Lecture 1

CME 295: Transformers & Large Language Models

 **Afshine Amidi & Shervine Amidi** 

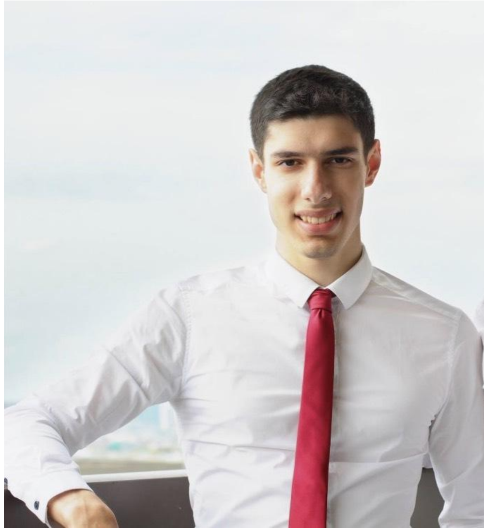
Stanford University

ICME

Teaching staff

Afshine and **Shervine**

Teaching staff



Afshine
Centrale Paris ('16), MIT ('17)
Uber, Google, Netflix

Shervine
Centrale Paris ('16), Stanford ('19)
Uber, Google, Netflix

Welcome to CME 295!

Goals.

1. Understand how **Transformers** work and how they **relate** to **LLMs**
2. Learn how **LLMs** are **trained** and used in various **applications**

Welcome to CME 295!

Goals.

1. Understand how **Transformers** work and how they **relate** to **LLMs**
2. Learn how **LLMs** are **trained** and used in various **applications**

Audience.

- Interested in LLMs
  - Career goal
  - Personal project
  - "AI literacy" or curiosity
- Prerequisite: machine learning basics, linear algebra

Logistics

Date & time.

- Fridays from 3:30pm to 5:20pm
- Thornton 110

Logistics

Date & time.

- Fridays from 3:30pm to 5:20pm
- Thornton 110

Details about the class.

- 2 units
- Letter or Credit/No credit
- Lectures are recorded
- Midterm (50% grade), scheduled on October 24th
- Final exam (50% grade), week of December 8th (exact date TBD)

Material

Class website. cme295.stanford.edu

- Contains syllabus & logistics
- Slides and recordings will be posted there

Material

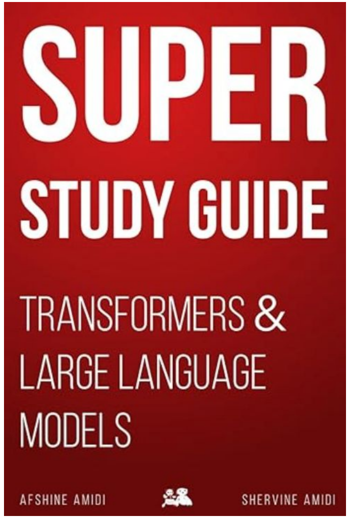
Class website. cme295.stanford.edu

- Contains syllabus & logistics
- Slides and recordings will be posted there

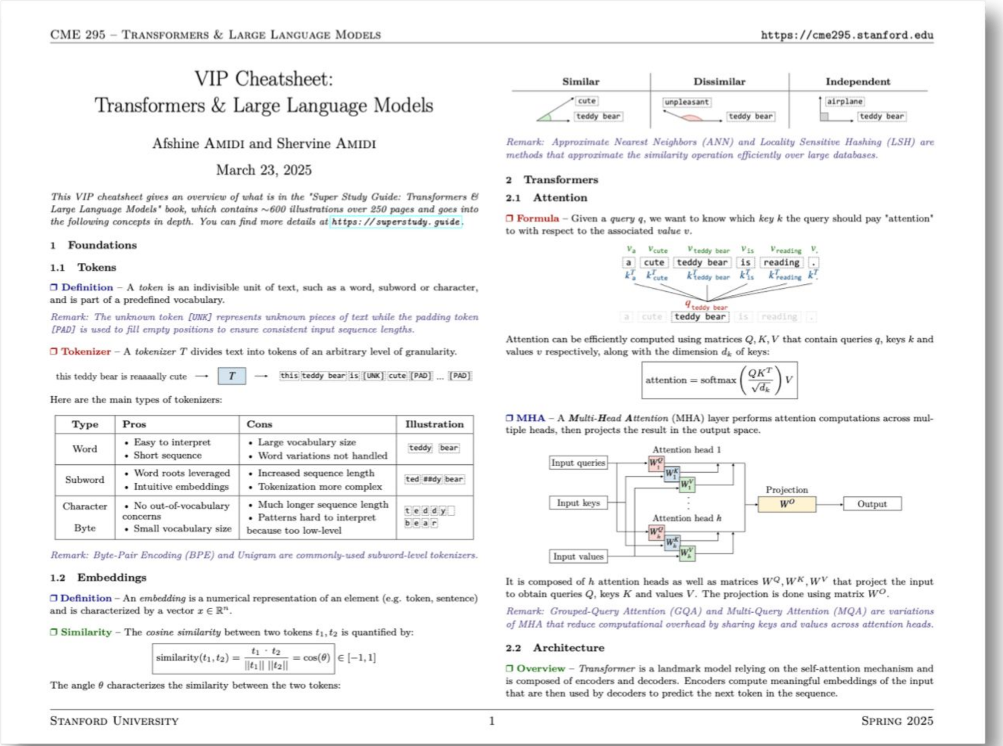
Class textbook.

Super Study Guide:
Transformers &
Large Language Models

Book: https://superstudy.guide



Material



CME 295 – Transformers & Large Language Models https://cme295.stanford.edu

VIP Cheatsheet:
Transformers & Large Language Models

Afshine Amidi and Shervine Amidi

March 23, 2025

This VIP cheatsheet gives an overview of what is in the "Super Study Guide: Transformers & Large Language Models" book, which contains ~600 illustrations over 250 pages and goes into the following concepts in depth. You can find more details at https://superstudy.guide.

1 Foundations

1.1 Tokens

❒ **Definition** – A *token* is an indivisible unit of text, such as a word, subword or character, and is part of a predefined vocabulary.

Remark: The unknown token [UNK] represents unknown pieces of text while the padding token [PAD] is used to fill empty positions to ensure consistent input sequence lengths.

❒ **Tokenizer** – A *tokenizer* T divides text into tokens of an arbitrary level of granularity.

this teddy bear is reaaaally cute ⟶ T ⟶ this | teddy | bear | is | [UNK] | cute | [PAD] | ... | [PAD]

Here are the main types of tokenizers:

Type	Pros	Cons	Illustration
Word	• Easy to interpret • Short sequence	• Large vocabulary size • Word variations not handled	teddy bear
Subword	• Word roots leveraged • Intuitive embeddings	• Increased sequence length • Tokenization more complex	ted ##dy bear
Character Byte	• No out-of-vocabulary concerns • Small vocabulary size	• Much longer sequence length • Patterns hard to interpret because too low-level	t e d d y b e a r

Remark: Byte-Pair Encoding (BPE) and Unigram are commonly-used subword-level tokenizers.

1.2 Embeddings

❒ **Definition** – An *embedding* is a numerical representation of an element (e.g. token, sentence) and is characterized by a vector $x \in \mathbb{R}^n$.

❒ **Similarity** – The *cosine similarity* between two tokens t_1, t_2 is quantified by:

$$\text{similarity}(t_1, t_2) = \frac{t_1 \cdot t_2}{\|t_1\|\,\|t_2\|} = \cos(\theta) \in [-1, 1]$$

The angle θ characterizes the similarity between the two tokens:

Similarity | Dissimilar | Independent

cute, teddy bear | unpleasant, teddy bear | airplane, teddy bear

Remark: Approximate Nearest Neighbors (ANN) and Locality Sensitive Hashing (LSH) are methods that approximate the similarity operation efficiently over large databases.

2 Transformers

2.1 Attention

❒ **Formula** – Given a *query* q, we want to know which *key* k the query should pay "attention" to with respect to the associated *value* v.

Attention can be efficiently computed using matrices Q, K, V that contain queries q, keys k and values v respectively, along with the dimension d_k of keys:

$$\text{attention} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

❒ **MHA** – A *Multi-Head Attention* (MHA) layer performs attention computations across multiple heads, then projects the result in the output space.

It is composed of h attention heads as well as matrices W^Q, W^K, W^V that project the input to obtain queries Q, keys K and values V. The projection is done using matrix W^O.

Remark: Grouped-Query Attention (GQA) and Multi-Query Attention (MQA) are variations of MHA that reduce computational overhead by sharing keys and values across attention heads.

2.2 Architecture

❒ **Overview** – *Transformer* is a landmark model relying on the self-attention mechanism and is composed of encoders and decoders. Encoders compute meaningful embeddings of the input that are then used by decoders to predict the next token in the sequence.

Stanford University 1 Spring 2025

VIP Cheatsheet

(also translated in 11 languages so far)

Link to PDF: https://github.com/afshinea/stanford-cme-295-transformers-large-language-models

Class communications

Canvas.

- Announcements
- Class discussions via Ed

Class communications

Canvas.

- Announcements
- Class discussions via Ed

Any other general **inquiries** / **questions**, email us:

- cme295-aut2526-staff@lists.stanford.edu
- afshine@stanford.edu, shervine@stanford.edu

Example of a slide in this class

Explanation of a CME 295 concept

Source & suggested reading, if interested

Some paper

Disclaimer before starting: many abbreviations....

BERT SFT BLEU FLAN BPE WNLI MLM

MRPC

GPT LaaJ LSTM PoS T5

QA

ROUGE GLUE PEFT PPO NER

F1 RAG PPL LLaMA C4 MT RLHF

GRU SQuAD WP DPO SP NLG METEOR

...but don't worry!

BERT, T5, GPT, LLaMA

Transformer-based models

LSTM, GRU, GloVe, BPE, CoT, ToT, SC, RAG

Misc architectures & techniques

SFT, PEFT, FLAN, RL, RM, RLHF, PPO, DPO

Training strategies

NER, PoS, MLM, NSP, MT, QA, NLG

Tasks

MNLI, WNLI, C4, SQuAD, GLUE, MRPC

Datasets

F1, PPL, ROUGE, BLEU, METEOR, LaaJ, WER

Metrics

CME 295
Transformers & Large Language Models

NLP overview

Tokenization

Word representation

RNNs

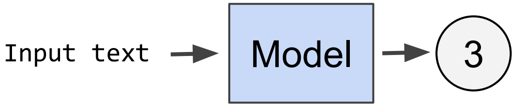
Self-attention mechanism

Transformer architecture

End-to-end example

NLP tasks overview

Classification



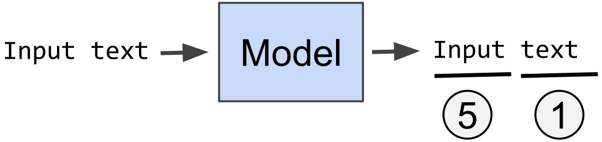
- Sentiment extraction
- Intent detection
- Language detection
- Topic modeling

NLP tasks overview

Classification

- Sentiment extraction
- Intent detection
- Language detection
- Topic modeling

“Multi”-classification



- Part of speech tagging
- Named entity recognition
- Dependency parsing
- Constituency parsing

NLP tasks overview

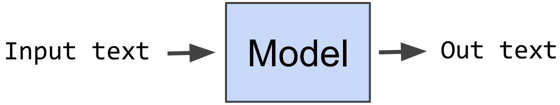
Classification

- Sentiment extraction
- Intent detection
- Language detection
- Topic modeling

“Multi”-classification

- Part of speech tagging
- Named entity recognition
- Dependency parsing
- Constituency parsing

Generation



- Machine translation
- Question answering
- Summarization
- Text generation

NLP task: Sentiment Extraction

NLP task: Sentiment Extraction

`This teddy bear is SO CUTE!` → Model → +

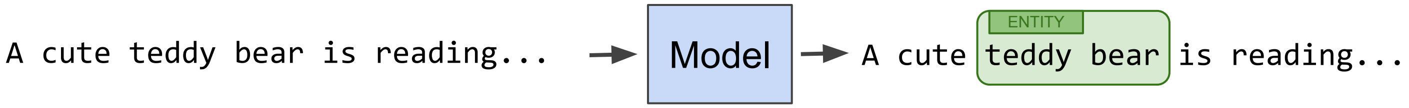
Datasets

Amazon reviews | 9.3 IMDB critiques | Twitter

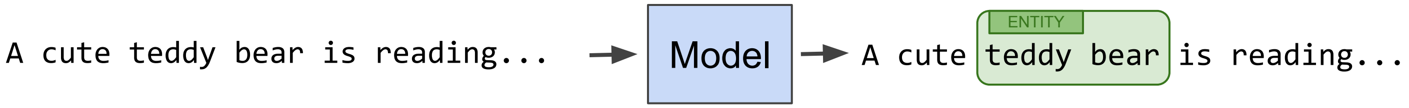
Evaluation metrics

- Accuracy → % of observations that were correctly predicted?
- Precision → % of predicted positive that were correct?
- Recall → % of actually positive that were correct?
- F1 score → score that is a function of precision and recall

NLP task: Named Entity Recognition



NLP task: Named Entity Recognition



Datasets

Annotated Reuters newspaper (CoNLL-2003, CoNLL++)

Evaluation metrics

- Accuracy
- Precision
- Recall
- F1 score

at a token level, per entity type

NLP task: Machine Translation

`A cute teddy bear is reading` → Model → `Un ours en peluche mignon lit`

NLP task: Machine Translation

`A cute teddy bear is reading` → **Model** → `Un ours en peluche mignon lit`

Datasets

WMT'14 English-French

WMT'14 English-German

Evaluation metrics

- BLEU → quality of text translated, similar to "precision"
- ROUGE → quality of text generated, similar to "recall"
- Perplexity → quantifies how 'surprised' the model is to see some words together

Formulas can be found on this website: http://www.statmt.org/book/slides/08-evaluation.pdf

High-level timeline

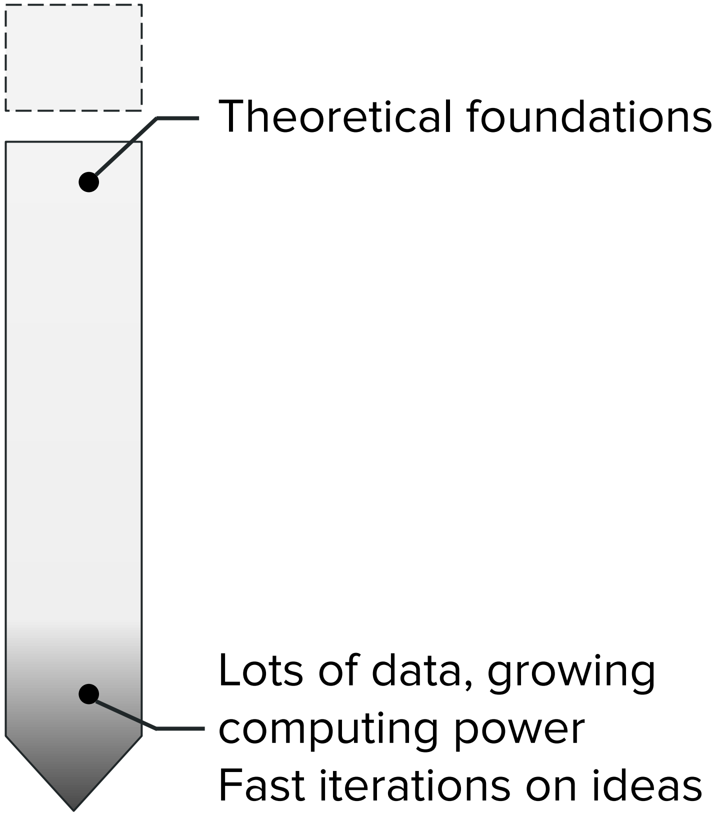
1980s Recurrent neural networks (RNNs)

1997 Long short-term memory (LSTM)

2013 Word2vec

2017 Transformers

2020s Large Language Models



CME 295
Transformers & Large Language Models

Tokenization

```
A cute teddy bear is reading.
```

Tokenization

`A cute teddy bear is reading.`

arbitrary	`A`	`cute`	`teddy bear`	`is`	`reading`	`.`

Tokenization

A cute teddy bear is reading.

	A	cute	teddy bear		is	reading	.
word	A	cute	teddy	bear	is	reading	.

Tokenization

`A cute teddy bear is reading.`

A	cute	teddy bear	is	reading	.

A	cute	teddy	bear	is	reading	.

sub-word

A	cute	ted	##dy	bear	is	read	##ing	.

Tokenization

A cute teddy bear is reading.

A	cute	teddy bear	is	reading	.

A	cute	teddy	bear	is	reading	.

A	cute	ted	##dy	bear	is	read	##ing	.

A	_	c	u	t	e	_	t	e	d	d	y	_	b	e	a	r	_	i	s	_	r	e	a	d	i	n	g	.

Tokenization summary

Method	Pros	Cons
Word-level	• Simple • Interpretable	• Risk of OOV • Does not leverage knowledge of root
Subword-level e.g. WordPiece, BPE	• Leverages common prefixes and suffixes • Learned from the data	• Risk of OOV, though less than word-level
Character-level	• Small chance of OOV • RoBUsT tO CASinG anD MIspeliNGs	• Makes computations slower • Embeddings not interpretable

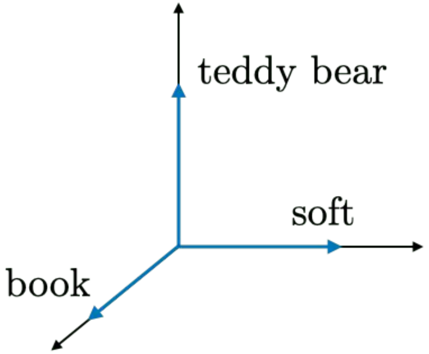
CME 295
Transformers & Large Language Models

- NLP overview
- Tokenization
- **Word representation**
- RNNs
- Self-attention mechanism
- Transformer architecture
- End-to-end example

Token representations

Motivation

Naive (one-hot) encoding



$$\text{soft} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \begin{aligned} \langle \text{teddy bear}, \ \text{book} \rangle &= 0 \\ \langle \text{teddy bear}, \ \text{soft} \rangle &= 0 \end{aligned}$$

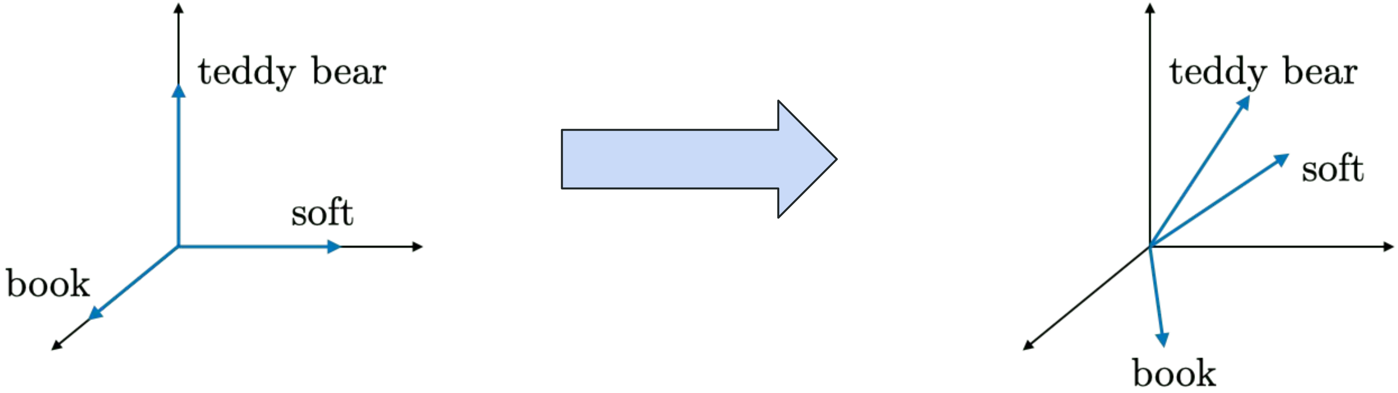
Figure adapted from "VIP cheatsheets for Stanford's CS 230", Amidi. stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Token representations

Motivation

Naive (one-hot) encoding

Learned embedding



$$\text{soft} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \begin{aligned} \langle \text{teddy bear},\ \text{book} \rangle &= 0 \\ \langle \text{teddy bear},\ \text{soft} \rangle &= 0 \end{aligned}$$

$$\text{soft} = \begin{pmatrix} 0.95 \\ 0.32 \\ 0.01 \end{pmatrix} \qquad \begin{aligned} \langle \text{teddy bear},\ \text{book} \rangle &\sim 0 \\ \langle \text{teddy bear},\ \text{soft} \rangle &\sim 1 \end{aligned}$$

Figure adapted from "VIP cheatsheets for Stanford's CS 230", Amidi. stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Word2vec

Overview

- Neural network with a **proxy task** over billions of words worth of text
- Learns an embedding layer

Proxy tasks

- CBOW (continuous bag of words)

...A cute teddy bear is reading...

- Skip-gram

...A cute teddy bear is reading...

Word2vec paper "Efficient Estimation of Word Representations in Vector Space", Mikolov et al., 2013.

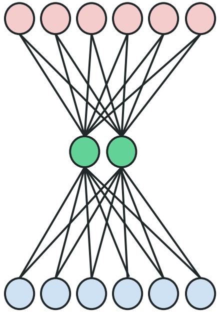
Word2vec

Architecture

output size V

hidden size d

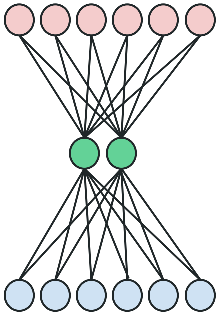
input size V



Word2vec

Example with predicting next word

A cute teddy bear is reading

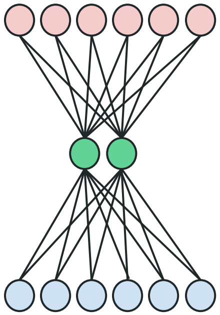


A cute teddy bear is reading

Word2vec

Example with predicting next word

A cute teddy bear is reading



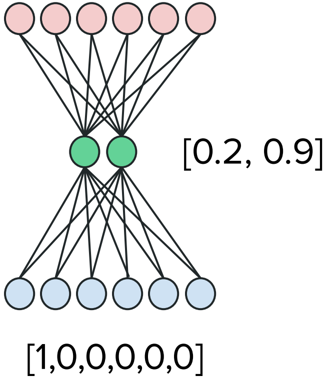
[1,0,0,0,0,0]

A cute teddy bear is reading

Word2vec

Example with predicting next word

A cute teddy bear is reading

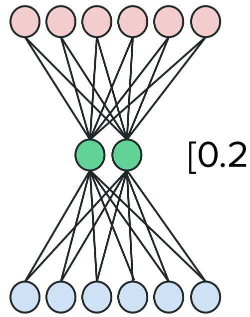


A cute teddy bear is reading

Word2vec

Example with predicting next word

A cute teddy bear is reading

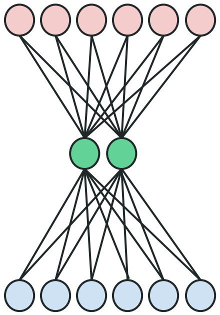


A cute teddy bear is reading

Word2vec

Example with predicting next word

A cute $\boxed{\text{teddy bear}}$ is reading

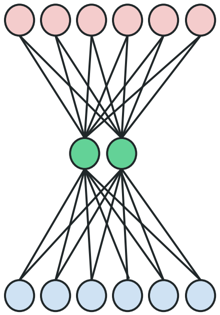


A $\boxed{\text{cute}}$ teddy bear is reading

Word2vec

Example with predicting next word

A cute teddy bear is reading



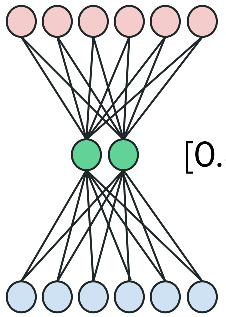
[0,1,0,0,0,0]

A cute teddy bear is reading

Word2vec

Example with predicting next word

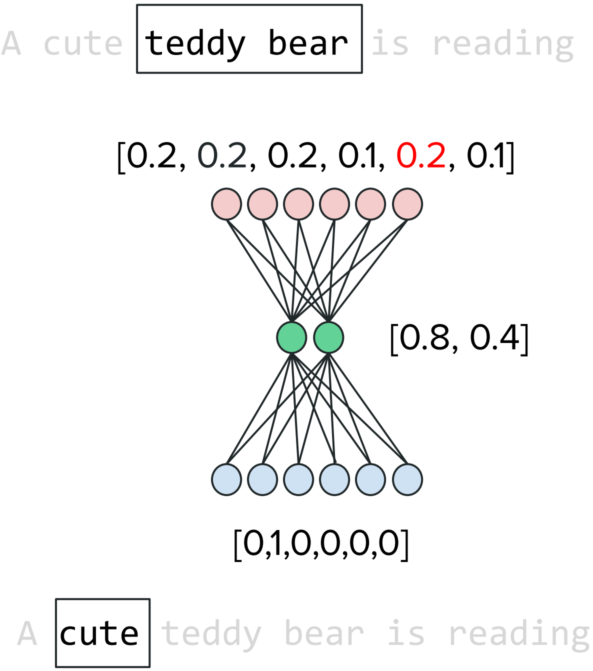
A cute teddy bear is reading



A cute teddy bear is reading

Word2vec

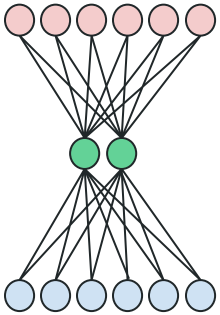
Example with predicting next word



Word2vec

Example with predicting next word

A cute teddy bear $\boxed{\text{is}}$ reading

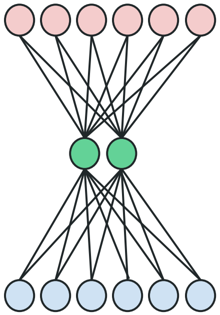


A cute $\boxed{\text{teddy bear}}$ is reading

Word2vec

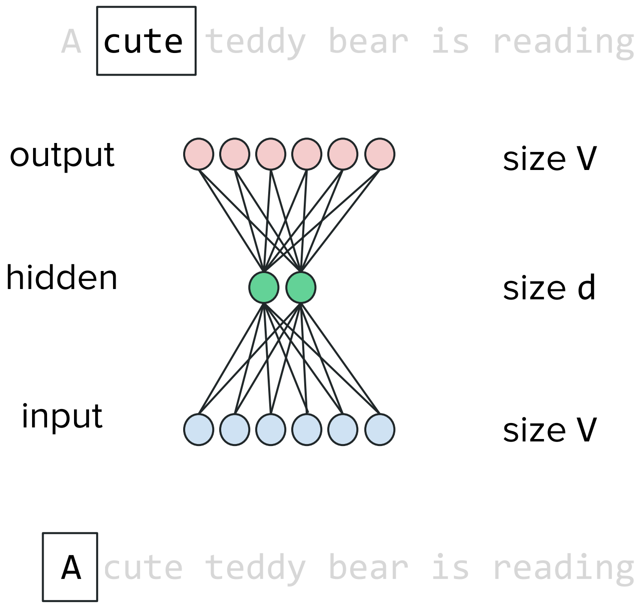
Example with predicting next word

A cute teddy bear is $\boxed{\text{reading}}$



A cute teddy bear $\boxed{\text{is}}$ reading

Word2vec



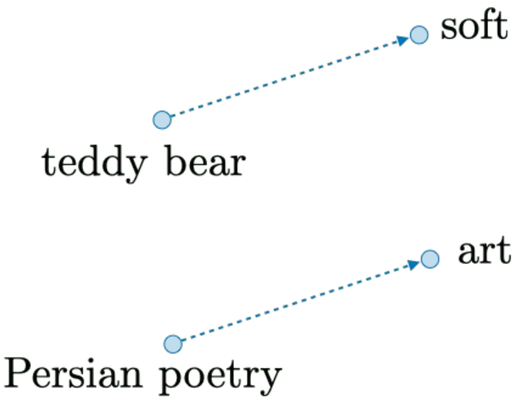


Figure adapted from "VIP cheatsheets for Stanford's CS 230", Amidi. *stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks*

CME 295
Transformers & Large Language Models

NLP overview

Tokenization

Word representation

RNNs

Self-attention mechanism

Transformer architecture

End-to-end example

Recurrent Neural Networks (RNNs)

Overview

- First introduced in the 80s
- Class of neural networks where connections form a temporal sequence

General form

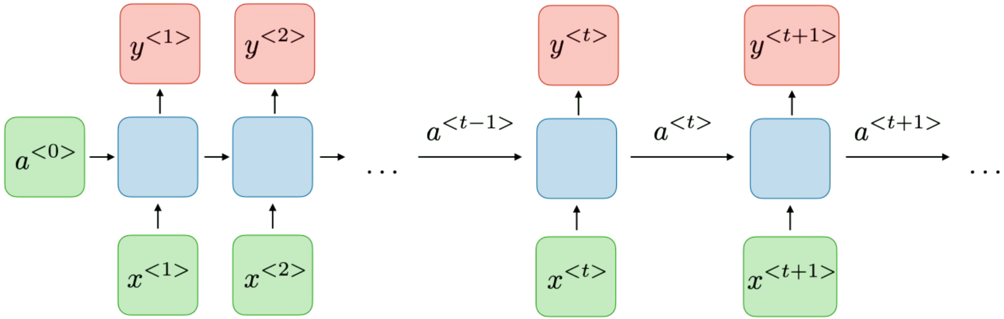
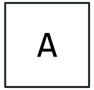
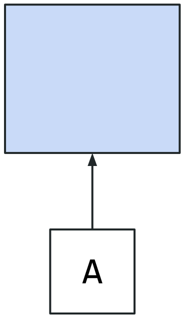


Figure adapted from "VIP cheatsheets for Stanford's CS 230", Amidi. stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

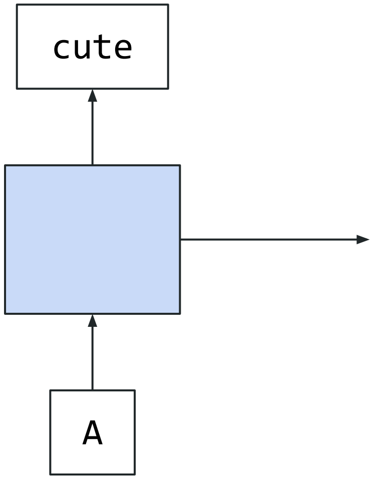
Recurrent Neural Networks (RNNs)



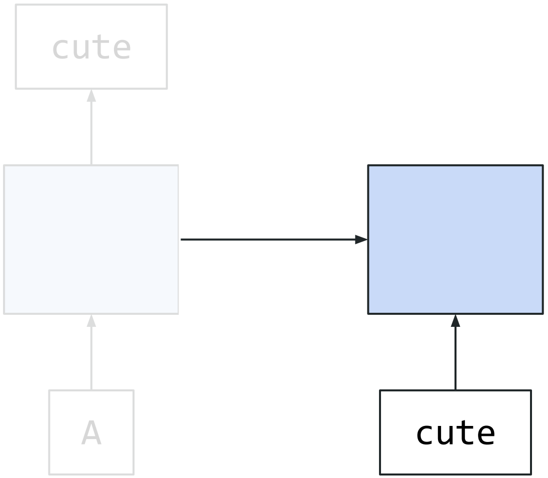
Recurrent Neural Networks (RNNs)



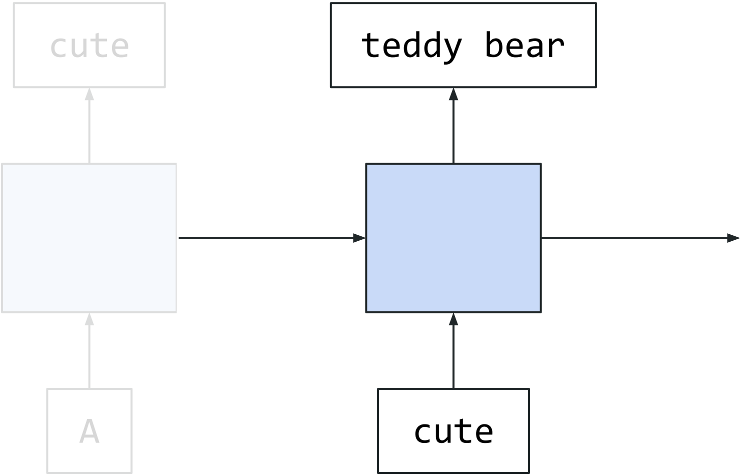
Recurrent Neural Networks (RNNs)



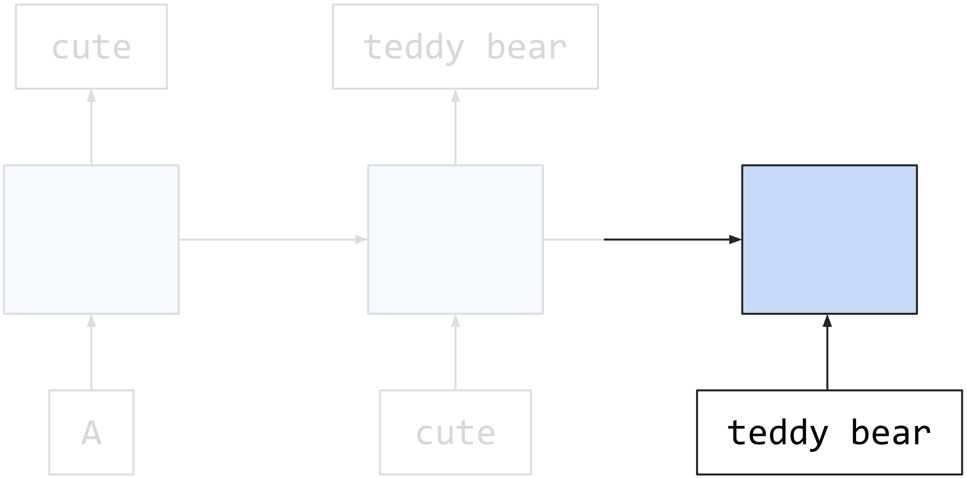
Recurrent Neural Networks (RNNs)



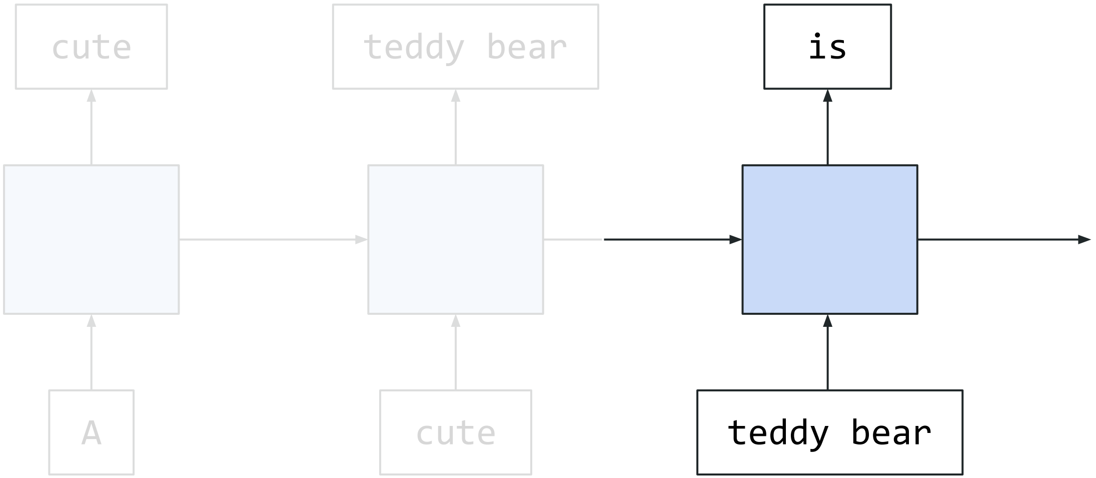
Recurrent Neural Networks (RNNs)



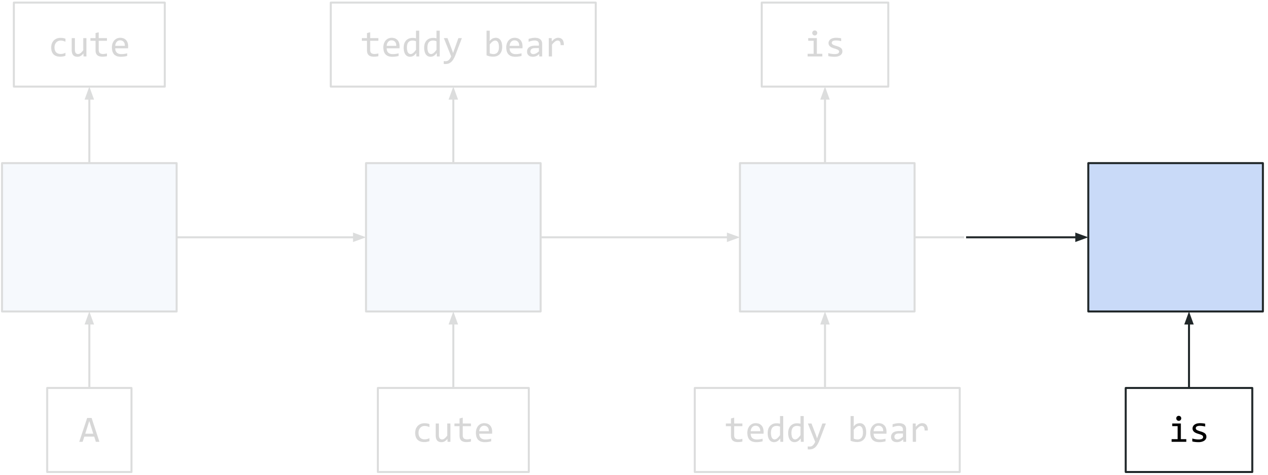
Recurrent Neural Networks (RNNs)



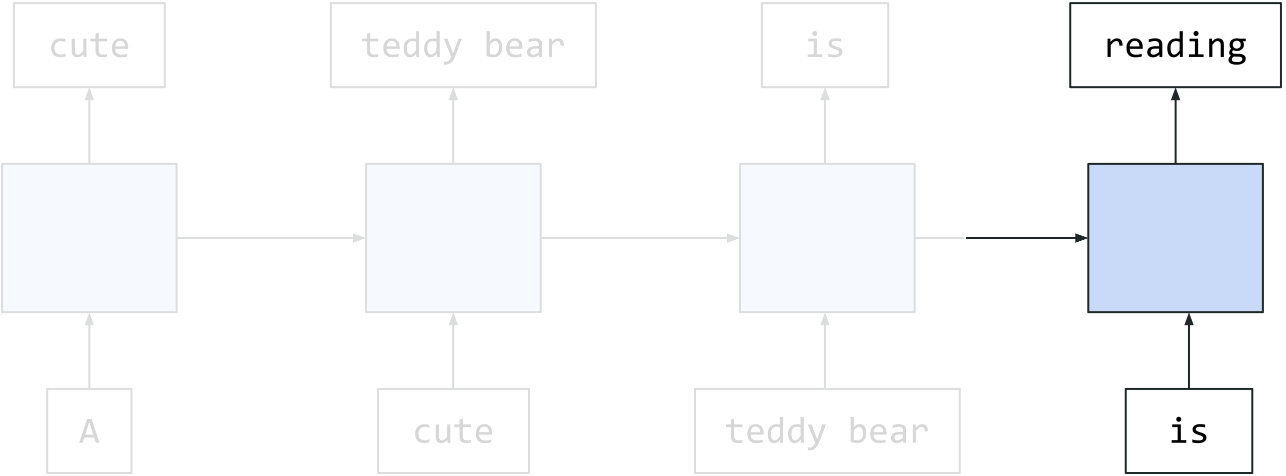
Recurrent Neural Networks (RNNs)



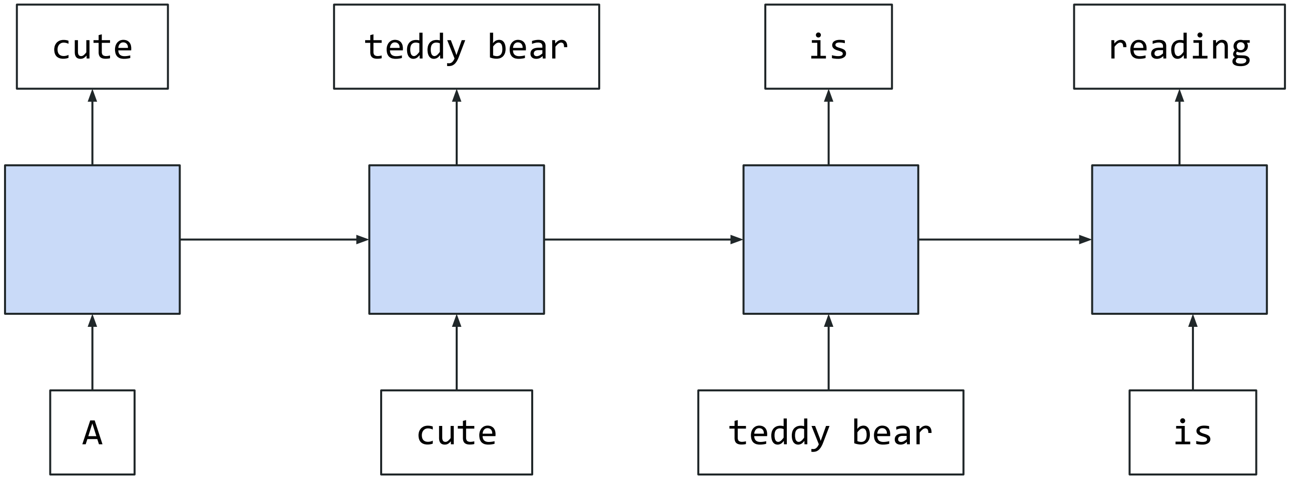
Recurrent Neural Networks (RNNs)



Recurrent Neural Networks (RNNs)

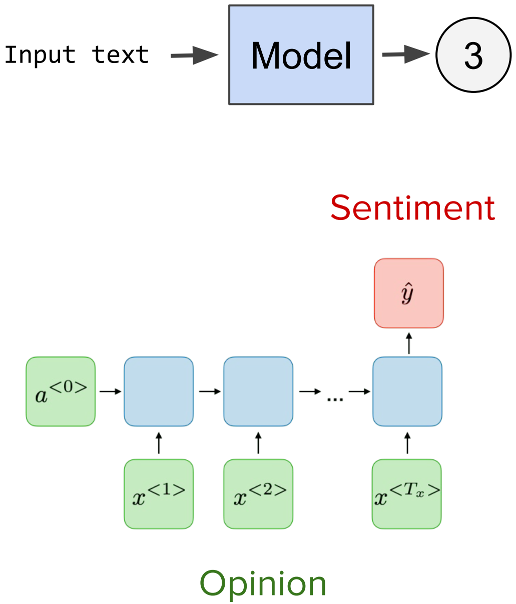


Recurrent Neural Networks (RNNs)

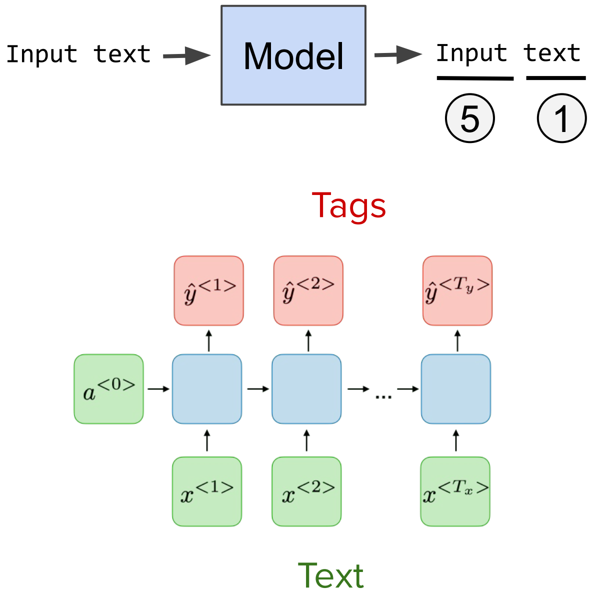


Recurrent Neural Networks (RNNs)

Classification



"Multi"-classification



Generation

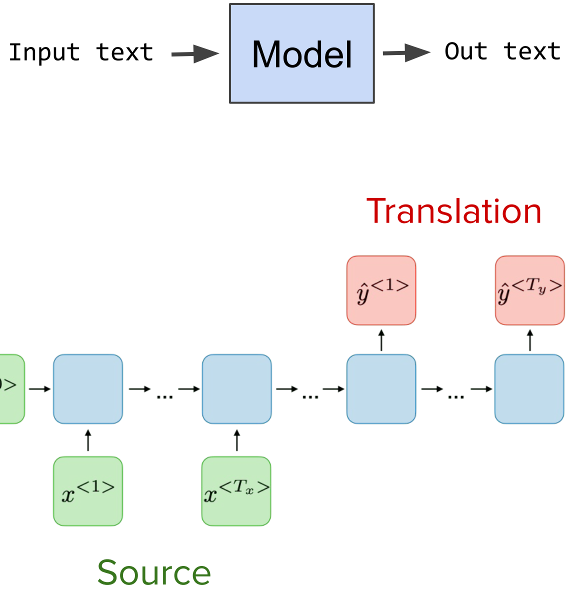


Figure adapted from "VIP cheatsheets for Stanford's CS 230", Amidi. stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

Long Short-Term Memory (LSTM)

Overview

- Introduced in "Long short-term memory" (1997)
- Uses a more structured approach in the cell's hidden state

General form

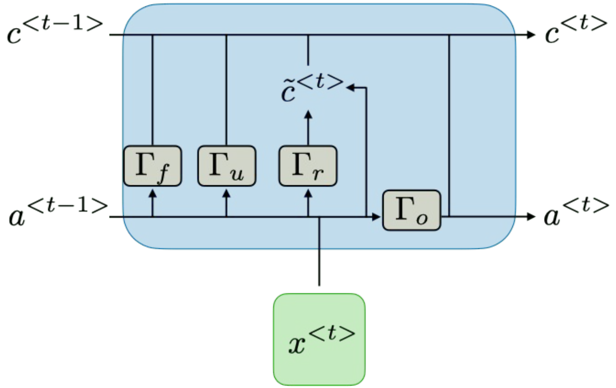


Figure adapted from "VIP cheatsheets for Stanford's CS 230", Amidi. *stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks*

Summary of main methods (non-exhaustive list)

Method	Pros	Cons
Word2vec e.g. CBOW, Skip-gram	• Very simple, yet powerful • Intuitive embeddings	• Word order does not count • Embeddings not context aware
Recurrent Neural Networks e.g. traditional RNN, LSTM	• Word order matters • State-of-the-art results	• Vanishing gradient problem • Slow computations

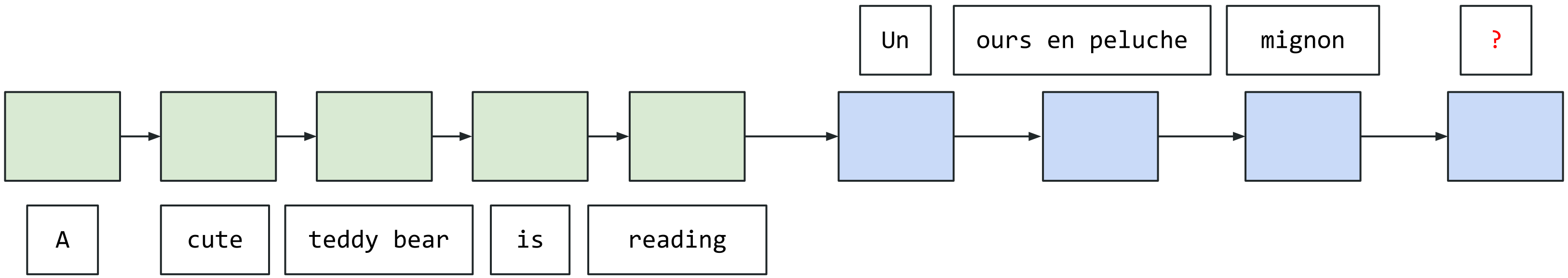
History of attention

- Introduced in 2014
- Translation tasks had a real issue with long-term dependencies
- Seq2seq unable to "remember" what input sentence was saying

"*Neural Machine Translation by Jointly Learning to Align and Translate*", *Bahdanau et al., 2014.*

History of attention

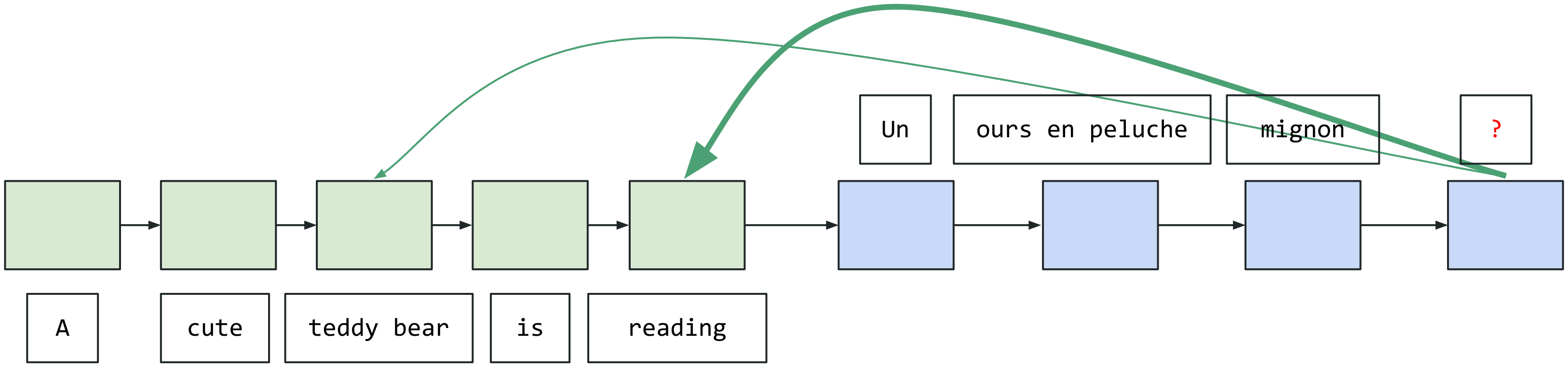
- Introduced in 2014
- Translation tasks had a real issue with long-term dependencies
- Seq2seq unable to "remember" what input sentence was saying



"*Neural Machine Translation by Jointly Learning to Align and Translate*", *Bahdanau et al., 2014.*

History of attention

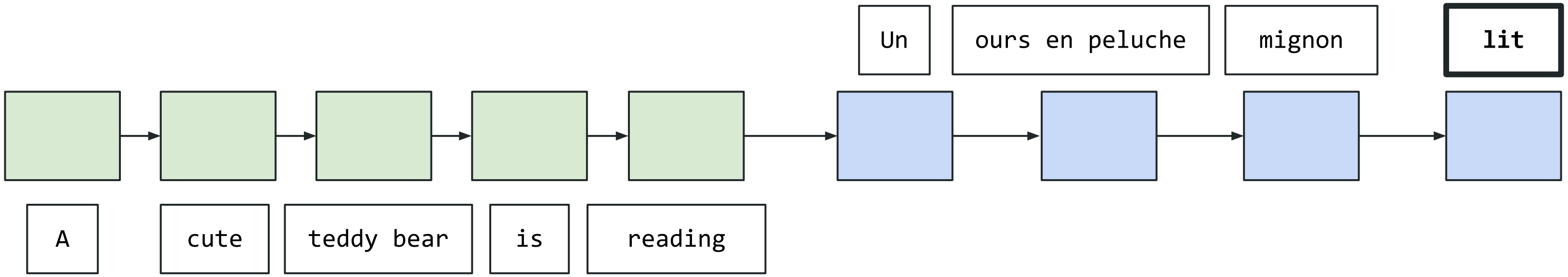
- Introduced in 2014
- Translation tasks had a real issue with long-term dependencies
- Seq2seq unable to "remember" what input sentence was saying



"Neural Machine Translation by Jointly Learning to Align and Translate", Bahdanau et al., 2014.

History of attention

- Introduced in 2014
- Translation tasks had a real issue with long-term dependencies
- Seq2seq unable to "remember" what input sentence was saying



"*Neural Machine Translation by Jointly Learning to Align and Translate*", *Bahdanau et al., 2014.*

Transformers & Large Language Models

NLP overview

Tokenization

Word representation

RNNs

Self-attention mechanism

Transformer architecture

End-to-end example

Overview of the Transformer

- Introduced in the 2017 paper "Attention is All You Need"
- Relies on the **self-attention** mechanism
- State-of-the-art results on machine translation tasks

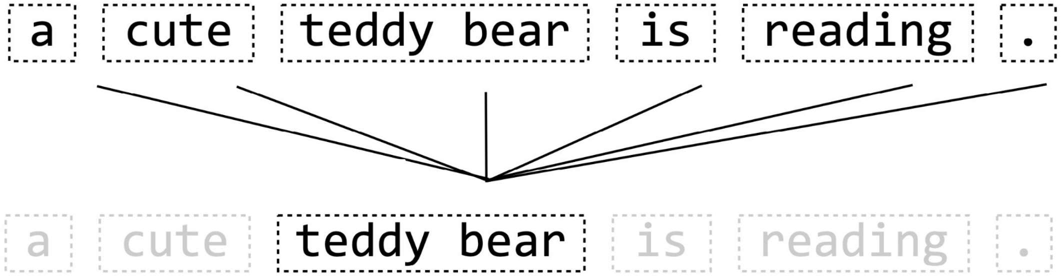
Overview of the Transformer

- Introduced in the 2017 paper "Attention is All You Need"
- Relies on the **self-attention** mechanism
- State-of-the-art results on machine translation tasks

a | cute | teddy bear | is | reading | .

Overview of the Transformer

- Introduced in the 2017 paper "Attention is All You Need"
- Relies on the **self-attention** mechanism
- State-of-the-art results on machine translation tasks



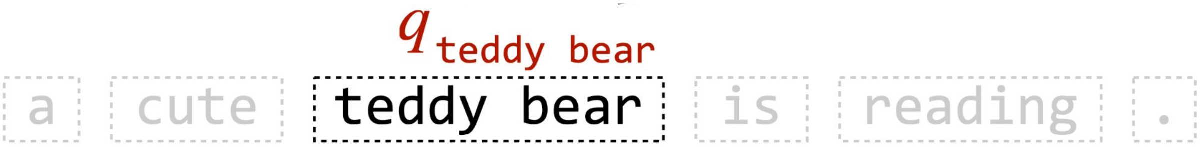
Attention mechanism

Concept of **Q**uery, **K**ey, **V**alue

a cute teddy bear is reading .

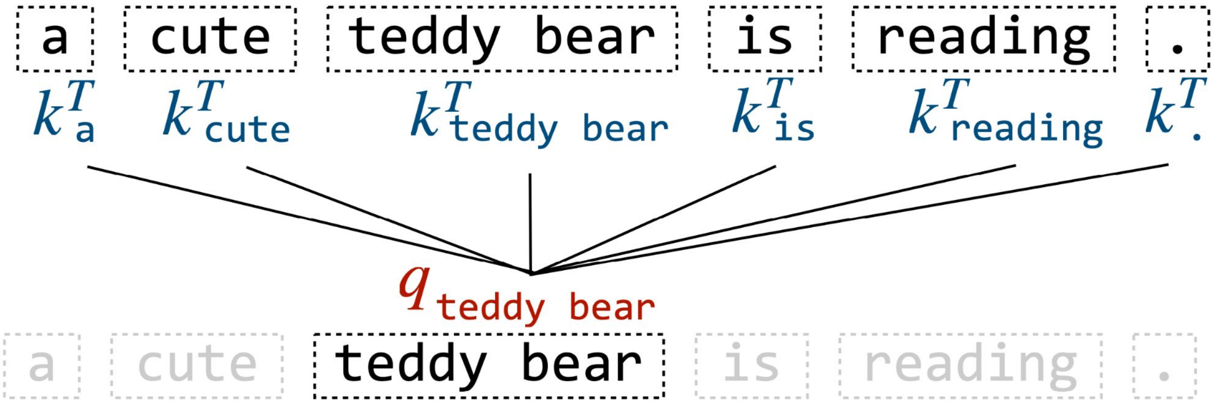
Attention mechanism

Concept of **Q**uery, **K**ey, **V**alue



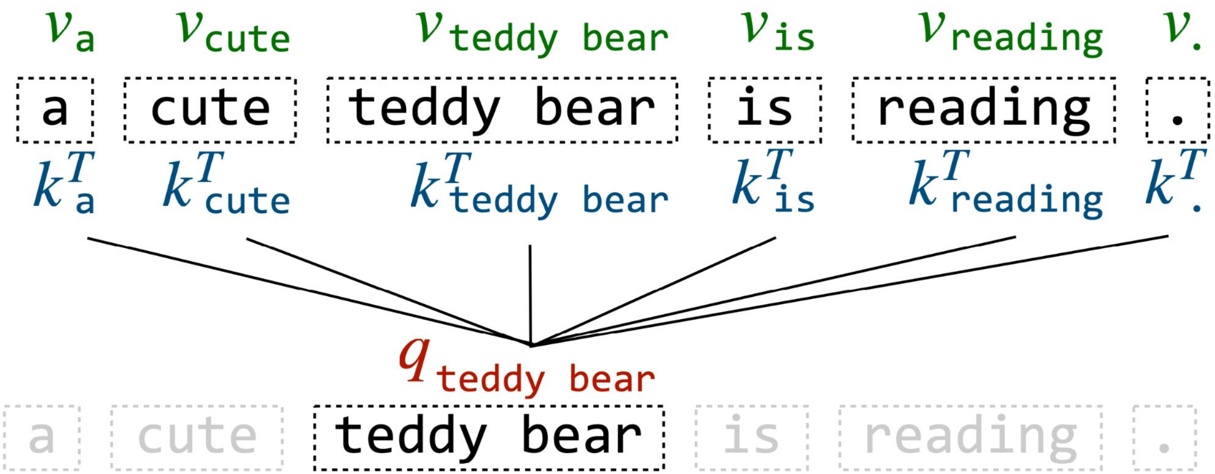
Attention mechanism

Concept of **Q**uery, **K**ey, **V**alue



Attention mechanism

Concept of **Q**uery, **K**ey, **V**alue



Attention mechanism

Efficient computations with matrices:

$$\boxed{\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V}$$

Attention mechanism

Efficient computations with matrices:

$$\boxed{\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V}$$

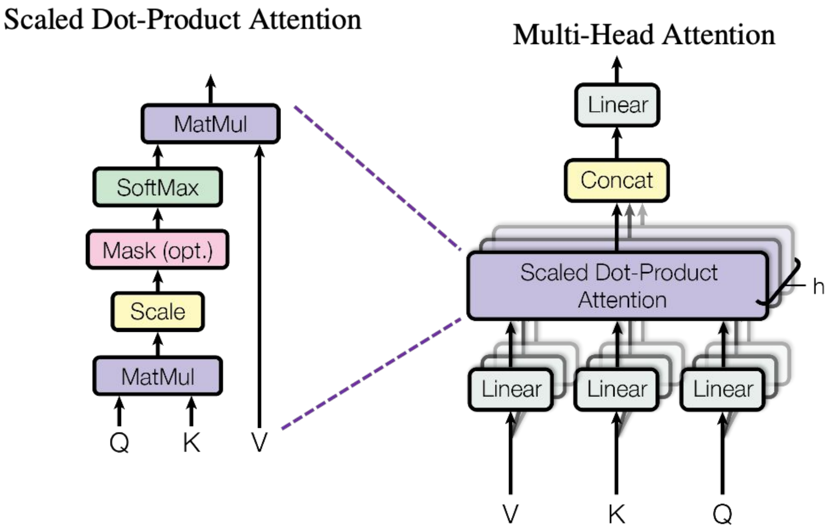
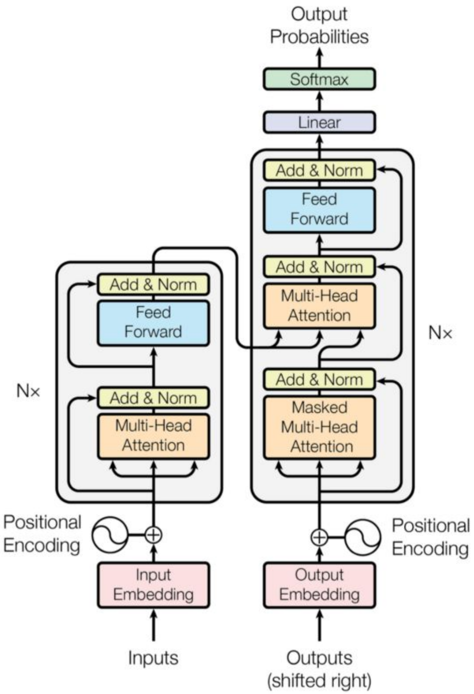


Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Transformers & Large Language Models

Transformer architecture



Attention layer (MHA)

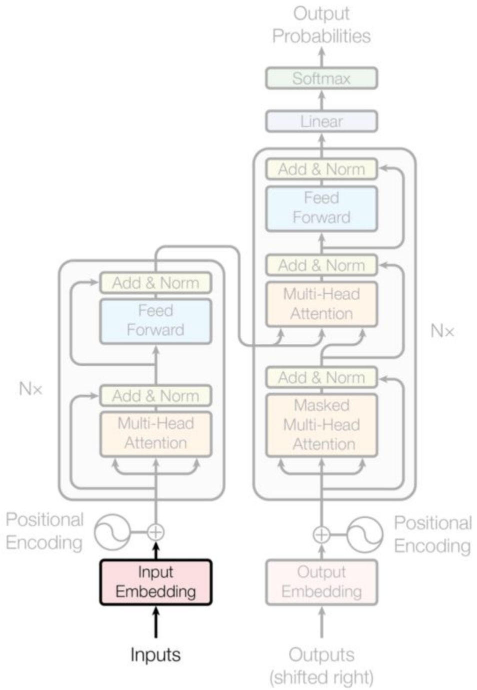
- Self-attention (Encoder-Encoder, Decoder-Decoder)
- Encoder-Decoder attention layer

Feed Forward Neural Network (FFNN)

Positional Encoding (PE)

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Input



Overview

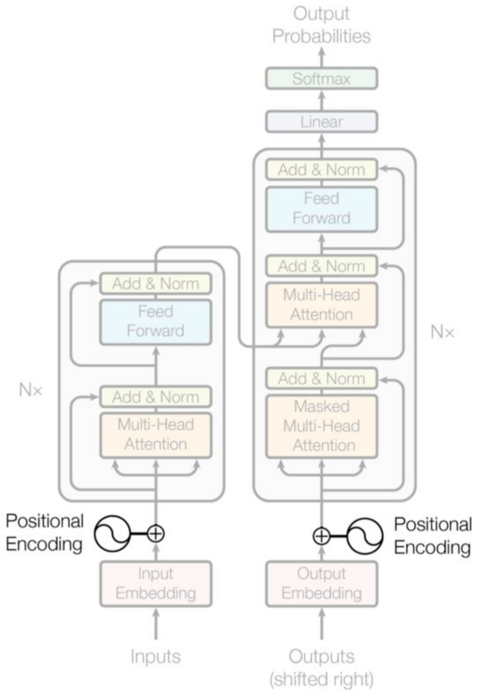
- Text is "tokenized"
- Learned embeddings for tokens

Parameters

- V: vocabulary size
- `d_model`: embedding dimensions

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

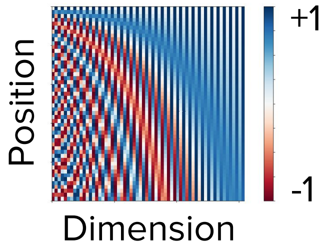
... with a trick!

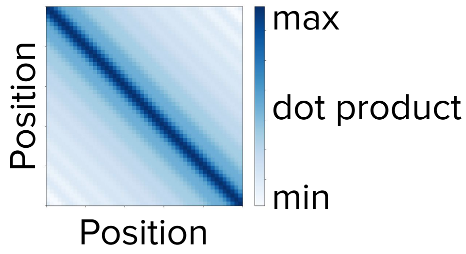


Positional encoding

Idea:

- Add **position information** to inputs
- Can be either learned or hardcoded

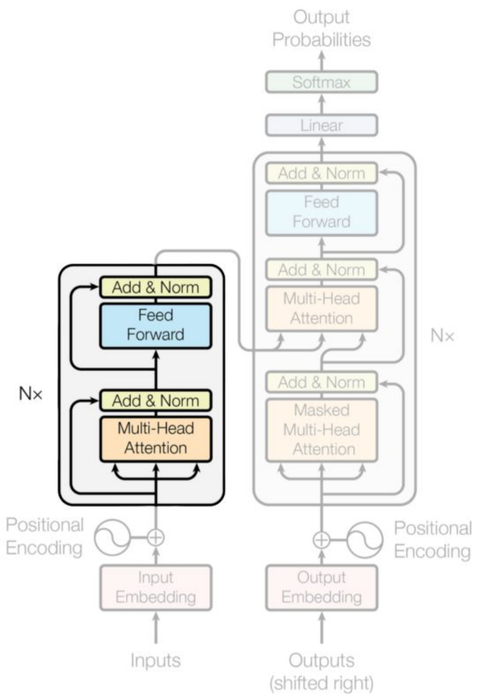




Goal: let model understand relative input position

Left figure adapted from "Attention Is All You Need", Vaswani et al., 2017.
Right figures adapted from "Transformer Architecture: The Positional Encoding", Kazemnejad, 2019.

Encoder



Overview

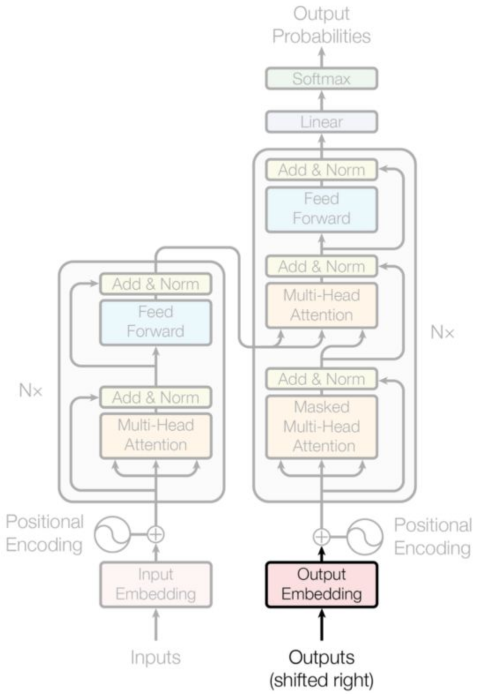
- Encoder-Encoder attention / self-attention
- Feed Forward Neural Network
- Normalization layer

Parameters

- N: layers stacked
- h: number of attention heads
- `d_FF`, `d_key`, `d_value`: sub-layer dimension
- `d_model`: embedding dimensions

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Output "shifted right"



Overview

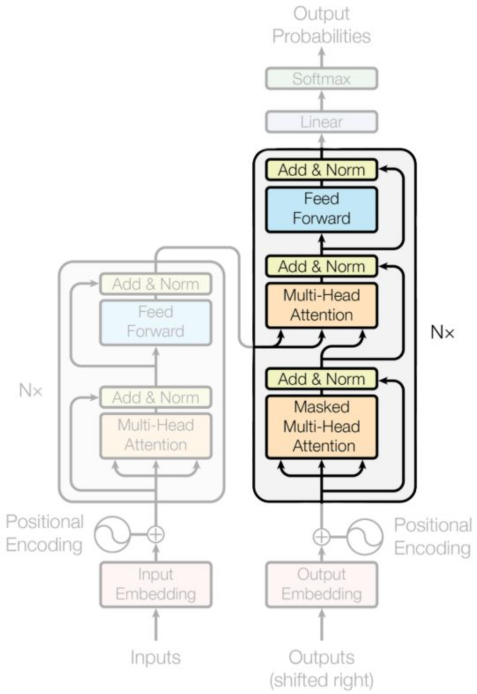
- Learned embeddings for output tokens
- In practice, will start with `[BOS]` during translation

Parameters

- V: vocabulary size
- `d_model`: embedding dimensions

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Decoder



Overview

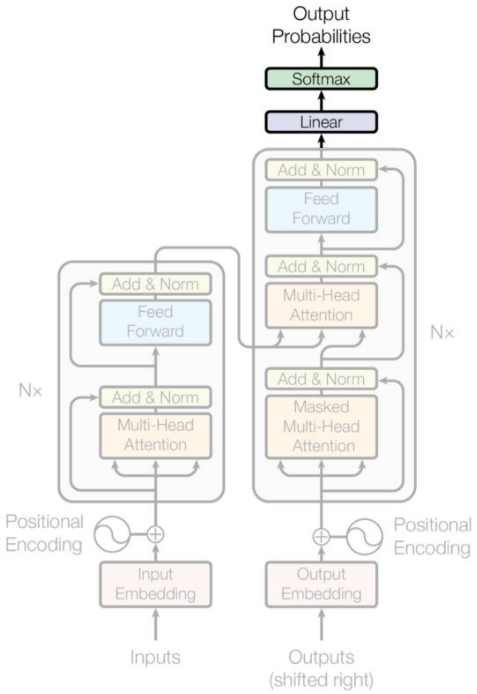
- Decoder-Decoder attention / self-attention
- Encoder-Decoder attention
- Feed Forward Neural Network
- Normalization layer

Parameters

- N: layers stacked
- h: number of attention heads
- `d_FF`, `d_key`, `d_value`: sub-layer dimension
- `d_model`: embedding dimensions

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Output



Overview

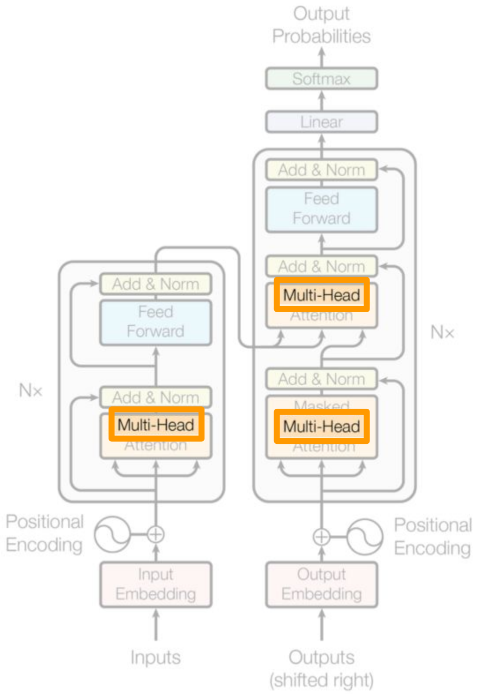
- Linear projection
- Classification problem that outputs probability of belonging to a class, where class = word

Parameters

- V: vocabulary size
- `d_model`: embedding dimensions

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Computational tricks



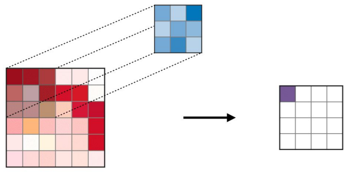
Multi-head attention

Idea:

- Run **multiple** self-attention layers in parallel

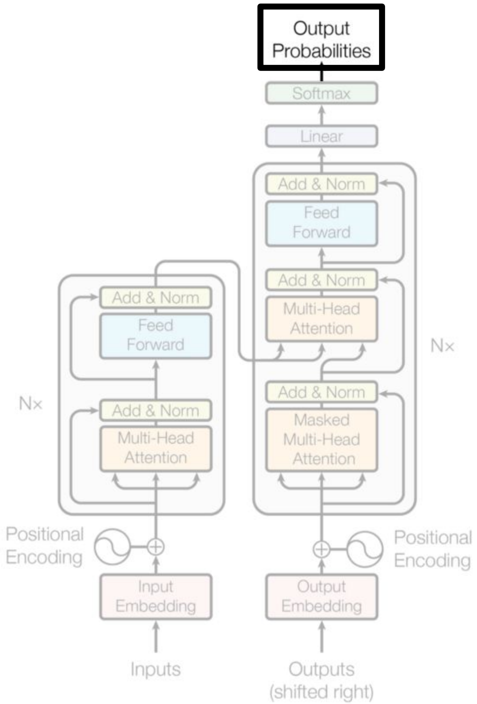
Benefits:

- Enables the model to capture different attention features in parallel
- Comparison: **multiple** filters of a convolutional layer in computer vision



Left figure adapted from "Attention Is All You Need", Vaswani et al., 2017. Right figure from "VIP cheatsheets for Stanford's CS 230", Amidi, 2019.

Computational tricks



Label smoothing

Idea:

- 2015 vision paper: overconfidence is bad
- Introduce **noise** in true labels

$$q(k|x) = \delta_{k,y} \longrightarrow q'(k|x) = (1-\epsilon)\delta_{k,y} + \epsilon u(k)$$

Benefits:

- General technique that prevents overfitting
- Improves accuracy and BLEU score

Figure adapted from "Attention Is All You Need", Vaswani et al., 2017.

Transformers & Large Language Models

NLP overview

Tokenization

Word representation

RNNs

Self-attention mechanism

Transformer architecture

End-to-end example

Stitching all the pieces together with an example

```
A cute teddy bear is reading.
```

Stitching all the pieces together with an example

`A`	`cute`	`teddy bear`	`is`	`reading`	`.`

Stitching all the pieces together with an example

[BOS]	A	cute	teddy bear	is	reading	.	[EOS]

Stitching all the pieces together with an example

[BOS]	A	cute	teddy bear	is	reading	.	[EOS]

Stitching all the pieces together with an example

embedding

[BOS] | A | cute | teddy bear | is | reading | . | [EOS]

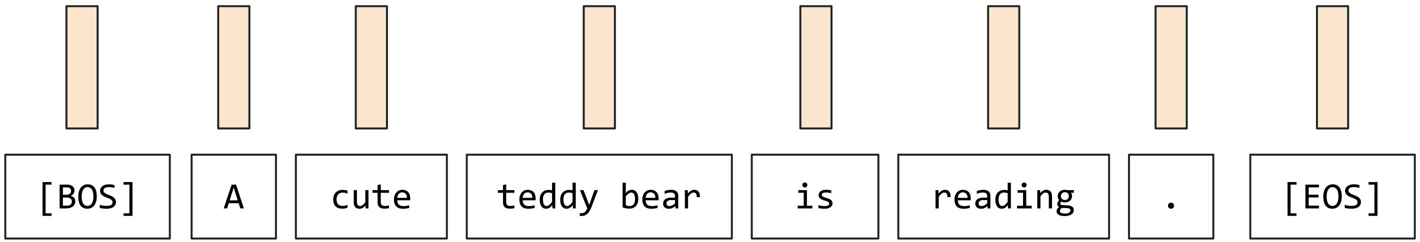
Stitching all the pieces together with an example

Stitching all the pieces together with an example

position-aware embedding

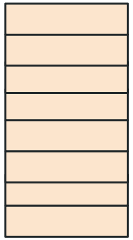
[BOS]	A	cute	teddy bear	is	reading	.	[EOS]

Stitching all the pieces together with an example



position-aware embeddings

Stitching all the pieces together with an example



[BOS] | A | cute | teddy bear | is | reading | . | [EOS]

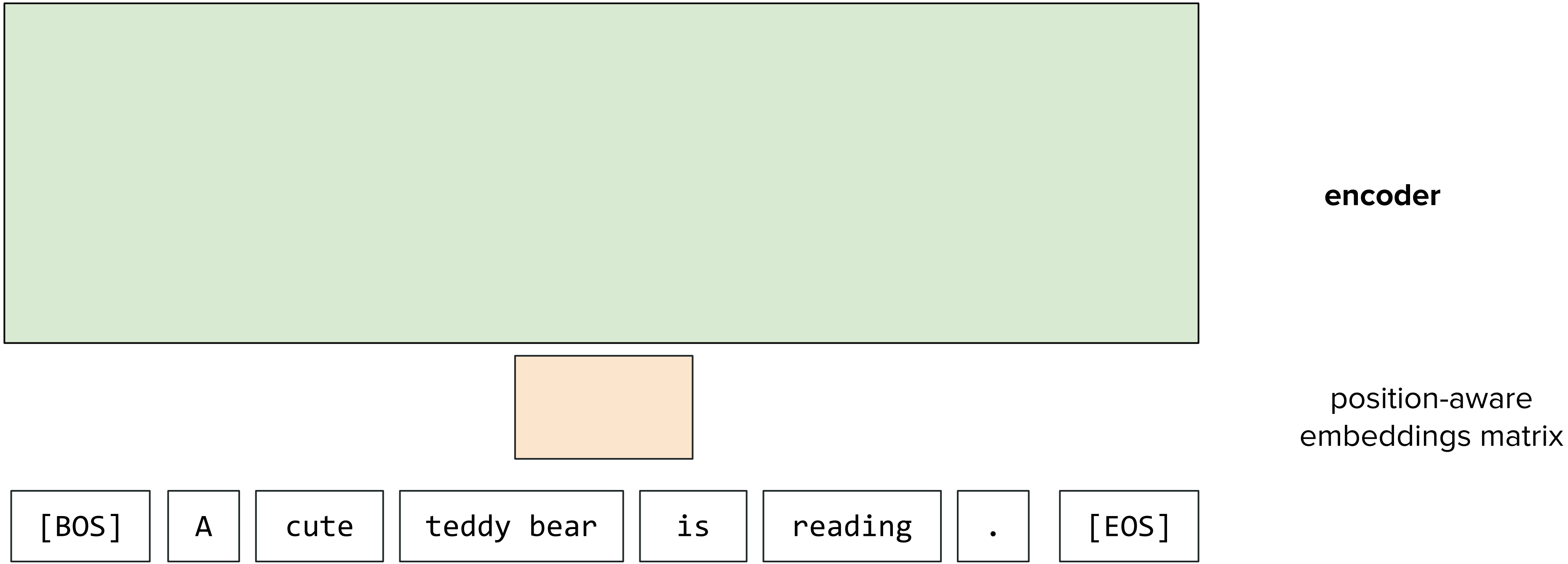
position-aware embeddings matrix

Stitching all the pieces together with an example

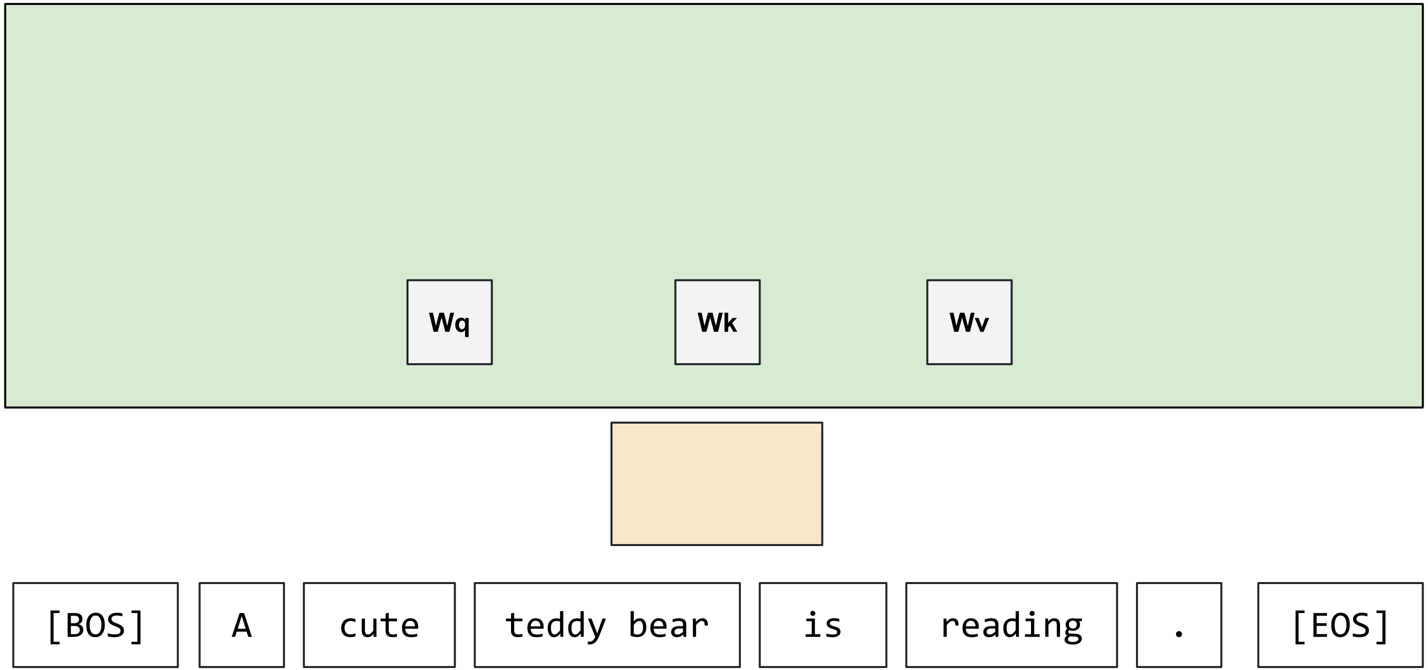
[BOS] A cute teddy bear is reading . [EOS]

position-aware embeddings matrix

Stitching all the pieces together with an example



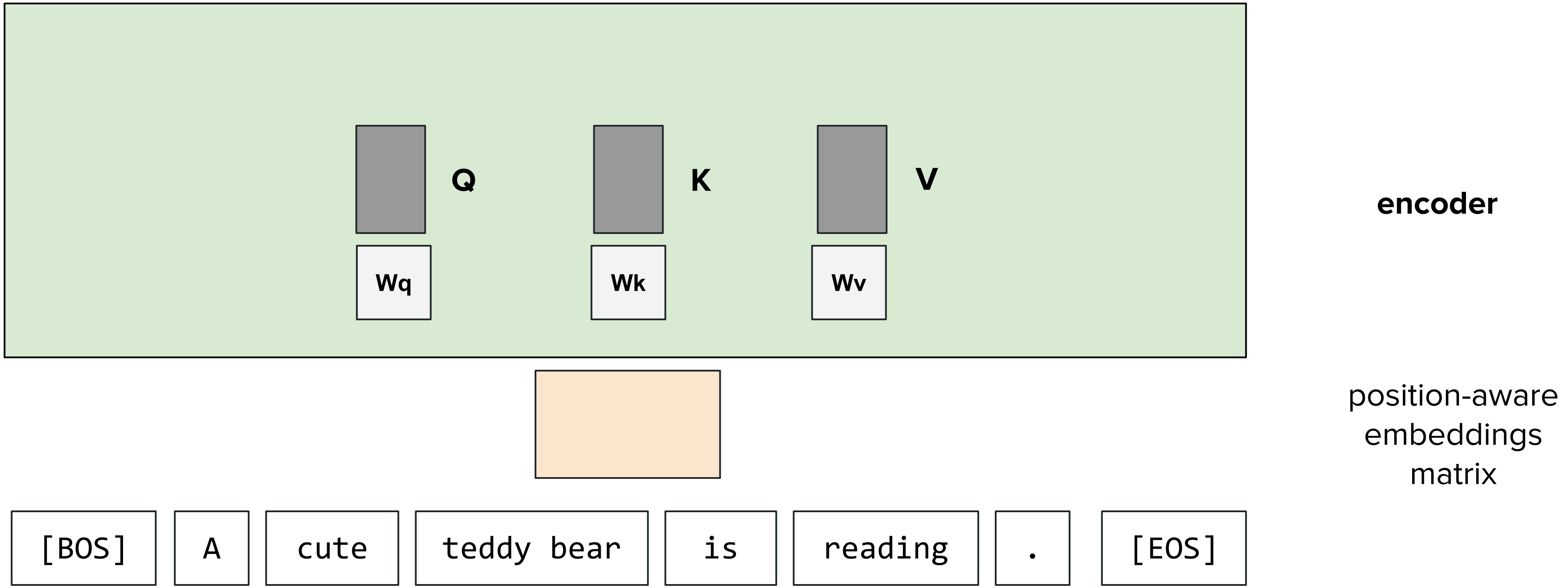
Stitching all the pieces together with an example



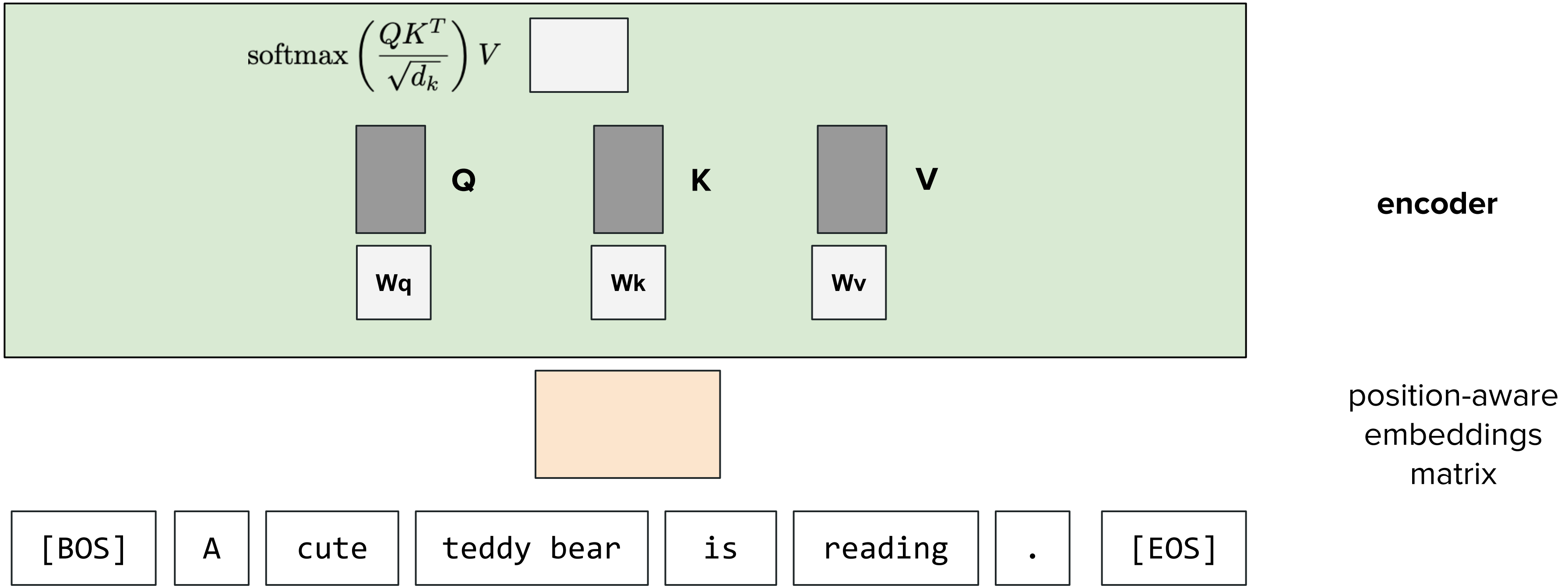
encoder

position-aware embeddings matrix

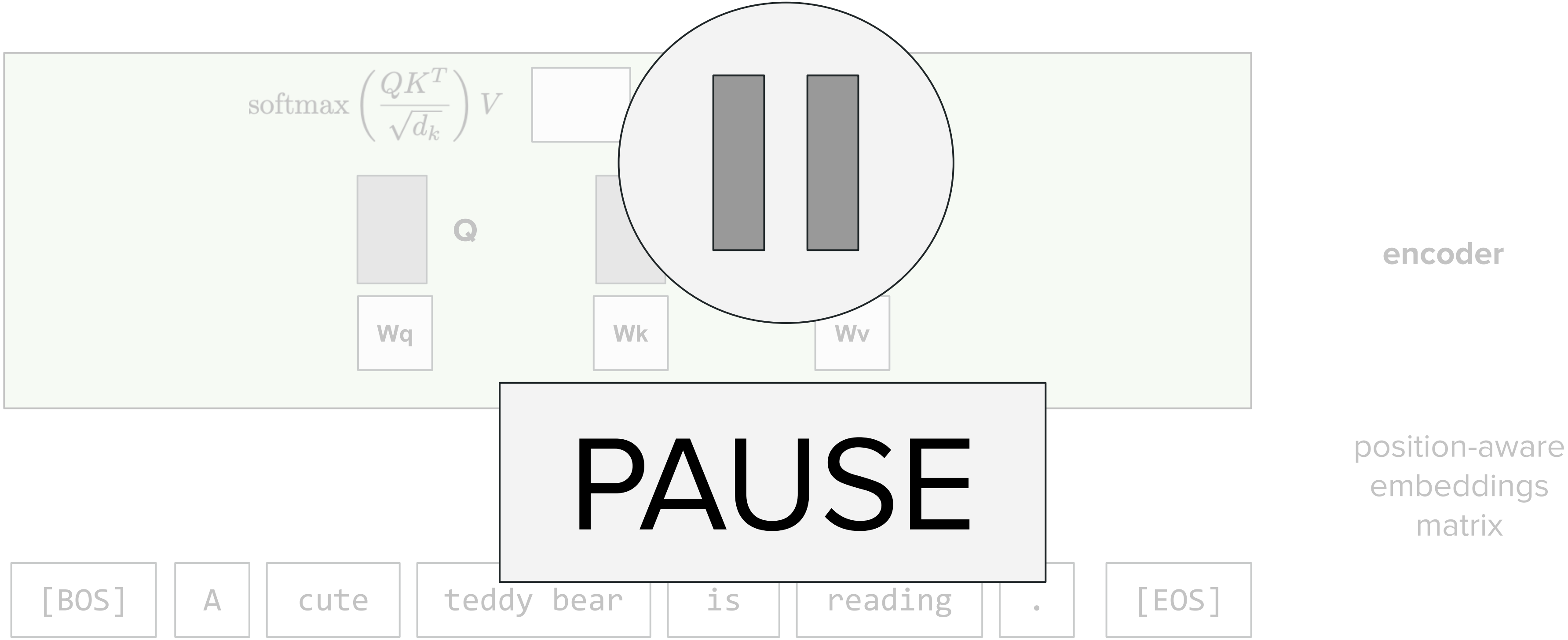
Stitching all the pieces together with an example



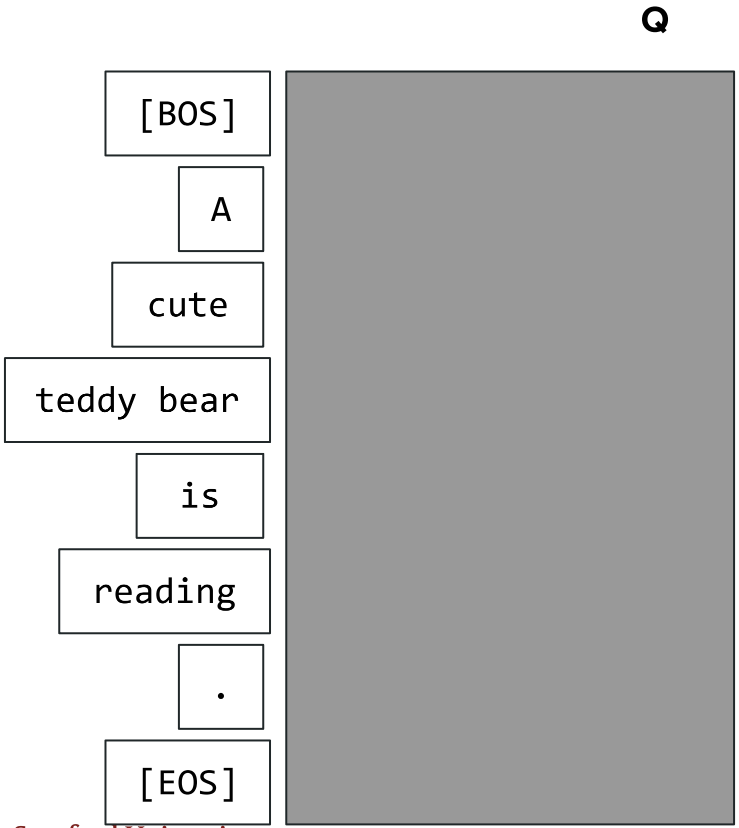
Stitching all the pieces together with an example



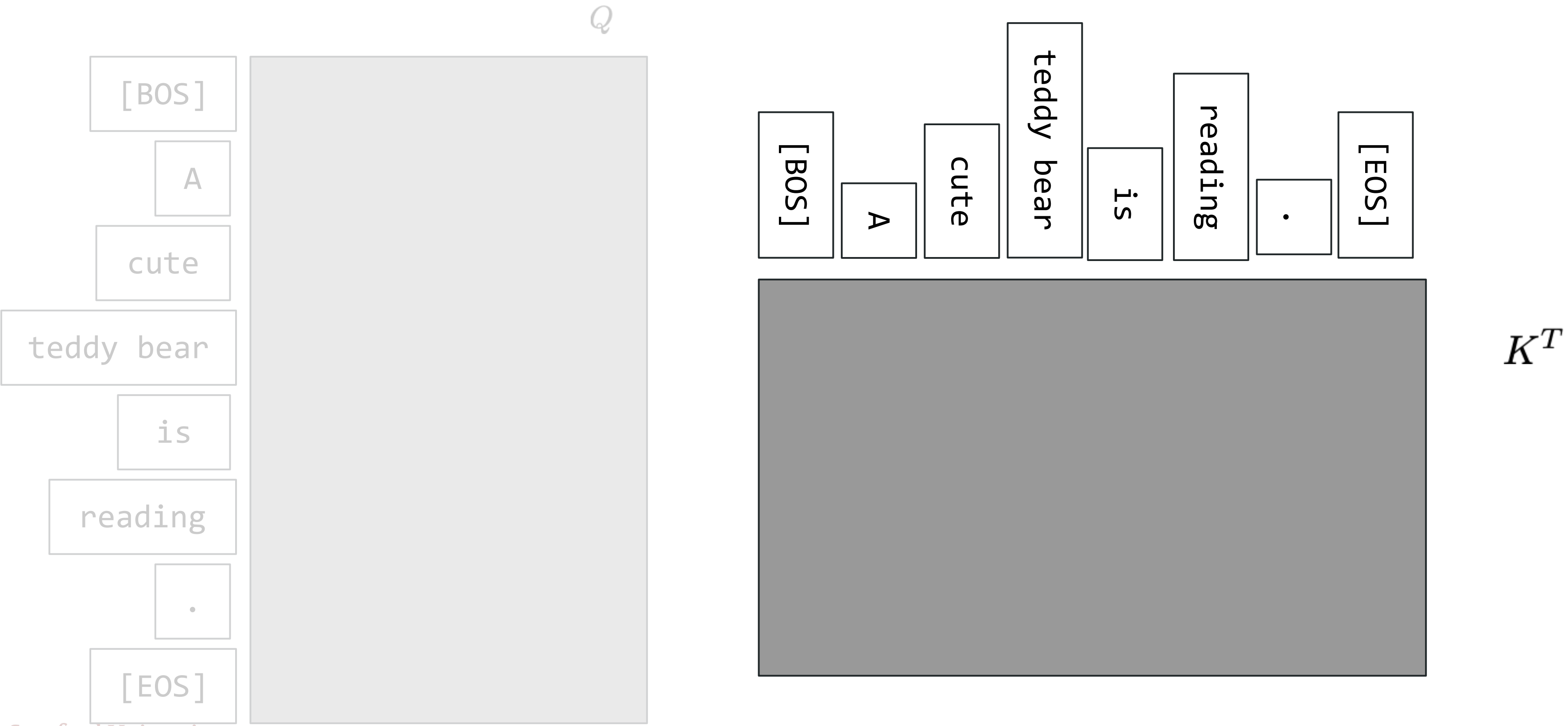
Stitching all the pieces together with an example



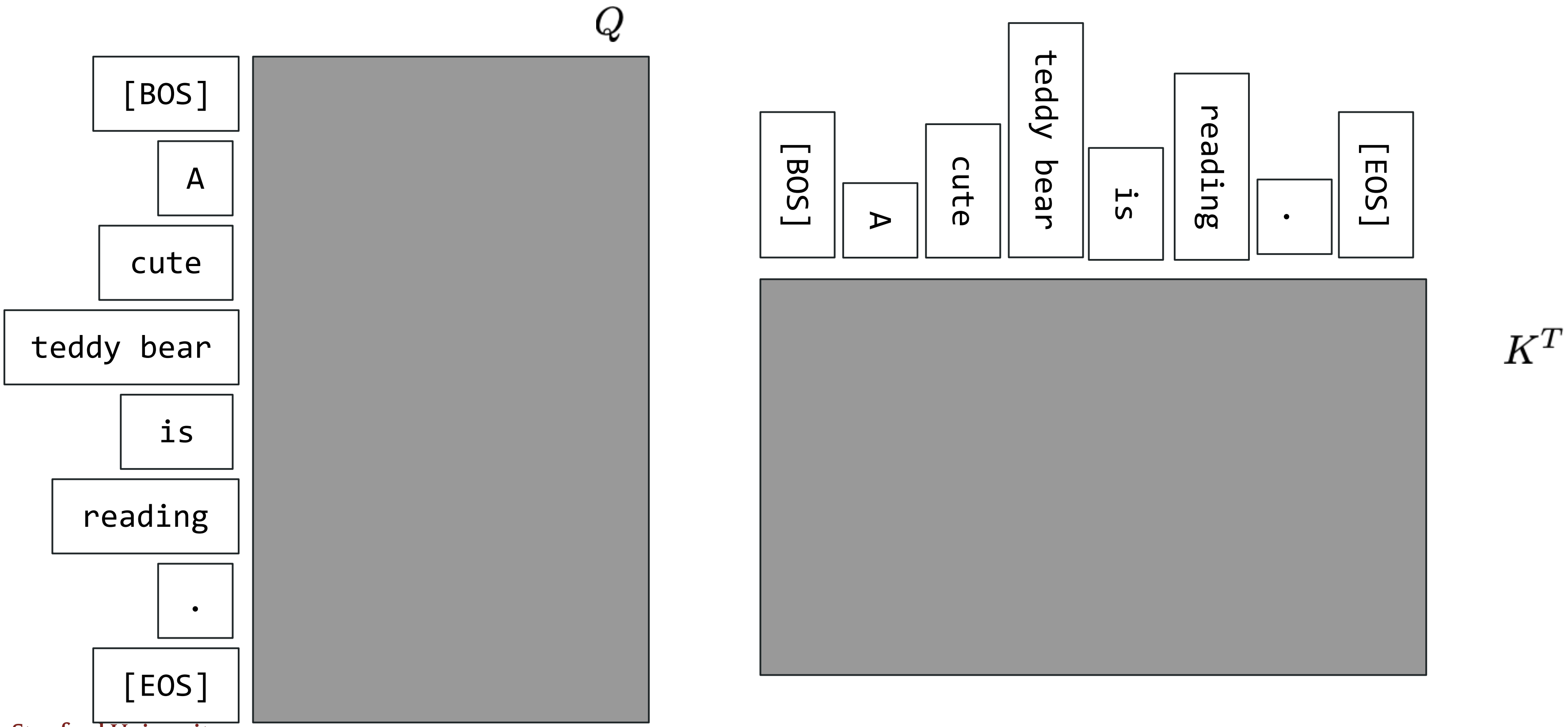
Stitching all the pieces together with an example



Stitching all the pieces together with an example



Stitching all the pieces together with an example



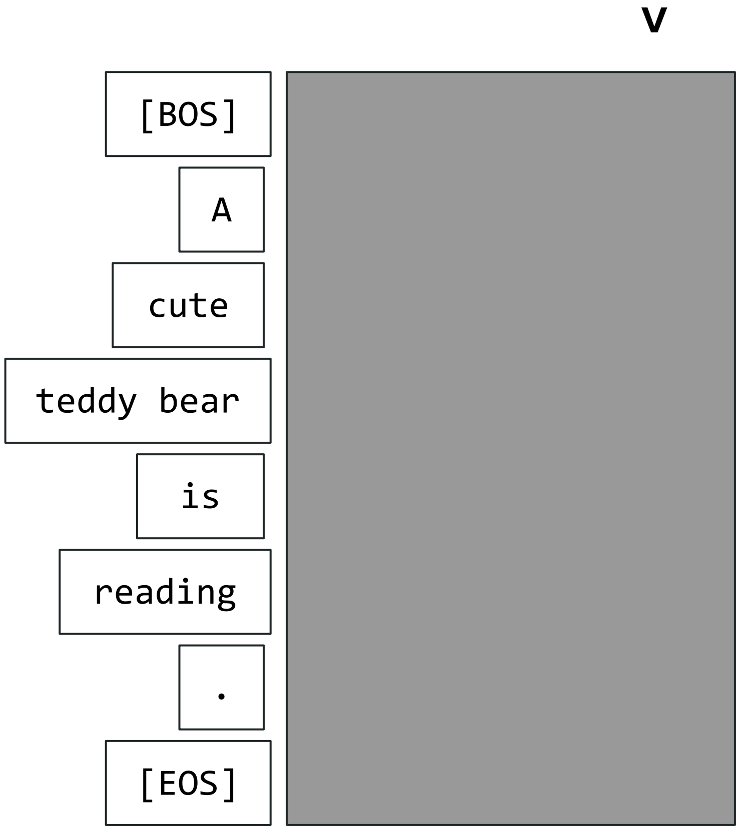
Stitching all the pieces together with an example

$$
\begin{array}{cccc}
\langle q_{[\mathrm{BOS}]}, k_{[\mathrm{BOS}]} \rangle & \langle q_{[\mathrm{BOS}]}, k_{\mathrm{A}} \rangle & \langle q_{[\mathrm{BOS}]}, k_{\mathrm{cute}} \rangle & \cdots \\
\langle q_{\mathrm{A}}, k_{[\mathrm{BOS}]} \rangle & \langle q_{\mathrm{A}}, k_{\mathrm{A}} \rangle & & \\
\langle q_{\mathrm{cute}}, k_{[\mathrm{BOS}]} \rangle & & \ddots & \\
\vdots & & &
\end{array}
\qquad QK^T
$$

Stitching all the pieces together with an example

$$QK^T$$

$$\begin{matrix}
\langle q_{[\mathrm{BOS}]}, k_{[\mathrm{BOS}]} \rangle & \langle q_{[\mathrm{BOS}]}, k_{\mathrm{A}} \rangle & \langle q_{[\mathrm{BOS}]}, k_{\mathrm{cute}} \rangle & \cdots \\
\langle q_{\mathrm{A}}, k_{[\mathrm{BOS}]} \rangle & \langle q_{\mathrm{A}}, k_{\mathrm{A}} \rangle & & \\
\langle q_{\mathrm{cute}}, k_{[\mathrm{BOS}]} \rangle & & \ddots & \\
\vdots & & &
\end{matrix}$$



Stitching all the pieces together with an example

$$\langle q_{[\mathrm{BOS}]}, k_{[\mathrm{BOS}]} \rangle \, v_{[\mathrm{BOS}]} + \langle q_{[\mathrm{BOS}]}, k_{\mathrm{A}} \rangle \, v_{\mathrm{A}} + \langle q_{[\mathrm{BOS}]}, k_{\mathrm{cute}} \rangle \, v_{\mathrm{cute}} + \ldots$$

$$\langle q_{\mathrm{A}}, k_{[\mathrm{BOS}]} \rangle \, v_{[\mathrm{BOS}]} + \langle q_{\mathrm{A}}, k_{\mathrm{A}} \rangle \, v_{\mathrm{A}} + \langle q_{\mathrm{A}}, k_{\mathrm{cute}} \rangle \, v_{\mathrm{cute}} + \ldots$$

$$\vdots$$

$$QK^T V$$

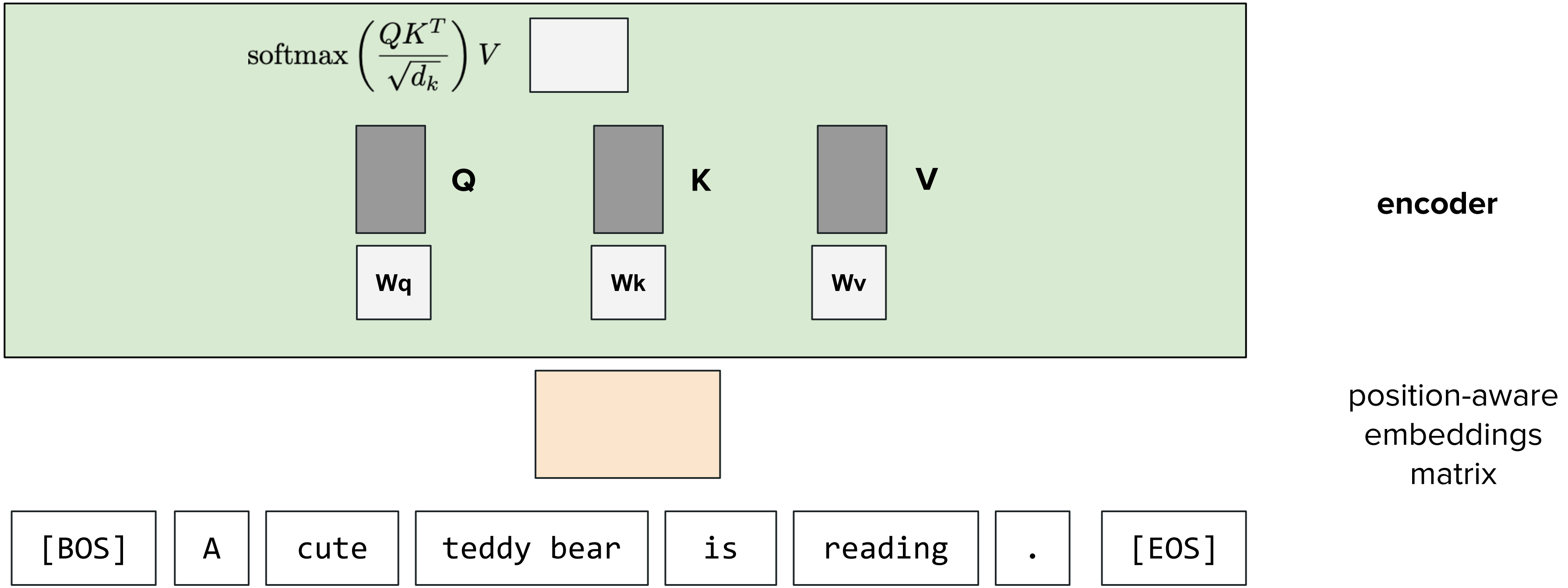
Stitching all the pieces together with an example

weighted average of values

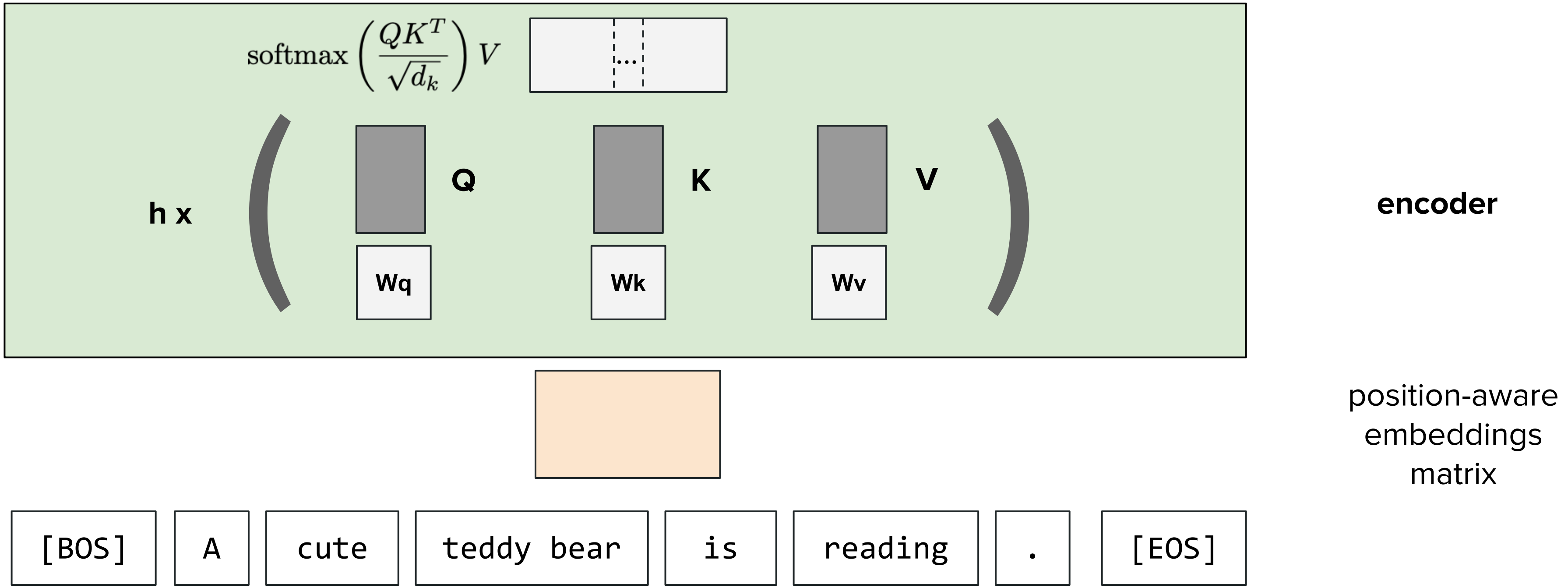
with weights being a function of $\langle q, k \rangle$

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

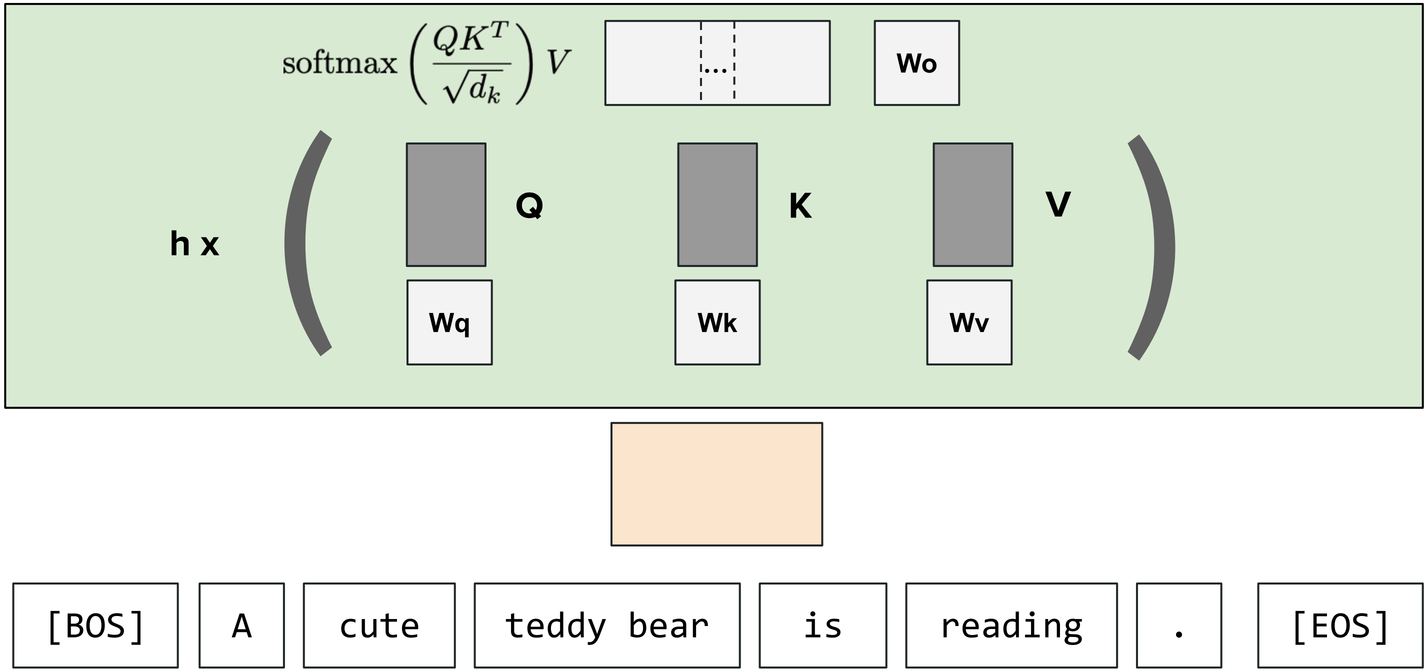
Stitching all the pieces together with an example



Stitching all the pieces together with an example



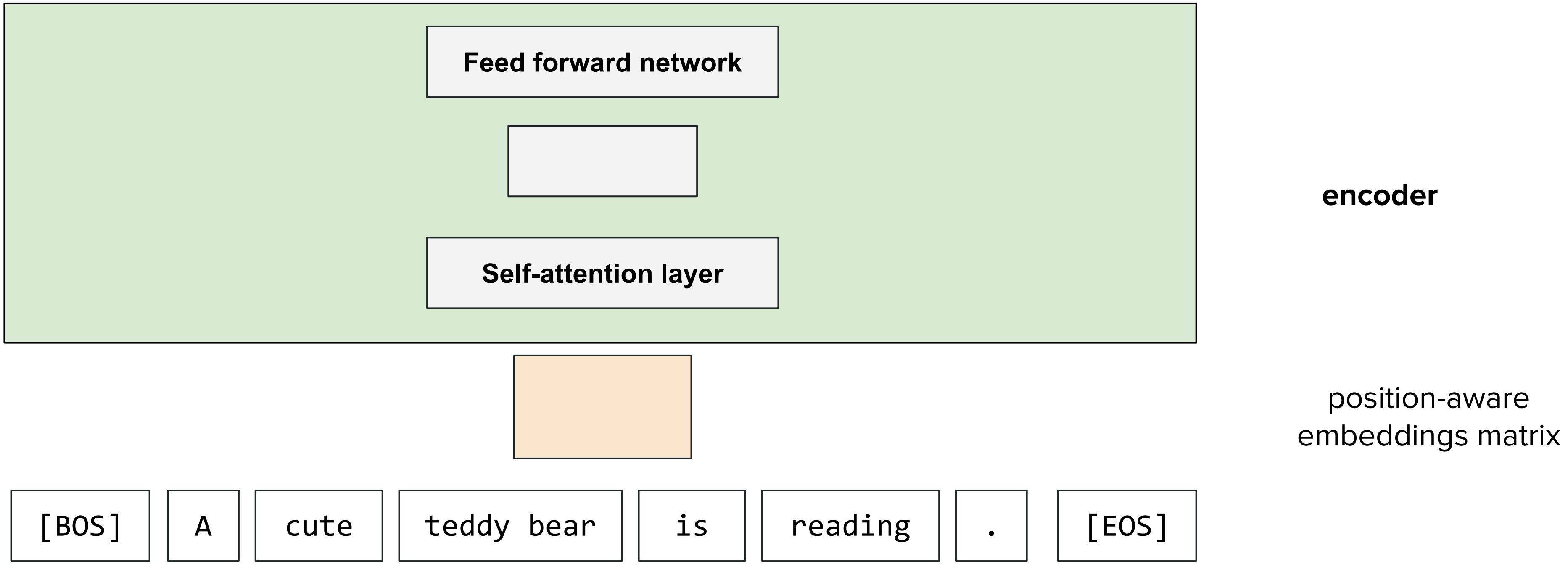
Stitching all the pieces together with an example



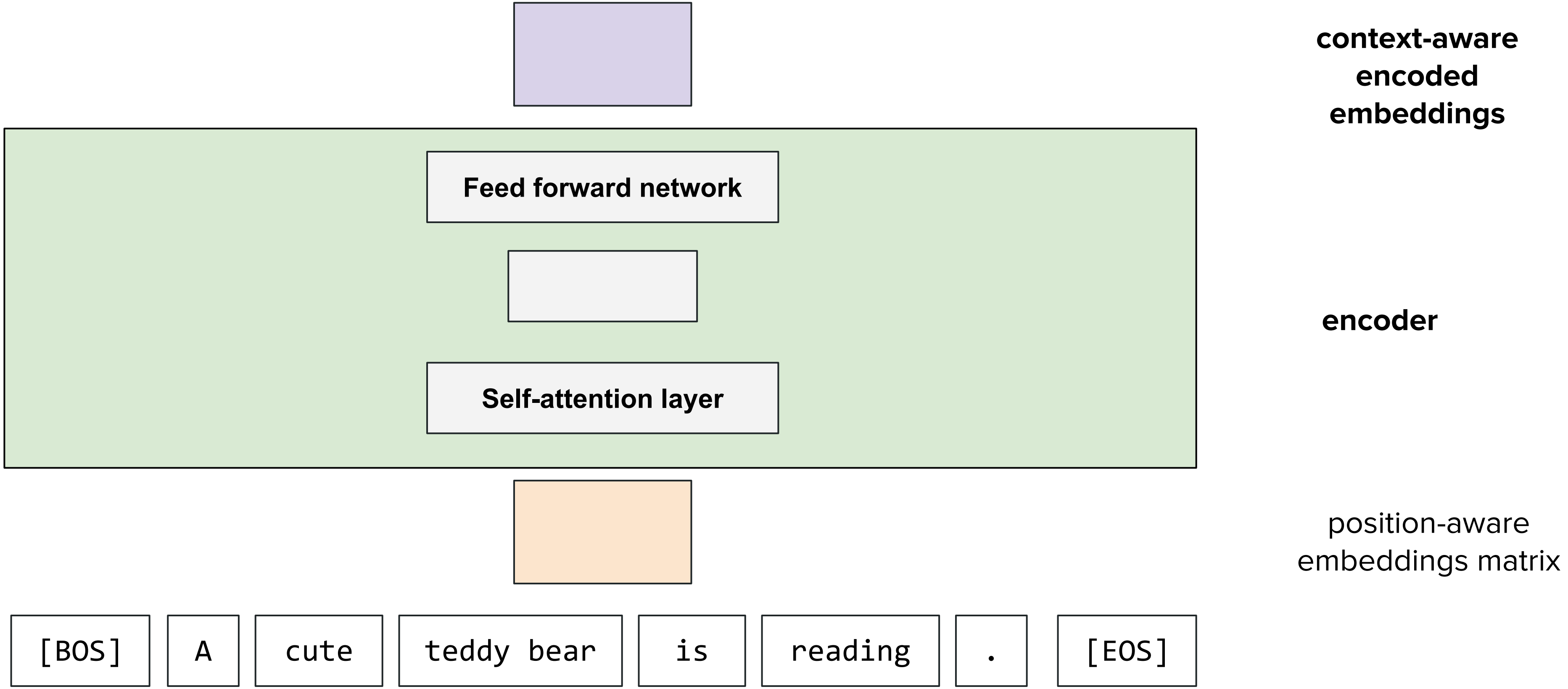
encoder

position-aware embeddings matrix

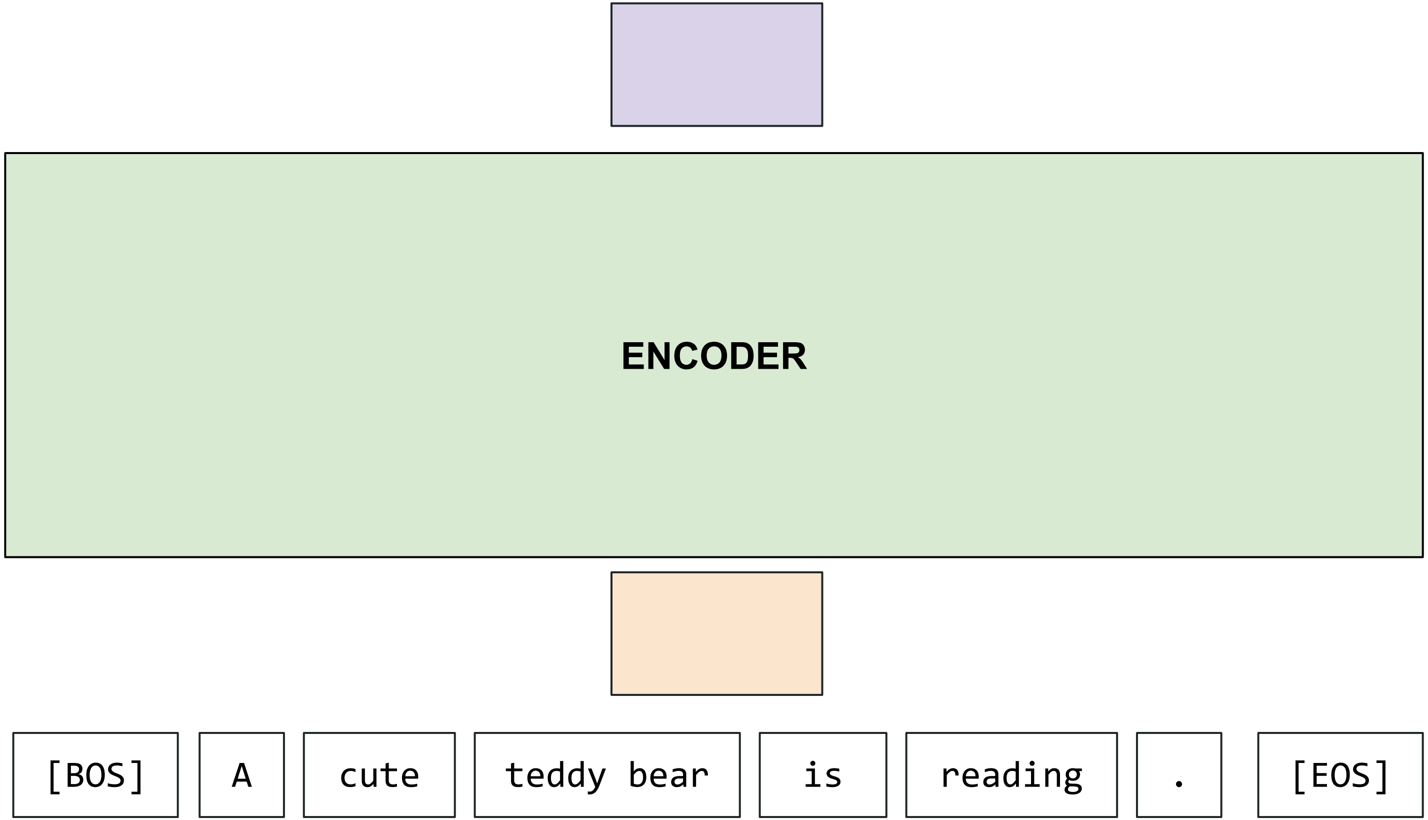
Stitching all the pieces together with an example



Stitching all the pieces together with an example



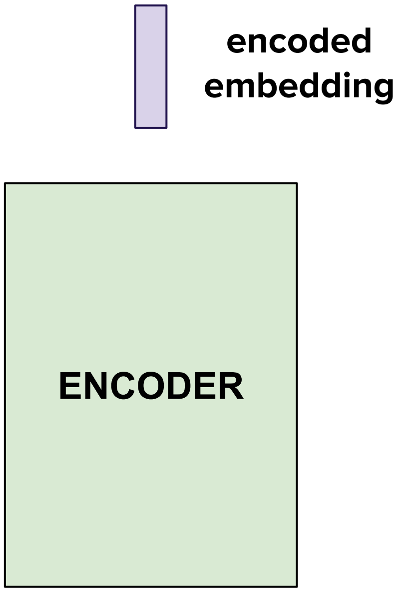
Stitching all the pieces together with an example



context-aware encoded embeddings

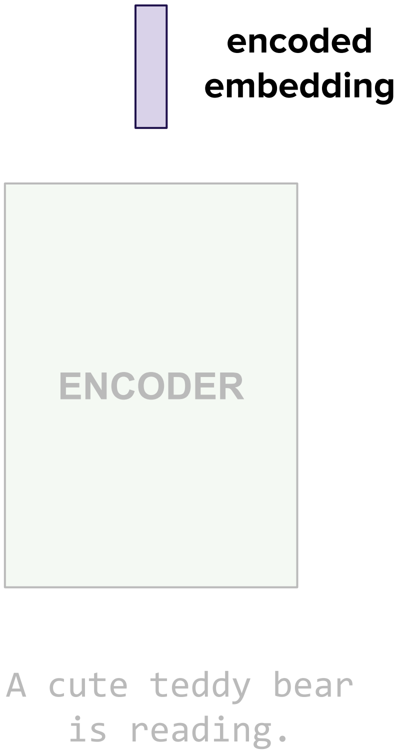
position-aware embeddings matrix

Stitching all the pieces together with an example

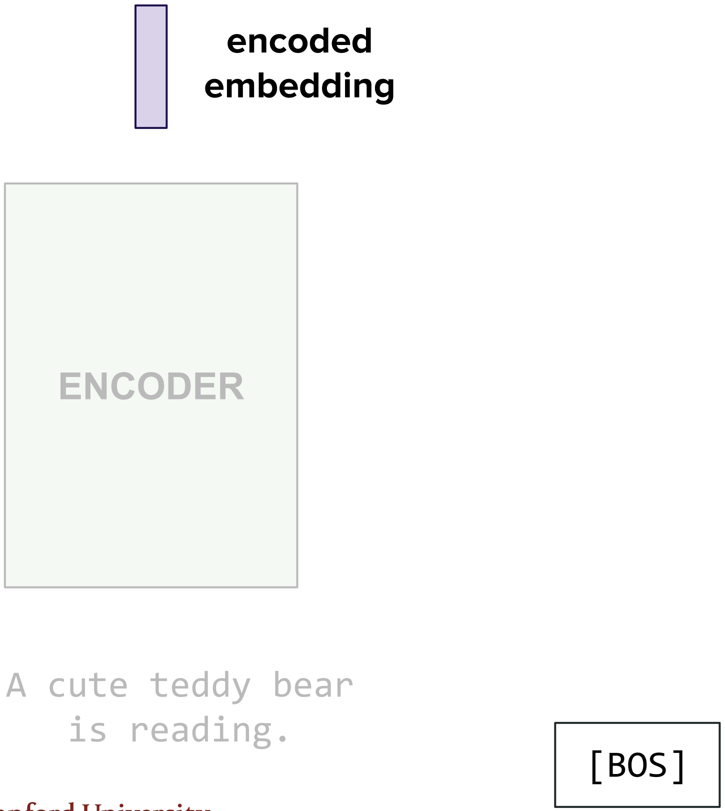



```
A cute teddy bear
   is reading.
```

Stitching all the pieces together with an example

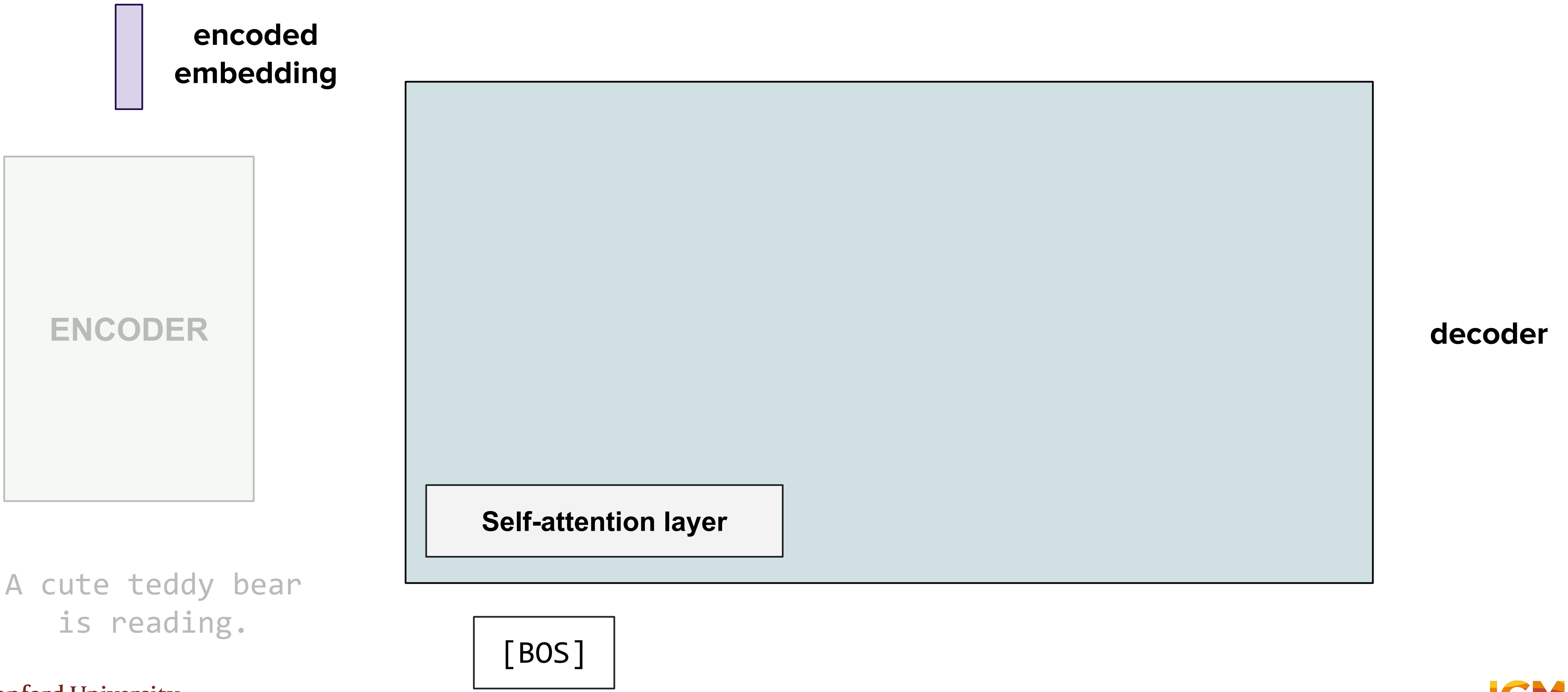


Stitching all the pieces together with an example

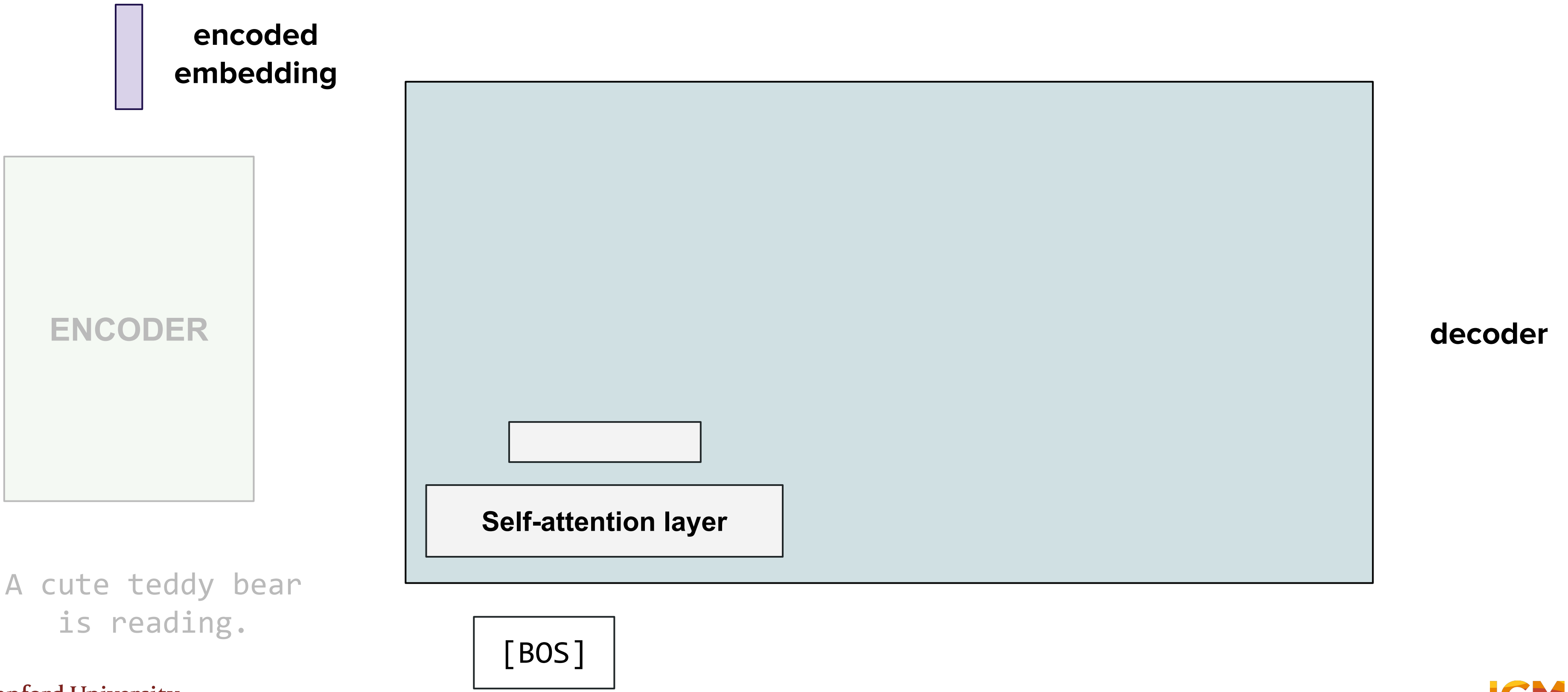


Stitching all the pieces together with an example

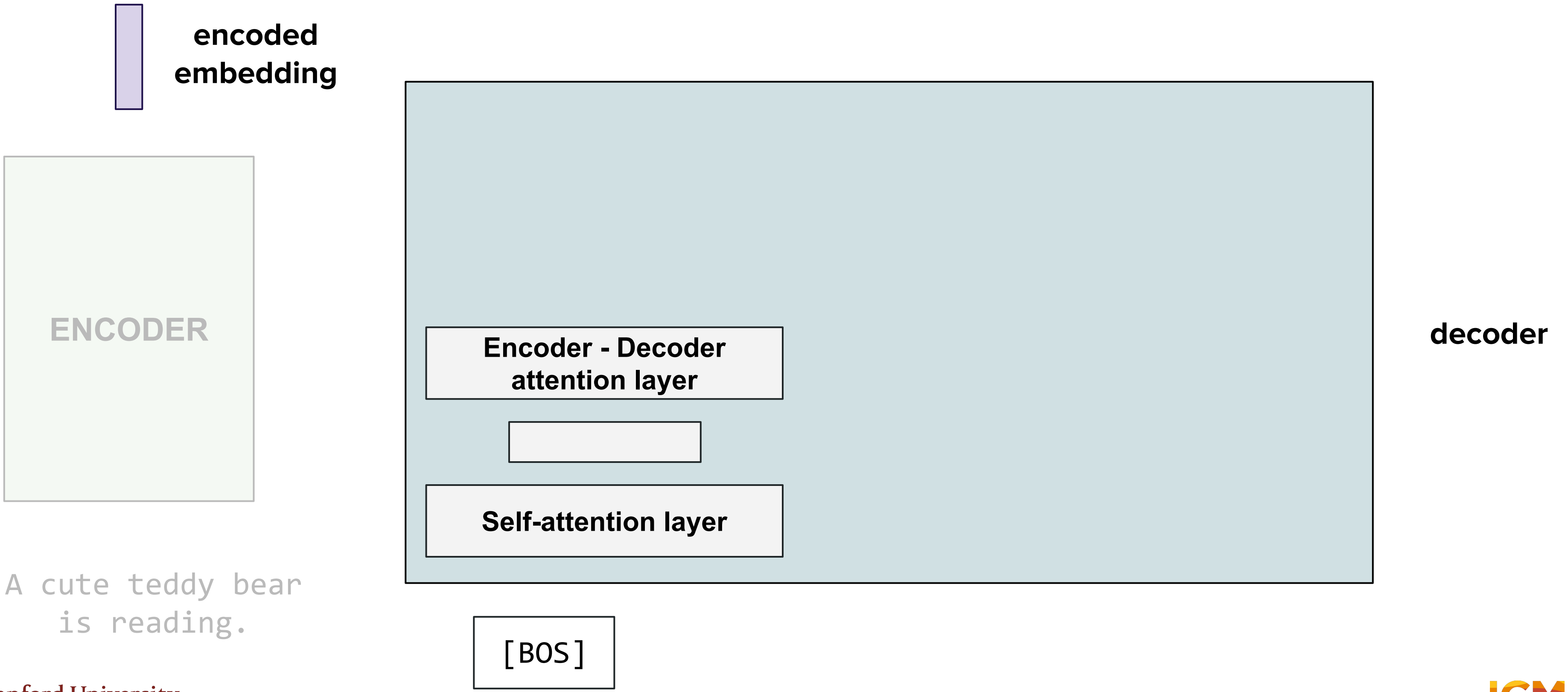
Stitching all the pieces together with an example



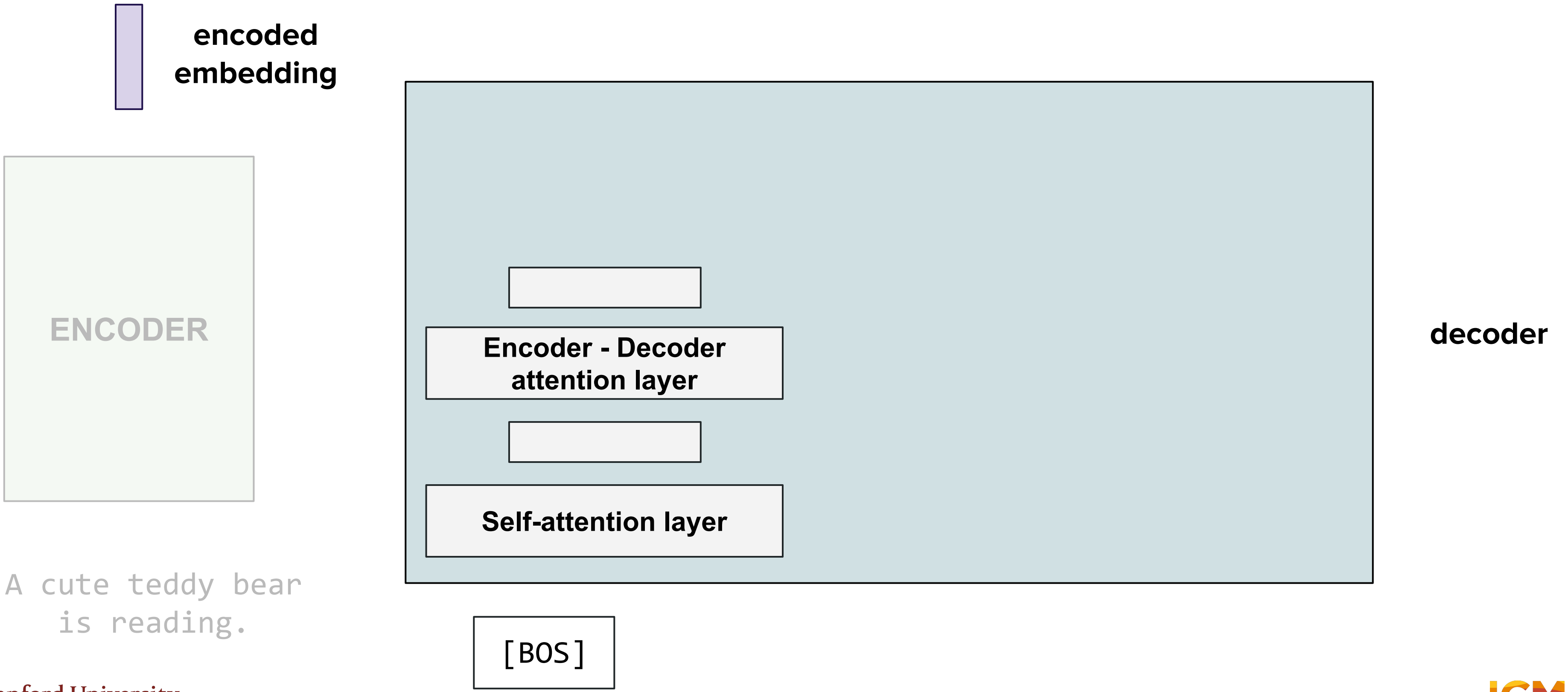
Stitching all the pieces together with an example



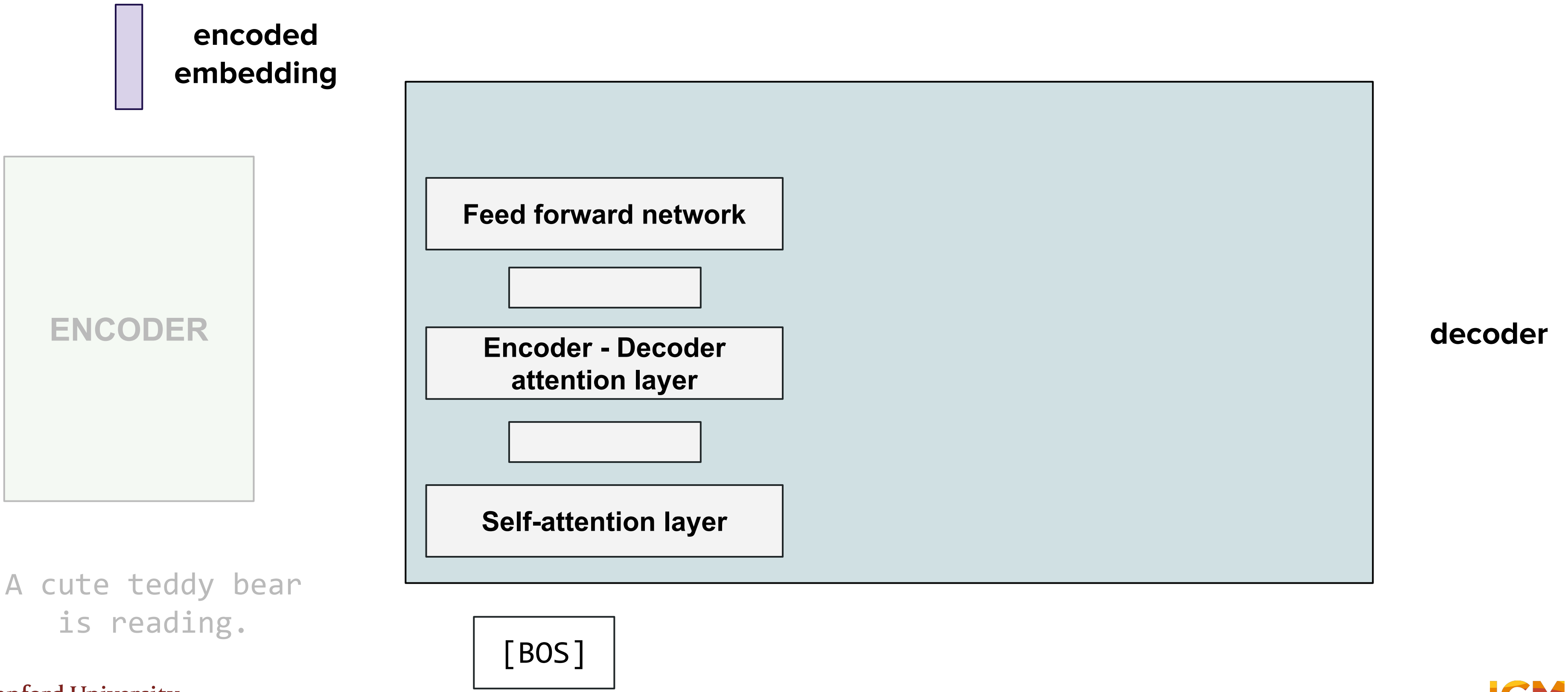
Stitching all the pieces together with an example



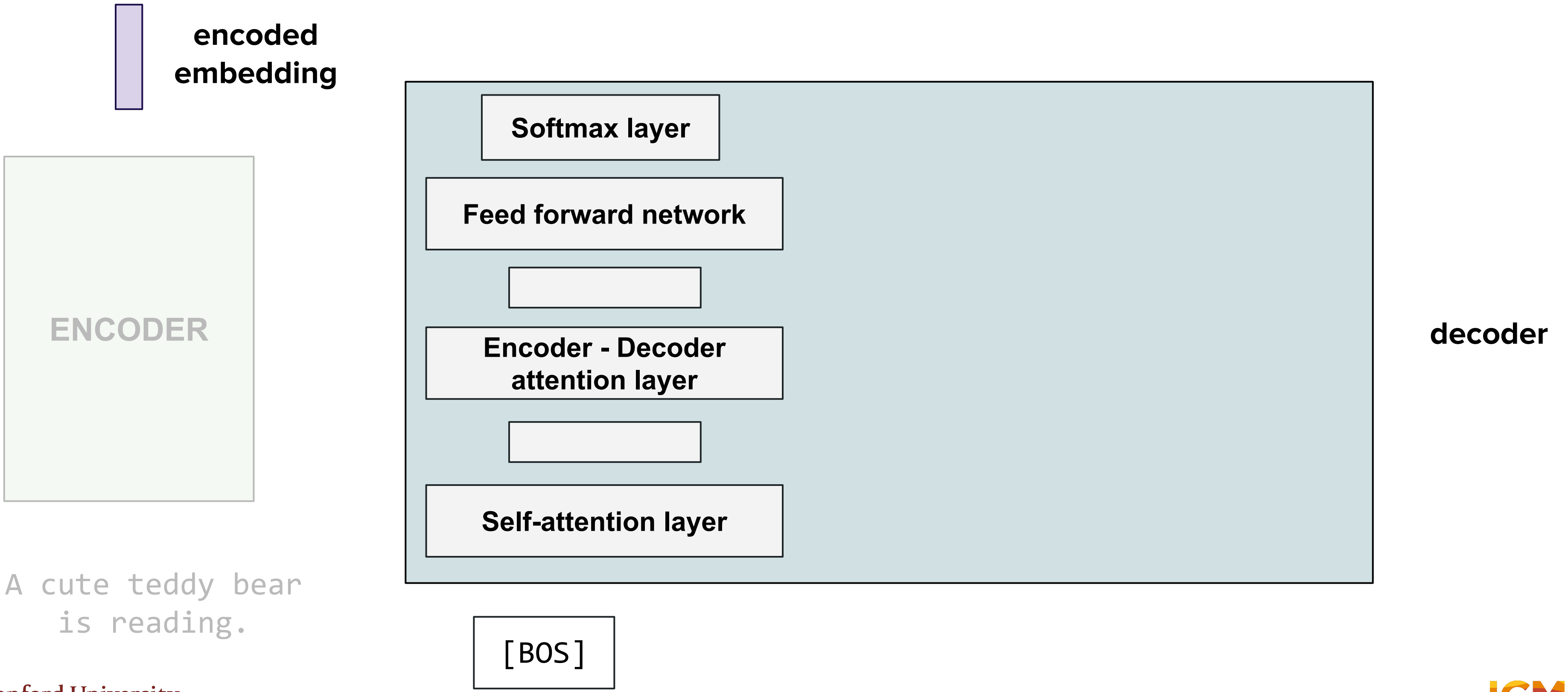
Stitching all the pieces together with an example



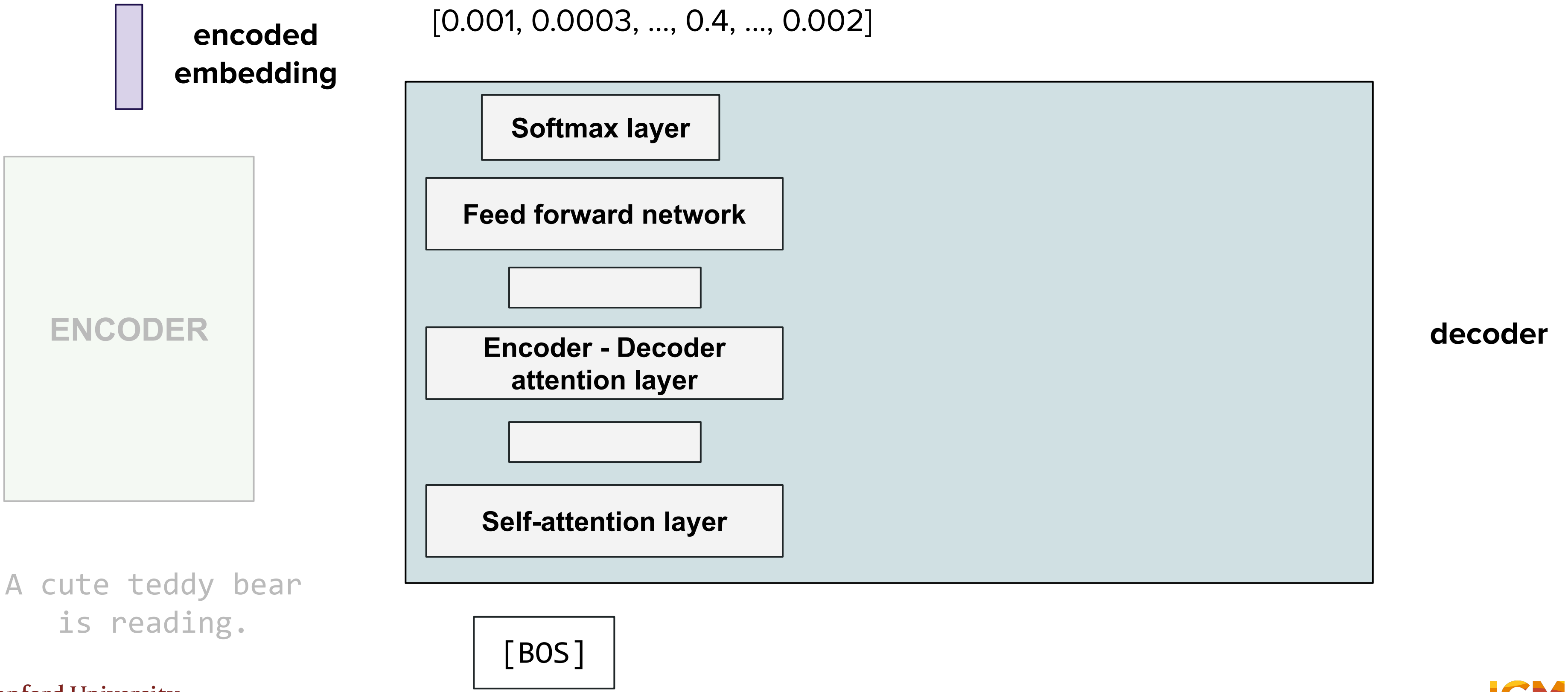
Stitching all the pieces together with an example



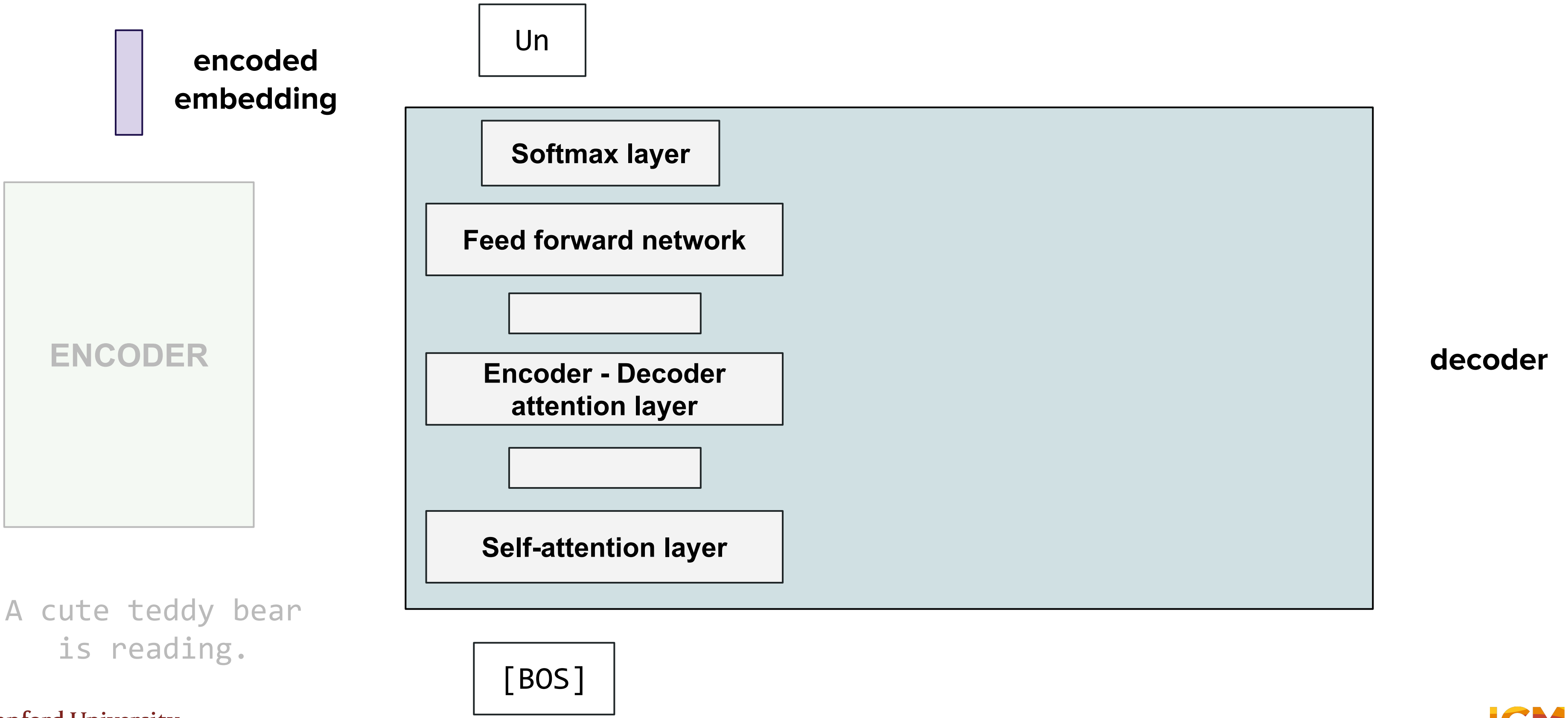
Stitching all the pieces together with an example



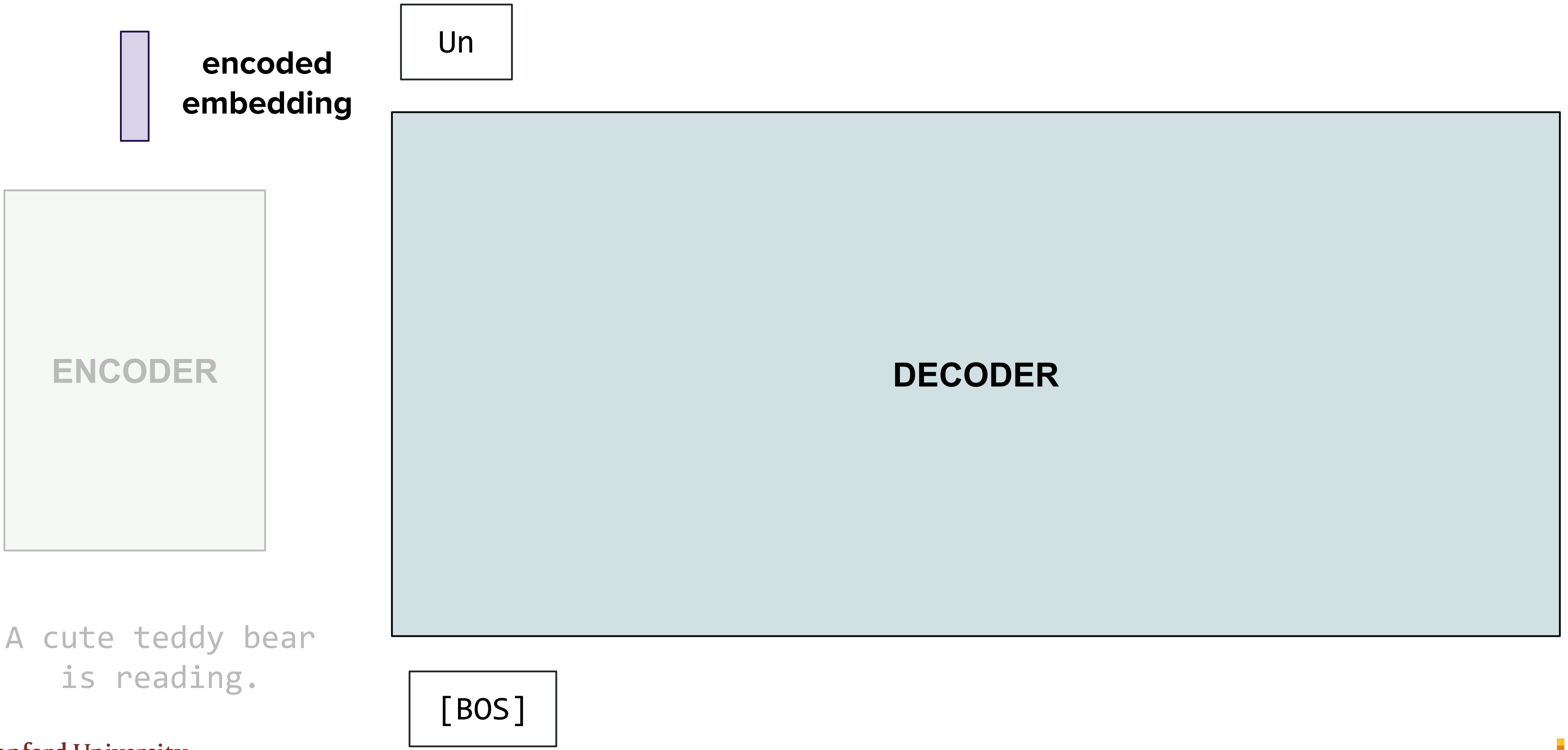
Stitching all the pieces together with an example



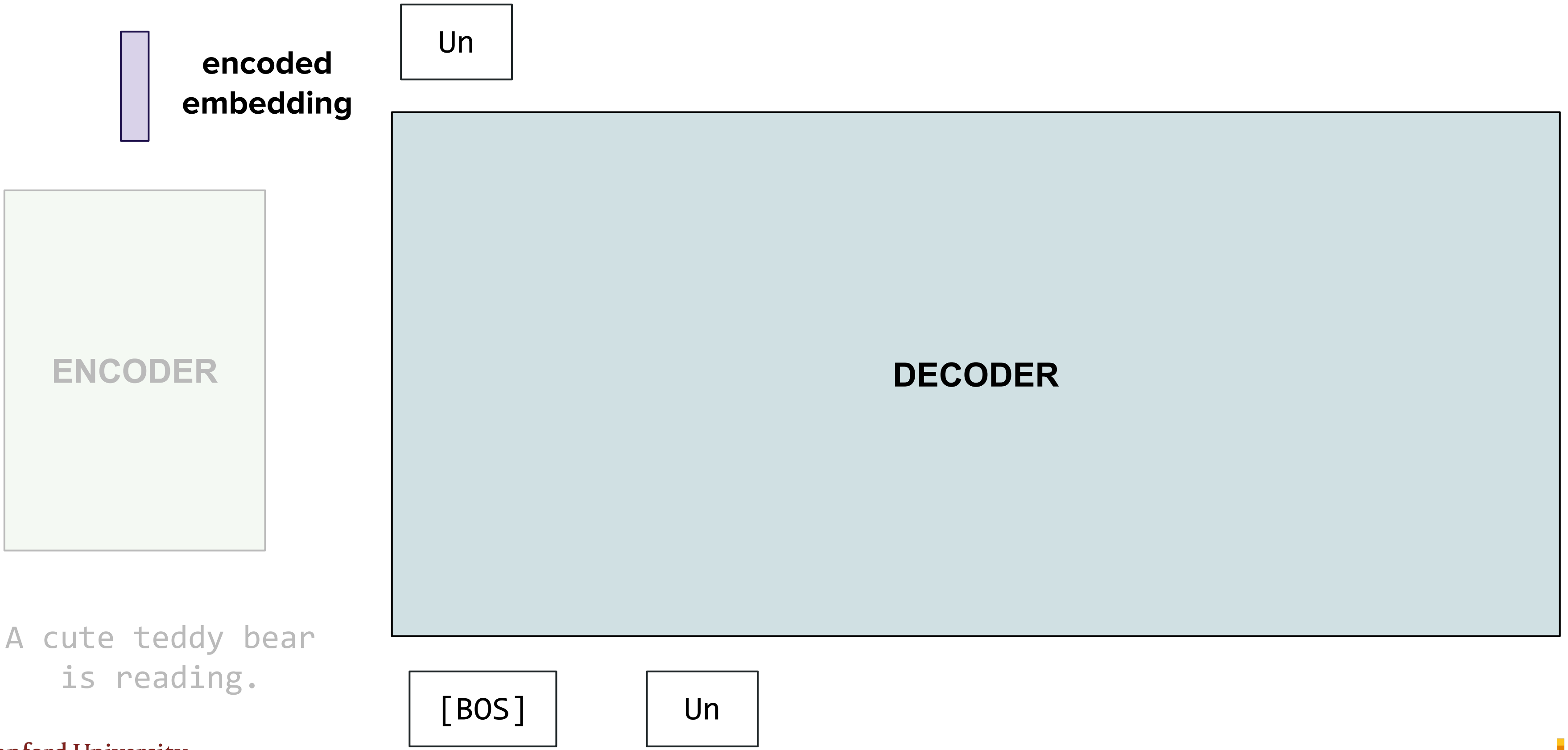
Stitching all the pieces together with an example



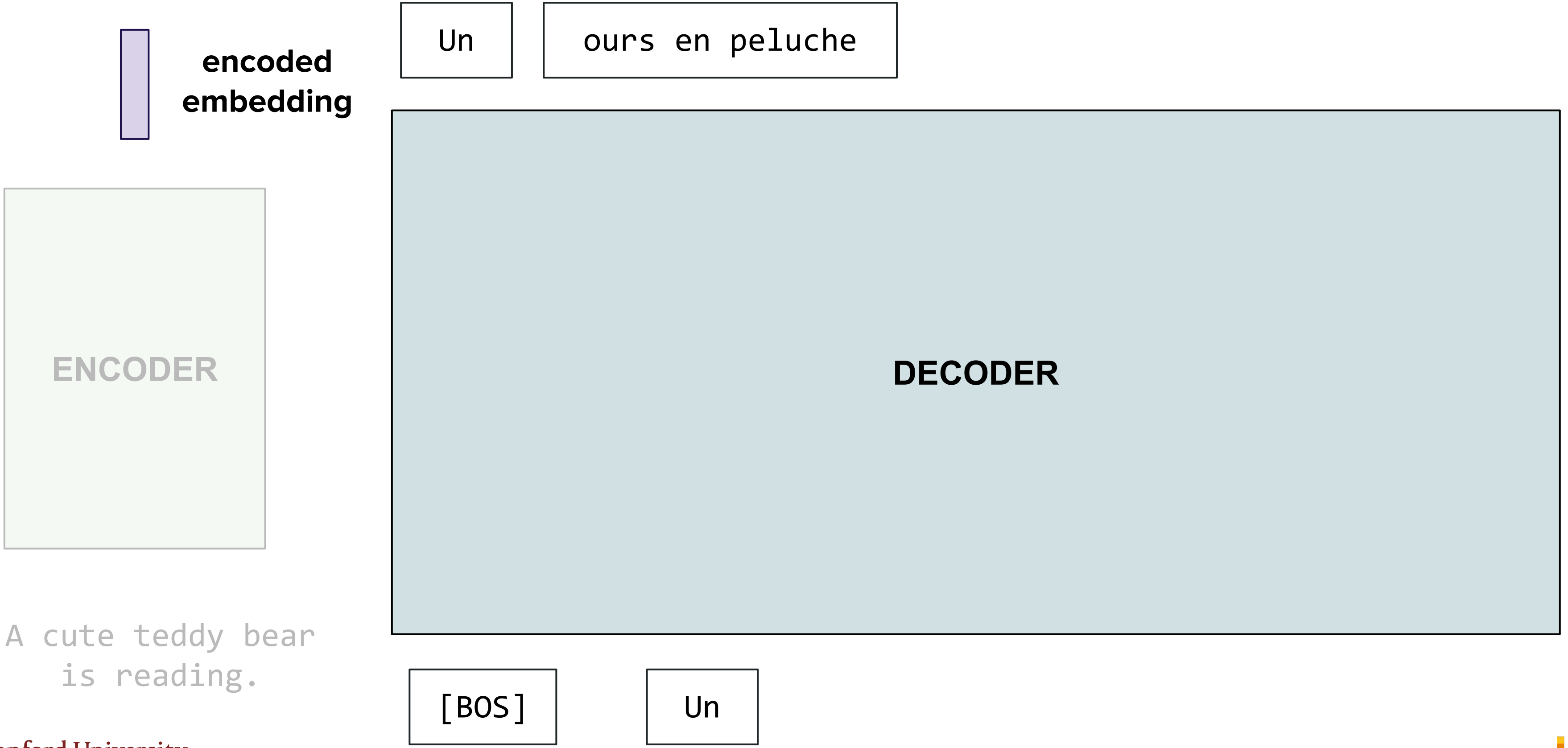
Stitching all the pieces together with an example



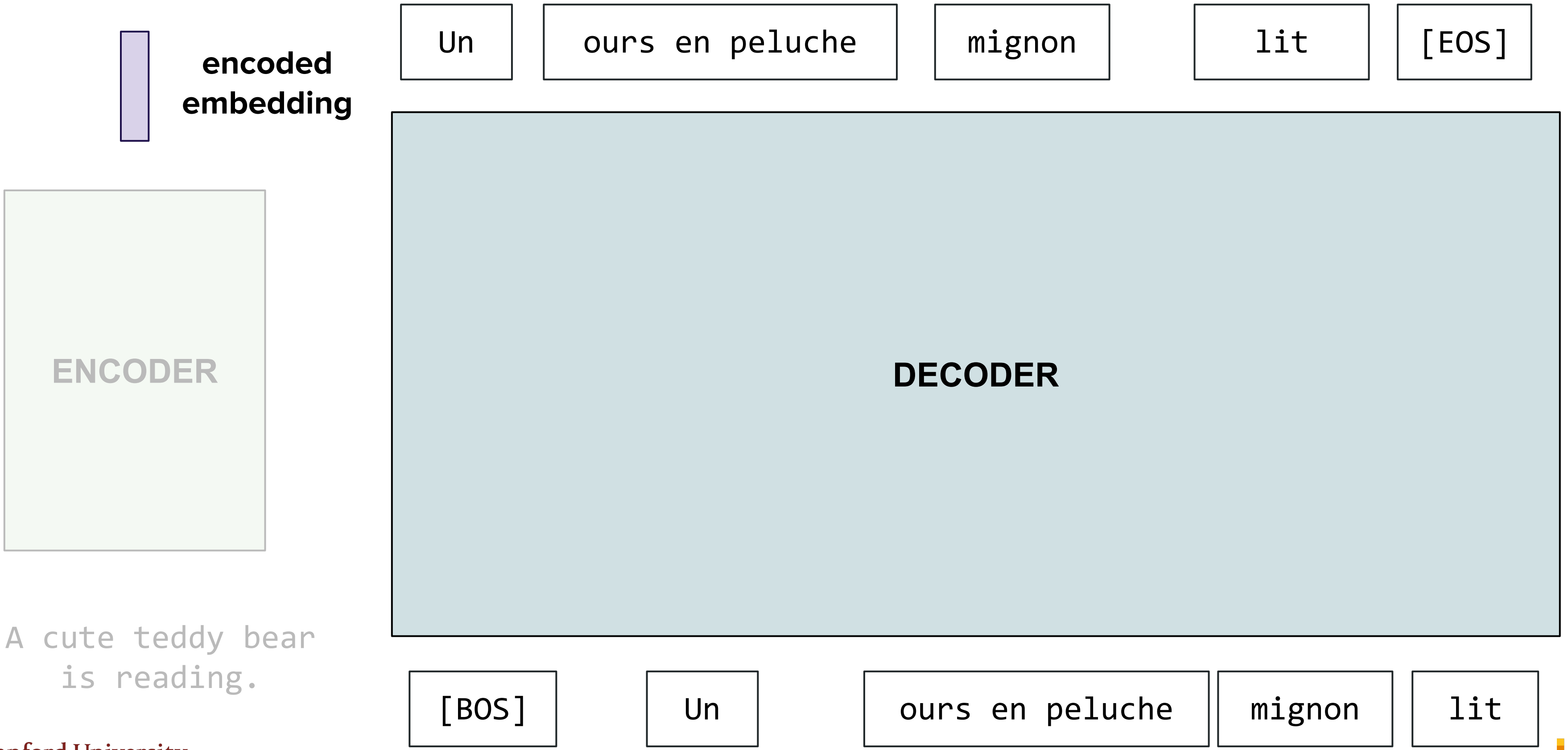
Stitching all the pieces together with an example



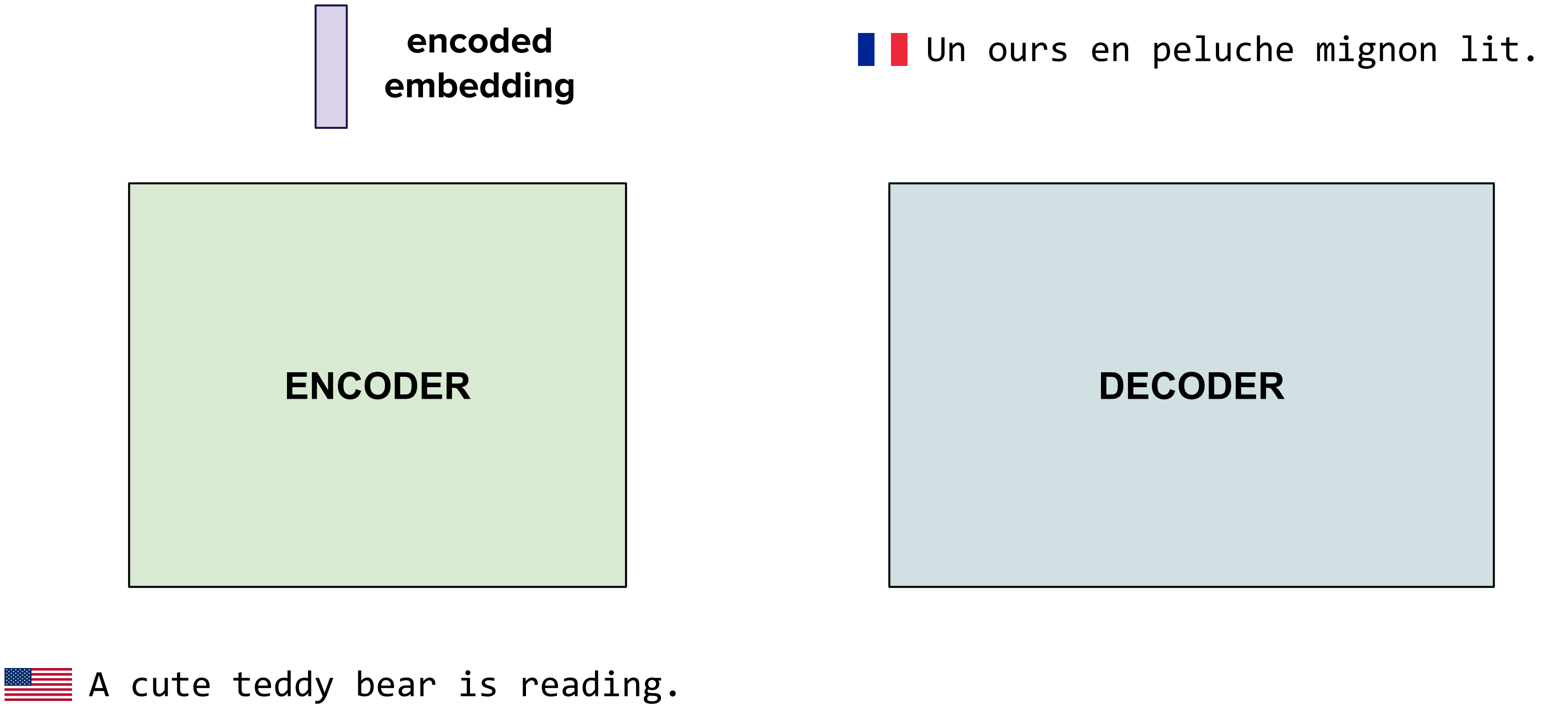
Stitching all the pieces together with an example



Stitching all the pieces together with an example



Stitching all the pieces together with an example



Thank you for your attention!