position as Input to Transformer



- Options:
  - **Position embeddings**: created by training with sequences of every length during training
  - **Position encodings**: a function mapping positions to vectors that the network learns to interpret (enables generalization to lengths not observed during training)

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

Today's Topics

- Transformer overview
- Self-attention
- Multi-head attention
- Common transformer ingredients
- Pioneering transformer: machine translation

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

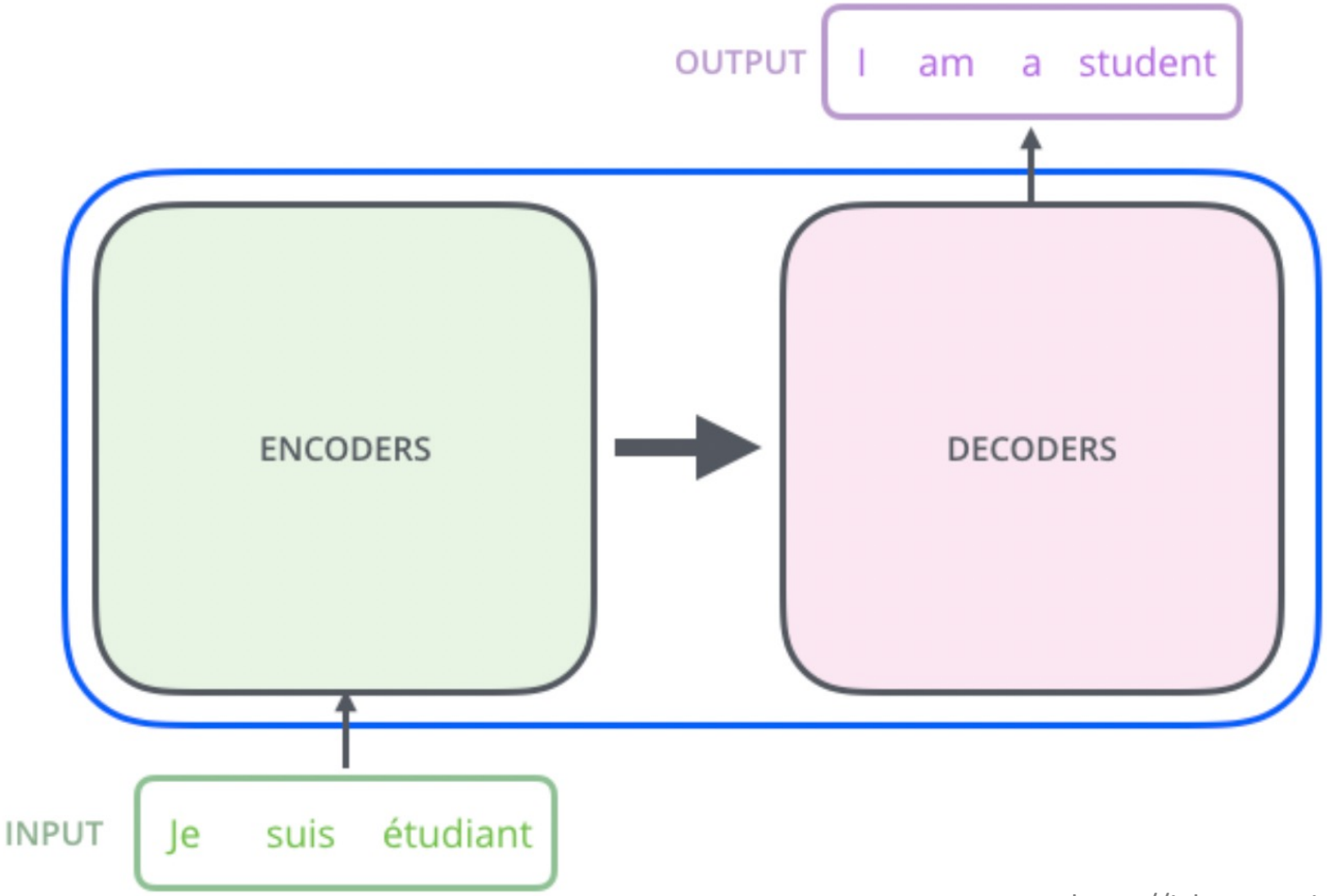
Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

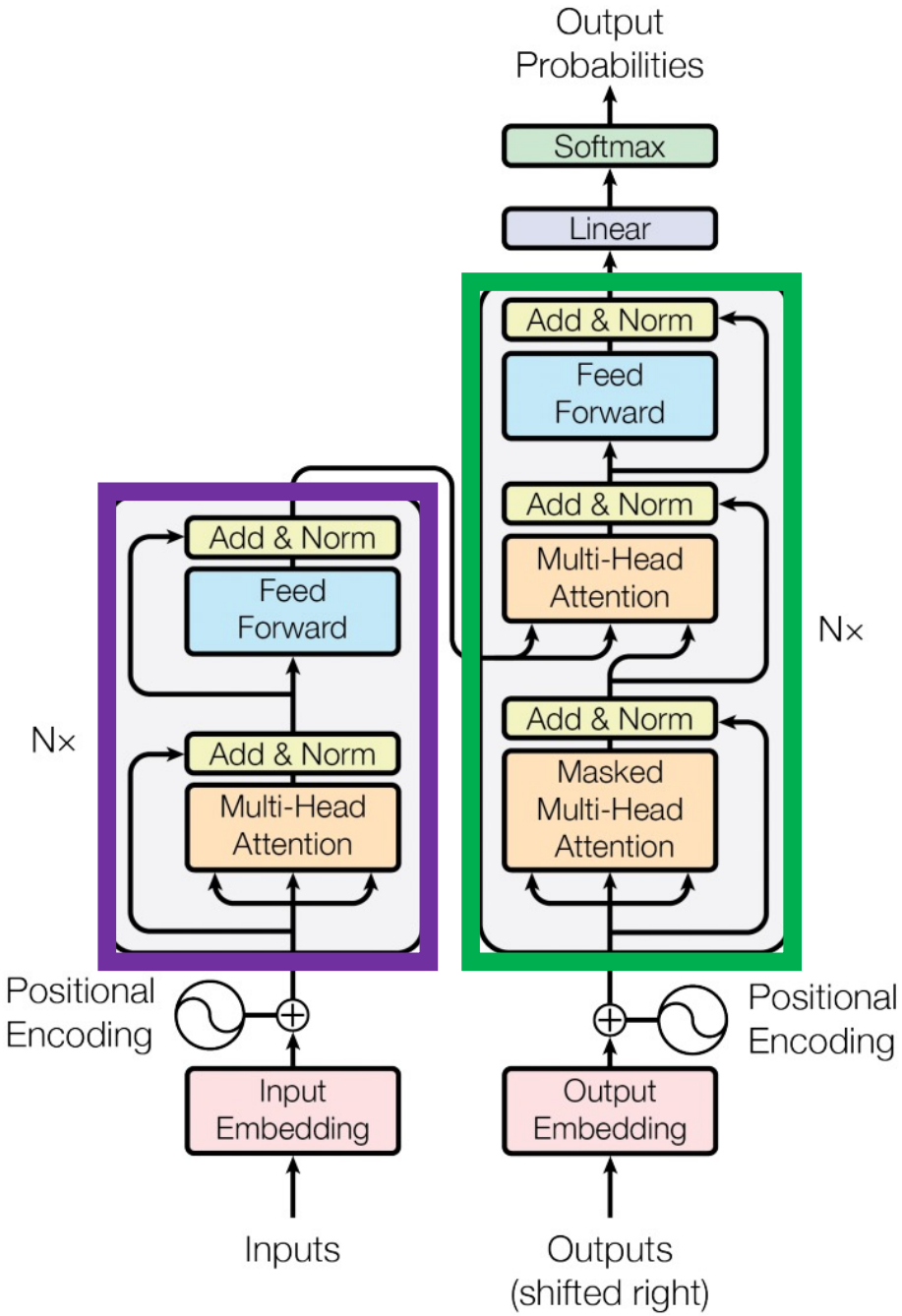
Target Application: Machine Translation



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

Architecture

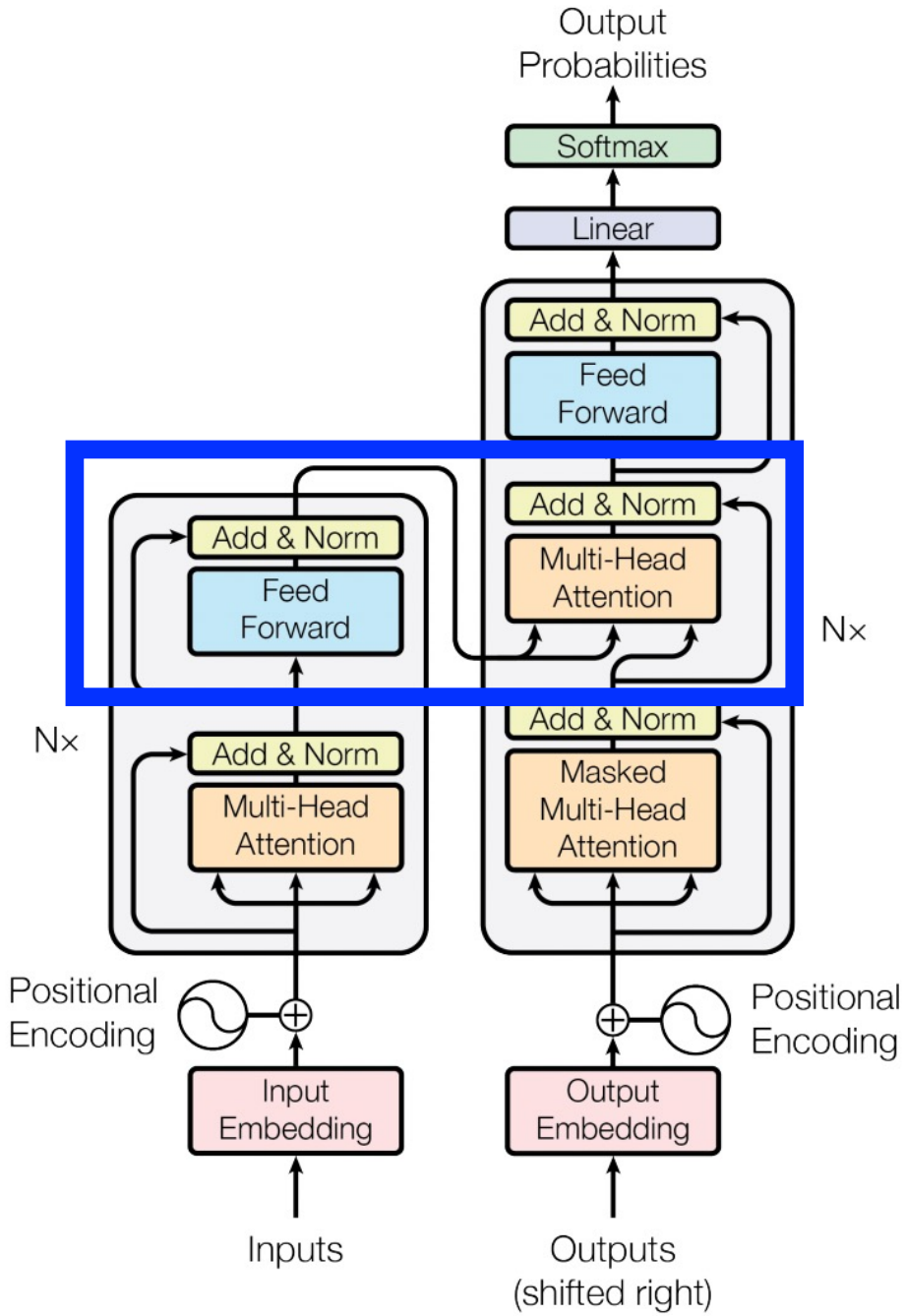
- Key Ingredient
  - Self-Attention in the encoder and decoder
- Other ingredients
  - Positional encoding
  - Layer normalization
  - Residual connections
  - Feed forward layers
- Nx = 6 chained blocks (encoder & decoder)



Vaswani et al. Attention Is All You Need. Neurips 2017.

Architecture

The decoder performs multi-head attention on the encoder output



Vaswani et al. Attention Is All You Need. Neurips 2017.