Deep Learning for NLP, part II

Stanford ICME Summer workshop 2021

Instructor: **Afshine Amidi**

18-20 August 2021

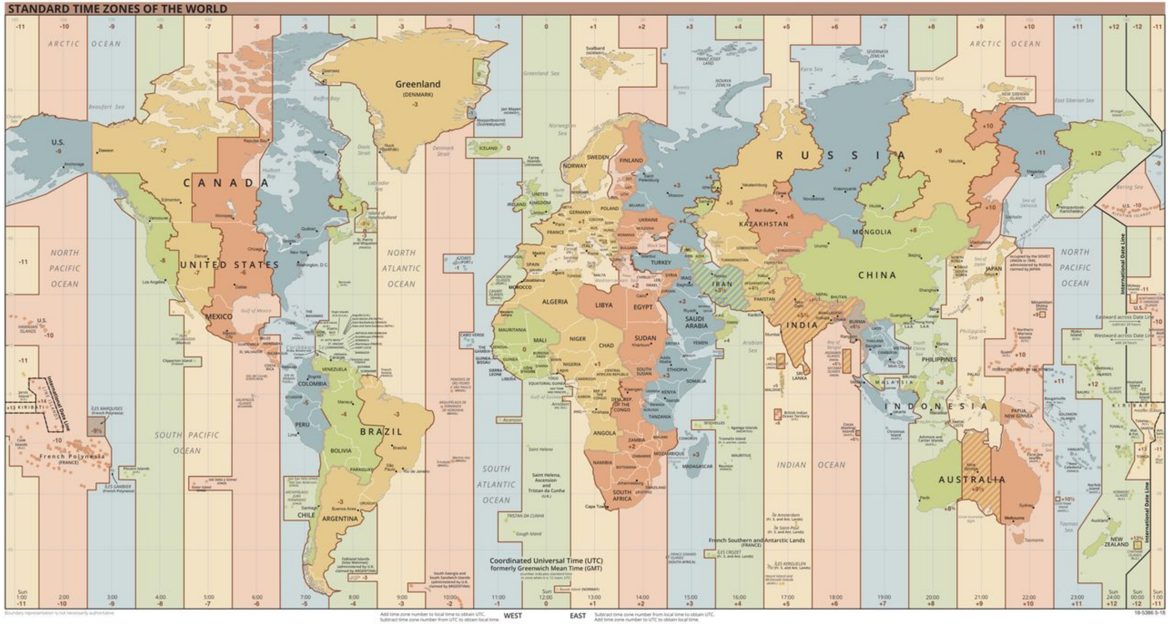
Teaching staff

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

Sam, TA
Stanford ('21)
Peerlift, Iris Labs

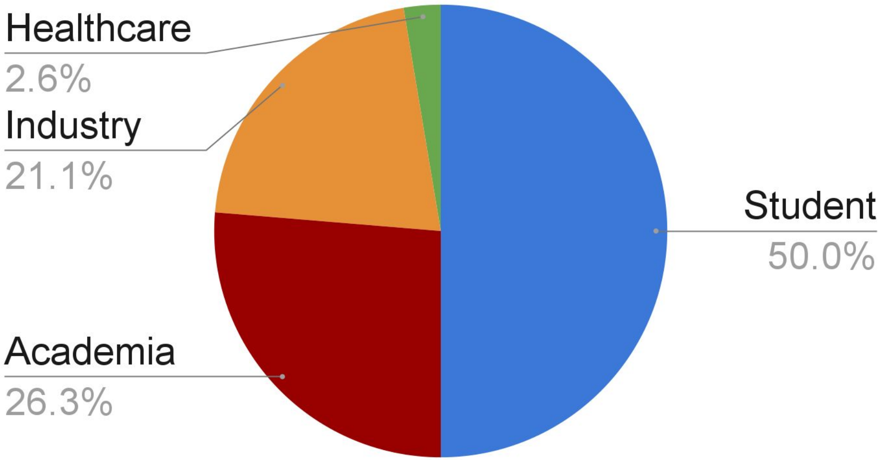
Poll results

What is (approximately) your timezone?



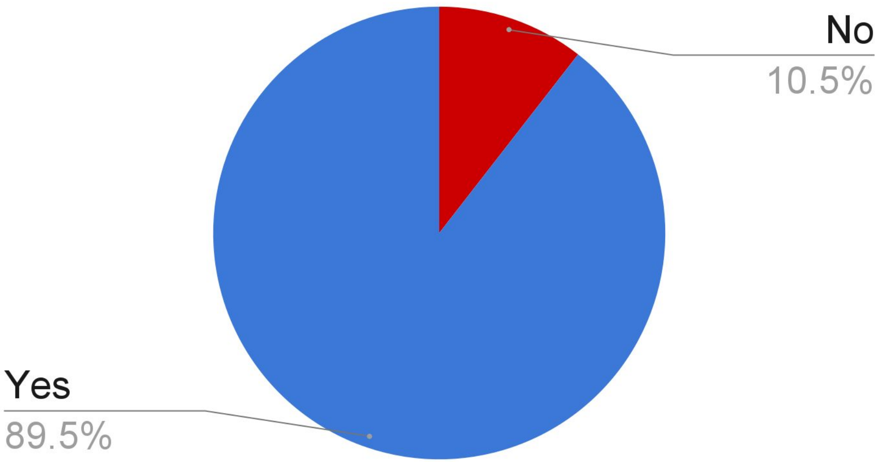
Poll results

Participant category



Poll results

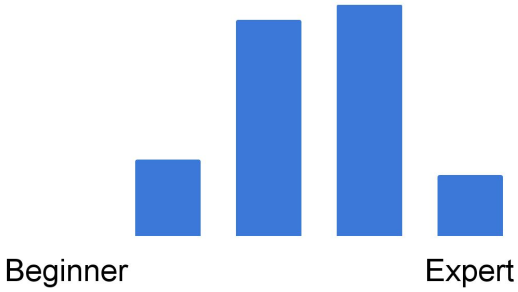
Have you attended the first part of the ICME Workshop NLP series?



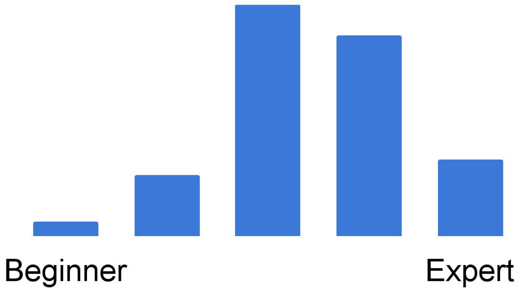
Poll results

Familiarity level

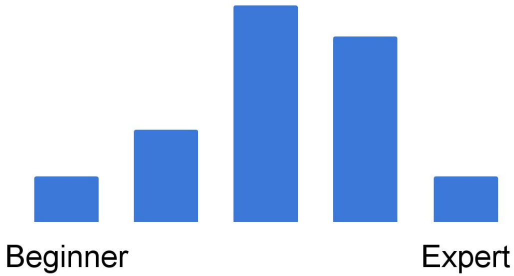
Linear Algebra



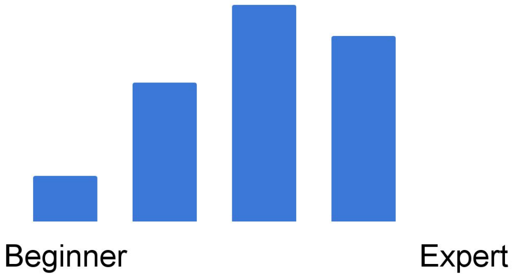
Machine Learning



Python and its ML packages



Natural Language Processing



Anonymized quotes from feedback section

"*Some **context on evolution of NLP** will be super helpful*"

Anonymized quotes from feedback section

*"Some **context on evolution of NLP** will be super helpful"*

*"Start from **medium level** and then go upwards to **higher difficulty**"*

Anonymized quotes from feedback section

*"Some **context on evolution of NLP** will be super helpful"*

*"Start from **medium level** and then go upwards to **higher difficulty**"*

*"Hope to get the **summary of the materials** (including additional articles/books) and **links to them** to have better understanding. Hope to **try BERT in practice** (in Python notebooks). [...]"*

*I am very interested in **hands on** experience. I hope this session will help to start running my first deep learning model.*

*"[...] My hope is we **dig into the code** and the **details** of running a program"*

*"Among the given topics, I am more interested in the application areas of **sentiment extraction**"*

Logistics

- **Two half-days**
  - Wednesday 8/18, 8am - 11am PT
  - Friday 8/20, 8am - 11am PT
- **Hands-on** format
  - ~2/3 slides
  - ~1/3 code via Colab
- **Questions**
  - Preferably ask questions via Ed
  - Pause from time to time to answer questions
  - After each break, dedicated time for Q&A
- **Homework** between the 2 days
  - Apply concepts in a practical use case
  - Completely optional, but recommended

Tentative schedule for today

8:00

Intro and background

9:00 BREAK

Questions

Transformers deep dive

10:00 BREAK

Questions

Hands-on practice

11:00

All times are in Pacific Time (UTC-7)

Tentative schedule for today

8:00

Intro and background

9:00

BREAK
Questions

Transformers deep dive

10:00

BREAK
Questions

Hands-on practice

11:00

All times are in Pacific Time (UTC-7)

Tentative schedule for today

8:00

Intro and background

BREAK

9:00

Questions

Transformers deep dive

10:00

BREAK

Questions

Hands-on practice

11:00

All times are in Pacific Time (UTC-7)

Tentative schedule for today

8:00

Intro and background

9:00

BREAK

Questions

Transformers deep dive

10:00

BREAK

Questions

Hands-on practice

11:00

All times are in Pacific Time (UTC-7)

Tentative schedule for today

8:00

Intro and background

BREAK

9:00

Questions

Transformers deep dive

BREAK

10:00

Questions

Hands-on practice

11:00

All times are in Pacific Time (UTC-7)

Deep Learning for NLP, part II

Stanford ICME Summer workshop 2021

Motivation and setup

Background

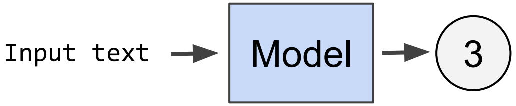
Transformers

BERT

Conclusion

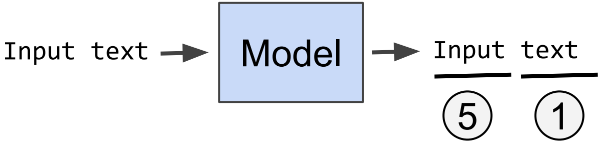
NLP tasks overview

Classification



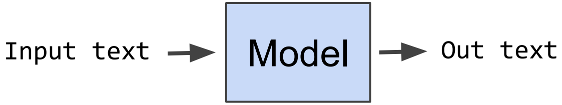
- Sentiment extraction
- Intent detection
- Language detection
- Topic modeling

"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

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

Datasets

Amazon reviews

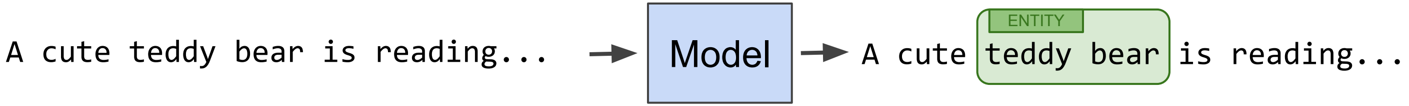
9.3 IMDB critiques

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



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`

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

Standardized benchmark for NLP

GLUE: **G**eneral **L**anguage **U**nderstanding **E**valuation

Grammatical correctness *CoLA*	Paraphrase *MRPC*	Similarity *QQP, STS-B*	Common sense *WNLI*
Entailment *RTE, MNLI*	Sentiment Extraction *SST-2*	Question Answering *QNLI*	

Glue score

More information about the GLUE tasks can be found here: https://gluebenchmark.com/tasks

Disclaimer before starting: many abbreviations....

BERT CoNLL BLEU ELMo BPE WNLI MLM

MRPC

GPT EM LSTM PoS

T5

QA

ROUGE GLUE WMT NER

F1 GloVe PPL mT5 C4 MT ACL

GRU SQuAD WP EMNLP SP NLG METEOR

...but don't worry!

BERT, DistilBERT, ALBERT, T5, mT5, GPT

Transformer-based models

LSTM, GRU, GloVe, ELMo, BPE, WP, SP

Some techniques

ACL, EMNLP, WMT, CoNLL

Conferences

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

Tasks

MNLI, WNLI, C4, SQuAD, GLUE, MRPC

Datasets

F1, PPL, ROUGE, BLEU, METEOR, EM

Metrics

Deep Learning for NLP, part II

Stanford ICME Summer workshop 2021

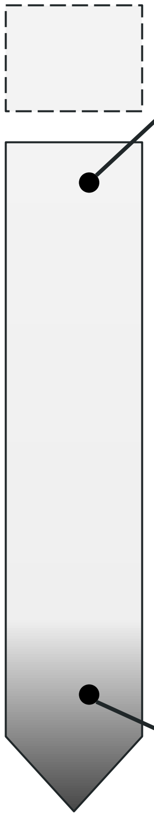
High-level timeline

1980s Recurrent neural networks (RNNs)

1997 Long short-term memory (LSTM)

2013 Word2vec

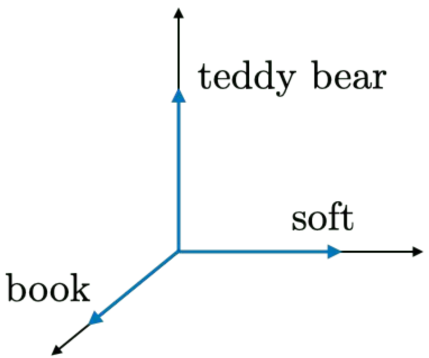
2017 Transformers



Word representations

Motivation

Naive (one-hot) encoding



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

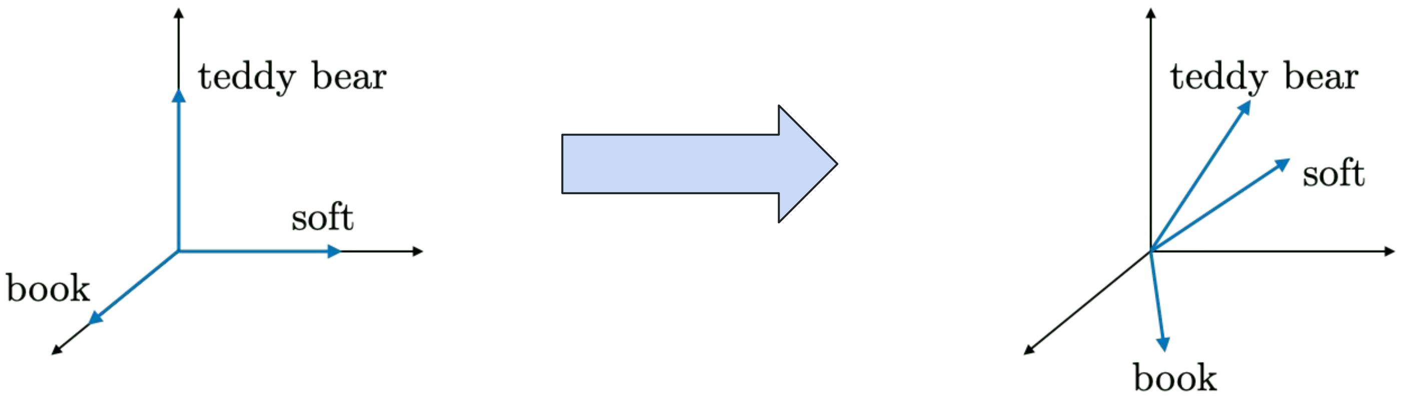
Figure adapted from "VIP cheatsheets for Stanford's CS 230", Amidi. *stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks*

Word representations

Motivation

Naive (one-hot) encoding

Learned embedding



$$\text{soft} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \begin{array}{c} \langle \text{teddy bear},\ \text{book} \rangle = 0 \\ \langle \text{teddy bear},\ \text{soft} \rangle = 0 \end{array}$$

$$\text{soft} = \begin{pmatrix} 0.95 \\ 0.32 \\ 0.01 \end{pmatrix} \quad \begin{array}{c} \langle \text{teddy bear},\ \text{book} \rangle \sim 0 \\ \langle \text{teddy bear},\ \text{soft} \rangle \sim 1 \end{array}$$

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.*

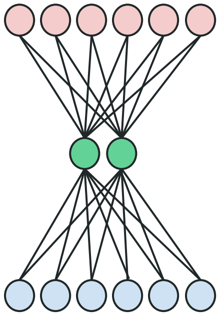
Word2vec

Architecture

output

hidden

input



size V

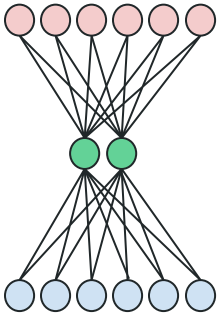
size d

size V

Word2vec

Example with left context window = 1

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

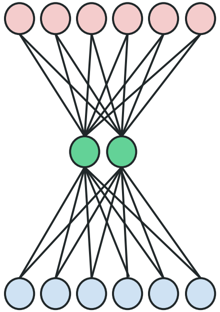


$\boxed{\texttt{A}}$ cute teddy bear is reading

Word2vec

Example with left context window = 1

A cute teddy bear is reading



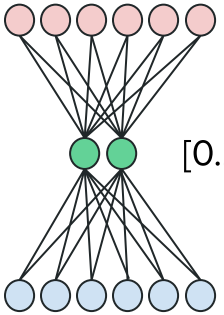
[1,0,0,0,0,0]

A cute teddy bear is reading

Word2vec

Example with left context window = 1

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



[0.2, 0.9]

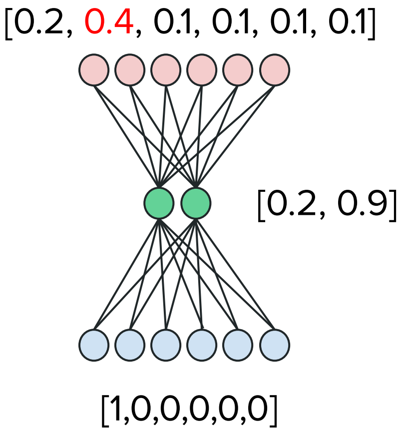
[1,0,0,0,0,0]

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

Word2vec

Example with left context window = 1

A **cute** teddy bear is reading

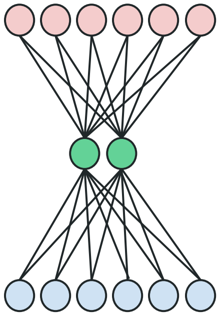


A cute teddy bear is reading

Word2vec

Example with left context window = 1

A cute teddy bear is reading

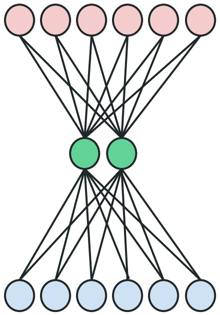


A cute teddy bear is reading

Word2vec

Example with left context window = 1

A cute **teddy bear** is reading



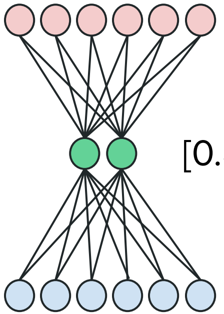
[0,1,0,0,0,0]

A **cute** teddy bear is reading

Word2vec

Example with left context window = 1

A cute teddy bear is reading



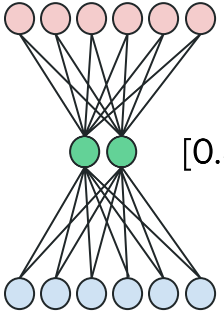
A cute teddy bear is reading

Word2vec

Example with left context window = 1

A cute teddy bear is reading

[0.2, 0.2, 0.1, 0.1, 0.2, 0.1]



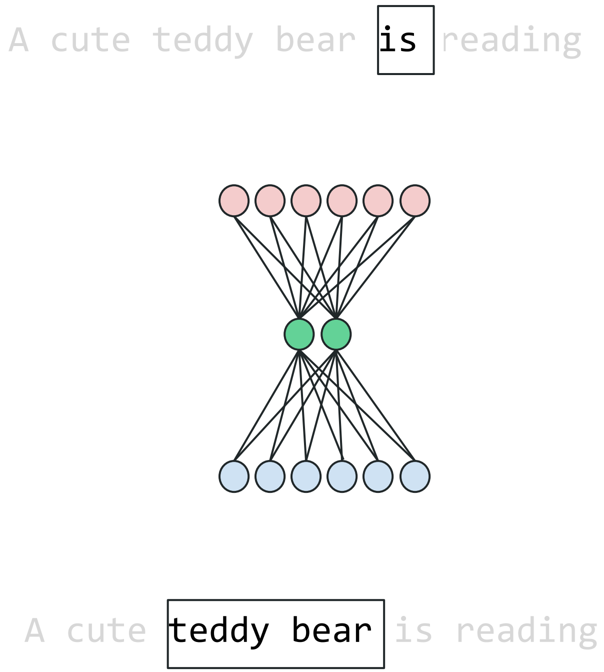
[0.8, 0.4]

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

A cute teddy bear is reading

Word2vec

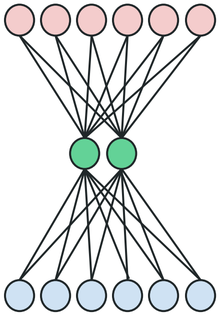
Example with left context window = 1



Word2vec

Example with left context window = 1

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

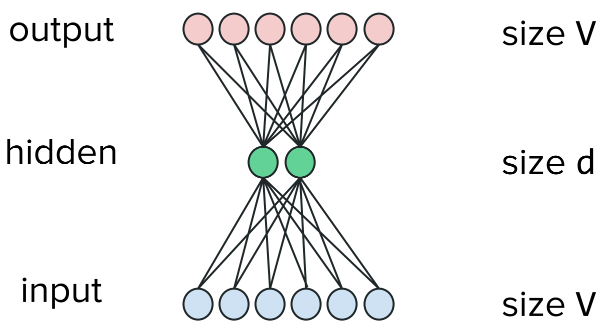


A cute teddy bear $\boxed{\text{is}}$ reading

Word2vec

Example with left context window = 1

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



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

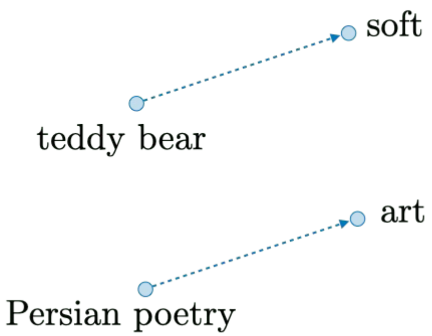


Figure adapted from “VIP cheatsheets for Stanford’s CS 230”, Amidi. *stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks*

Recurrent Neural Networks (RNNs)

Overview

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

General form

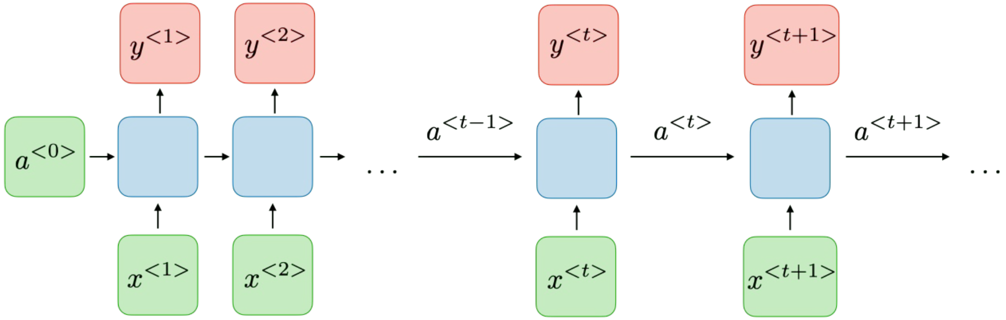
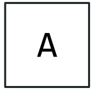
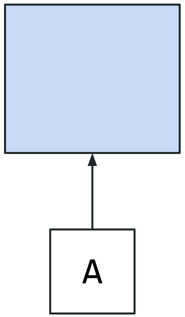


Figure adapted from "VIP cheatsheets for Stanford's CS 230", Amidi. stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

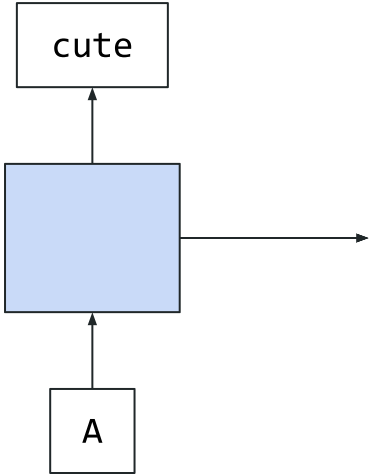
Recurrent Neural Networks (RNNs)



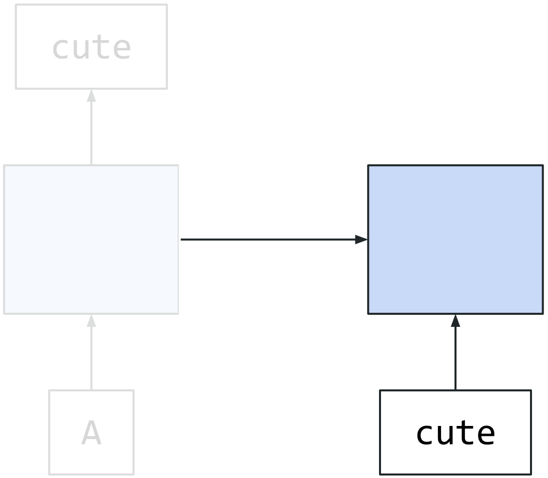
Recurrent Neural Networks (RNNs)



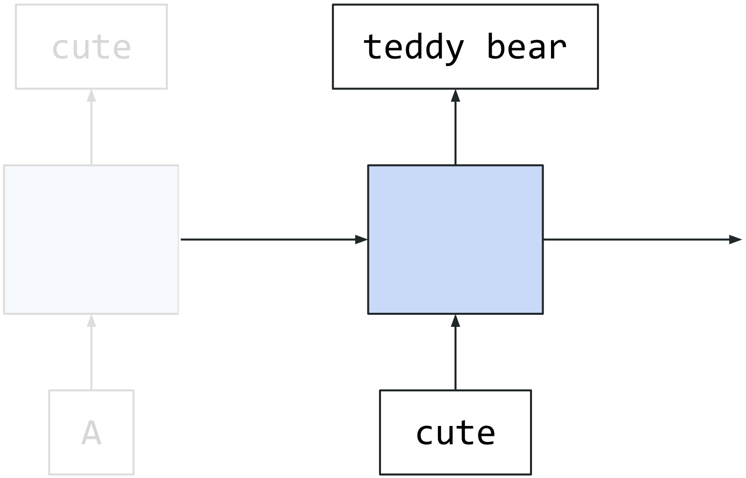
Recurrent Neural Networks (RNNs)



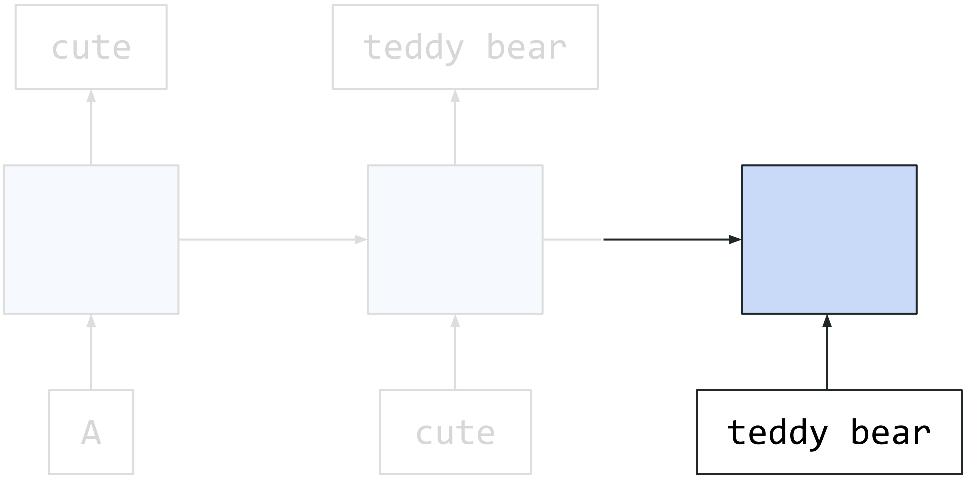
Recurrent Neural Networks (RNNs)



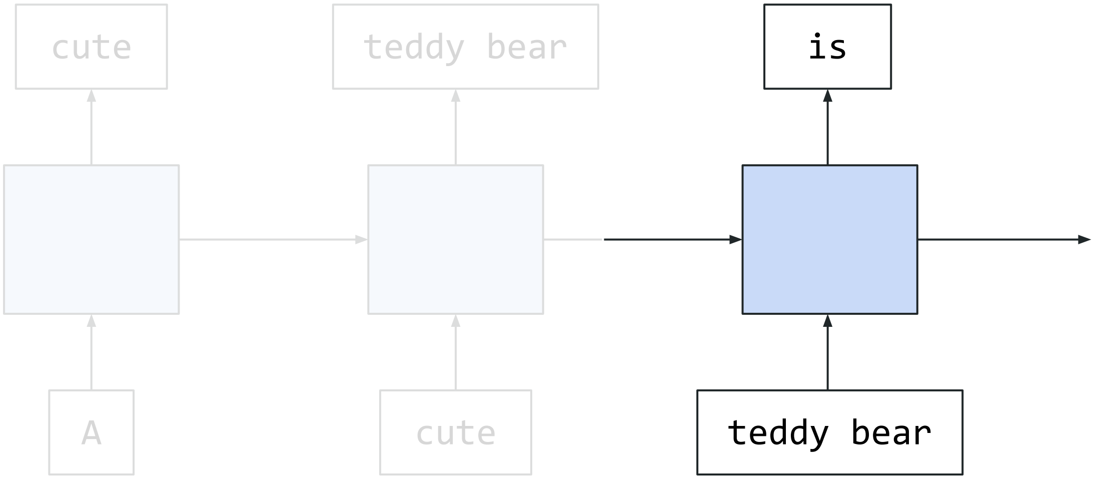
Recurrent Neural Networks (RNNs)



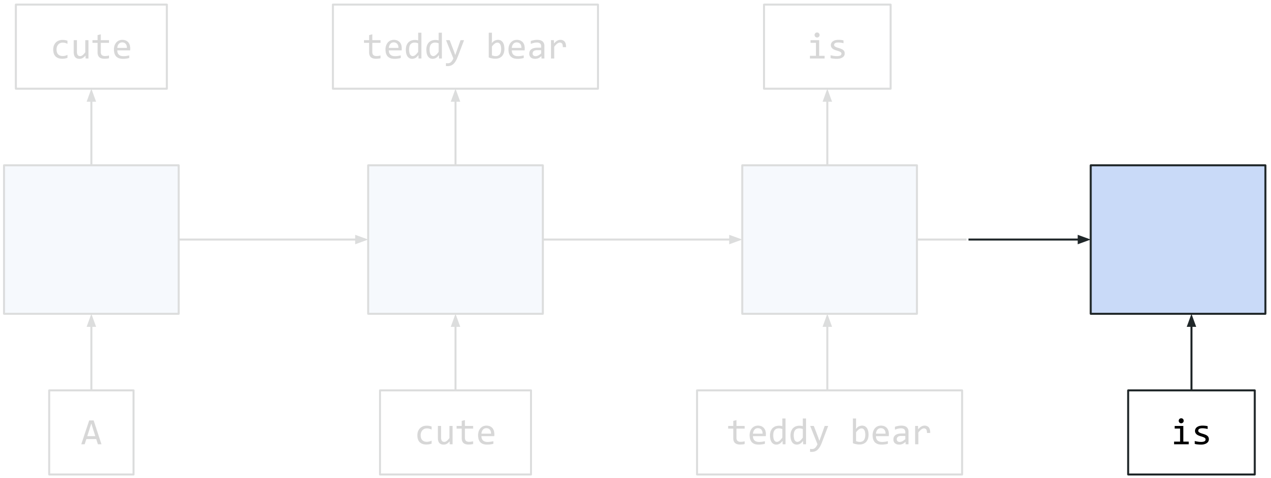
Recurrent Neural Networks (RNNs)



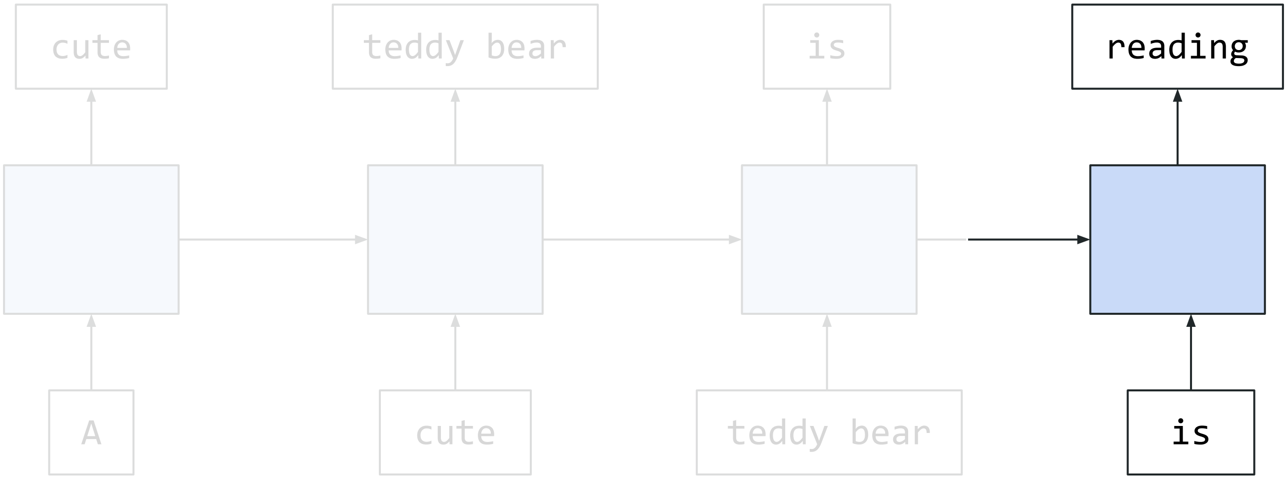
Recurrent Neural Networks (RNNs)



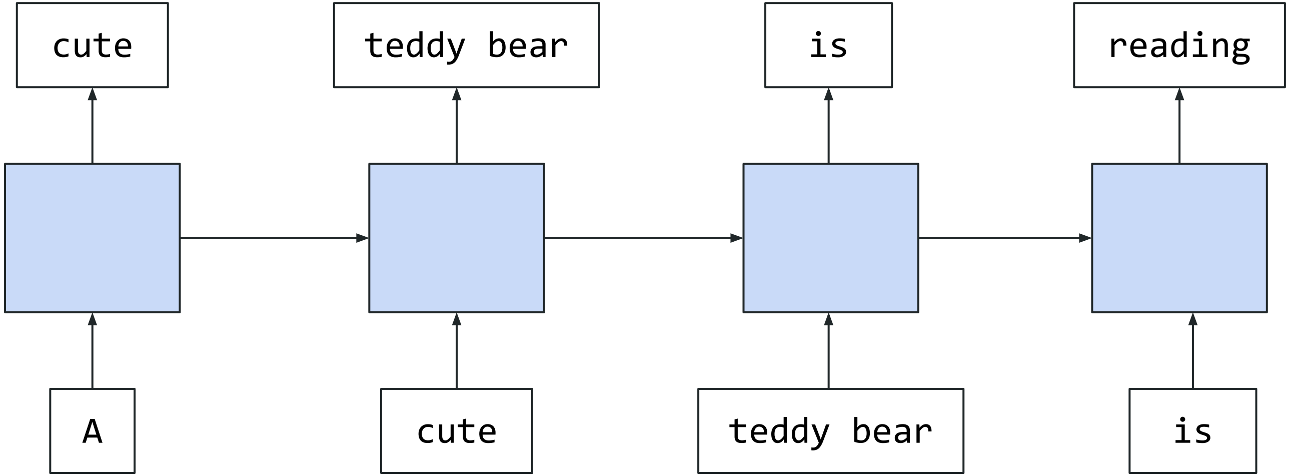
Recurrent Neural Networks (RNNs)



Recurrent Neural Networks (RNNs)

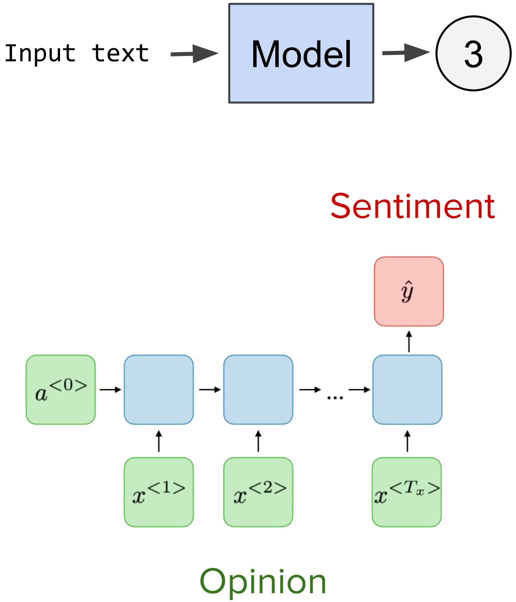


Recurrent Neural Networks (RNNs)

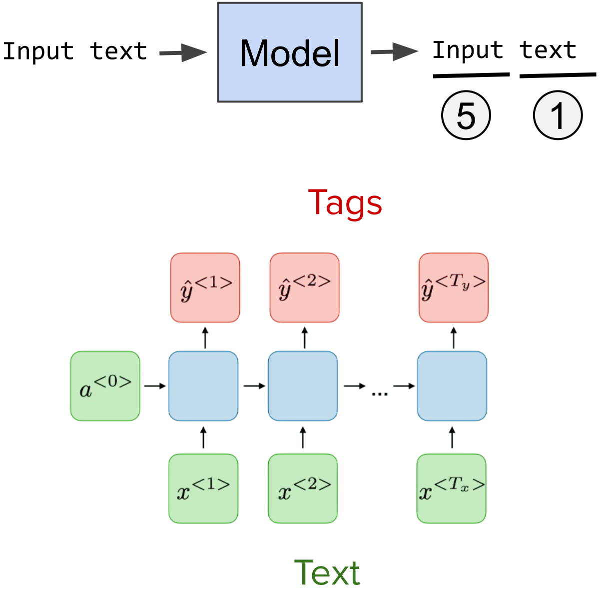


Recurrent Neural Networks (RNNs)

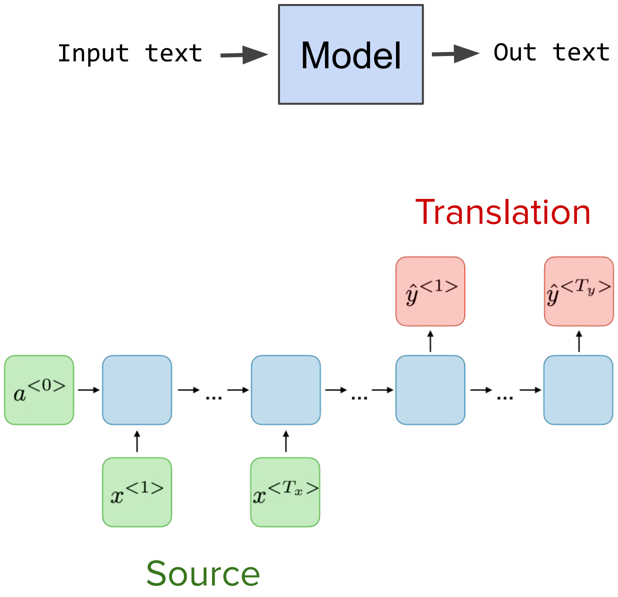
Classification



"Multi"-classification



Generation



Figures 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

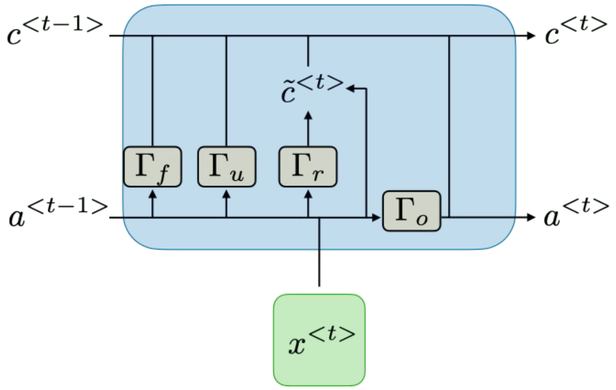
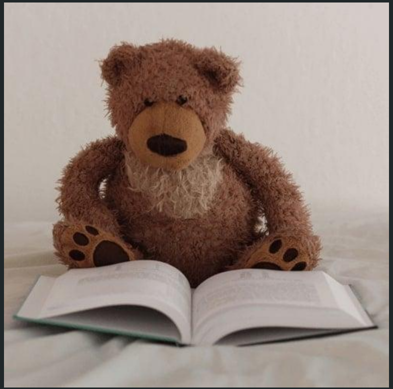


Figure adapted from "VIP cheatsheets for Stanford's CS 230", Amidi. stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

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 ● Embeddings not context aware ● Slow computations

Break + questions



Deep Learning for NLP, part II

Stanford ICME Summer workshop 2021

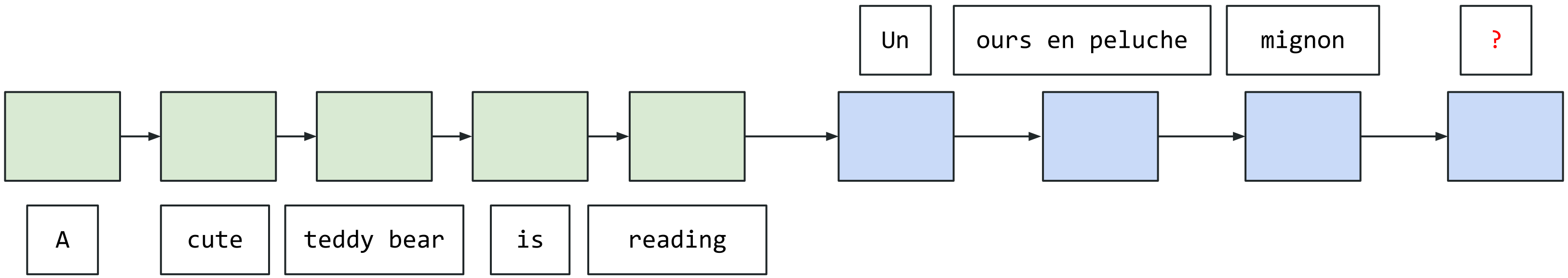
History of attention

- 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

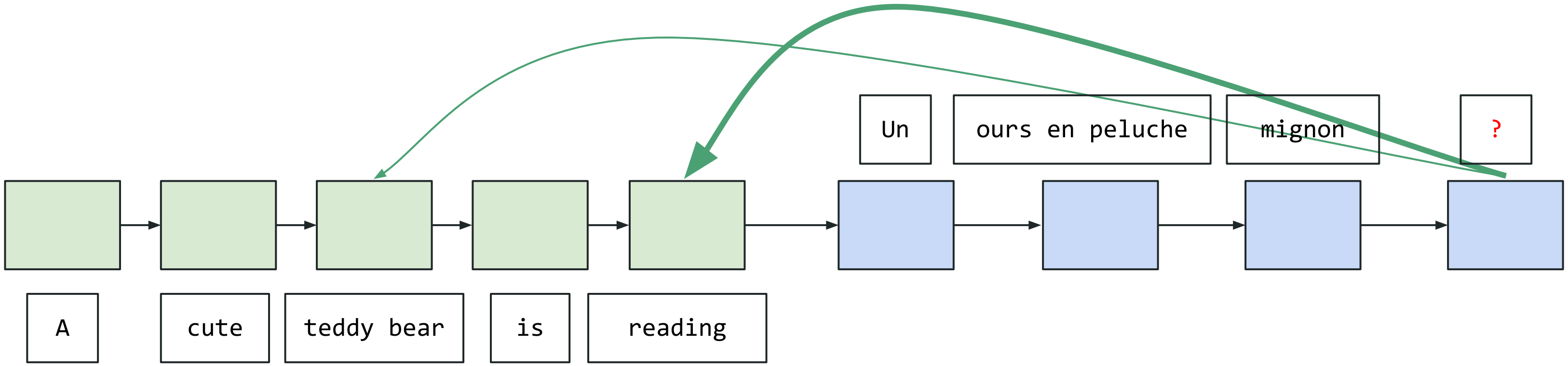
- Introduced in 2014
- Translation tasks had a real issue with long-term dependencies
- Seq2seq unable to "remember" what input sentence was saying



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

History of attention

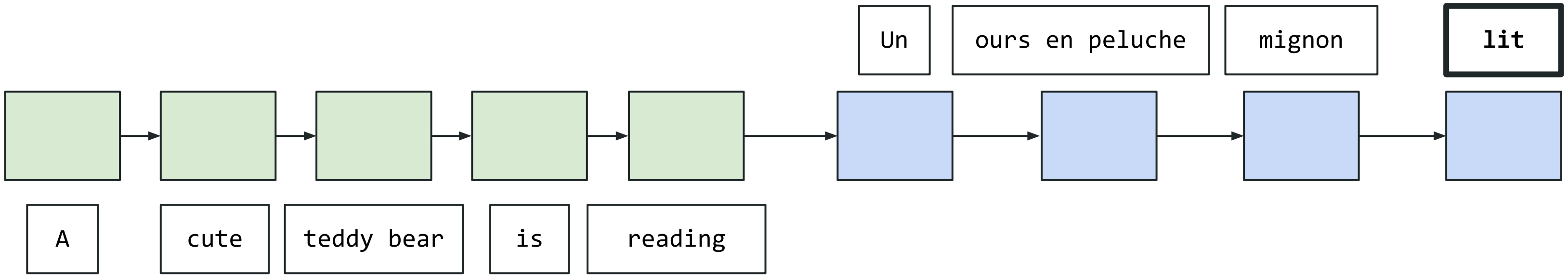
- Introduced in 2014
- Translation tasks had a real issue with long-term dependencies
- Seq2seq unable to "remember" what input sentence was saying



"*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



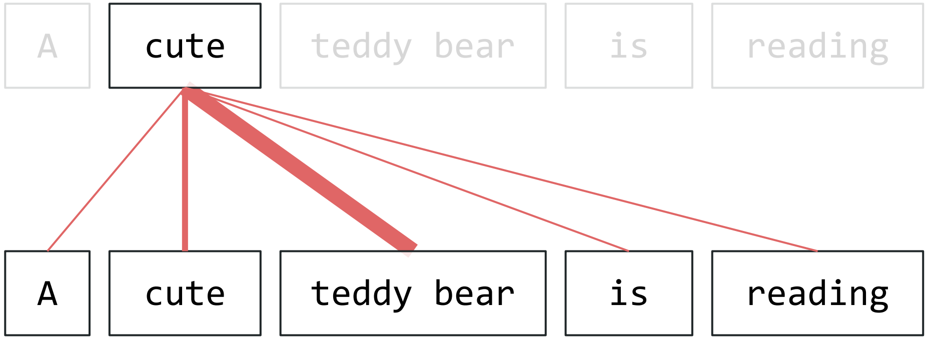
"*Neural Machine Translation by Jointly Learning to Align and Translate*", *Bahdanau et al., 2014.*

Overview of the Transformer

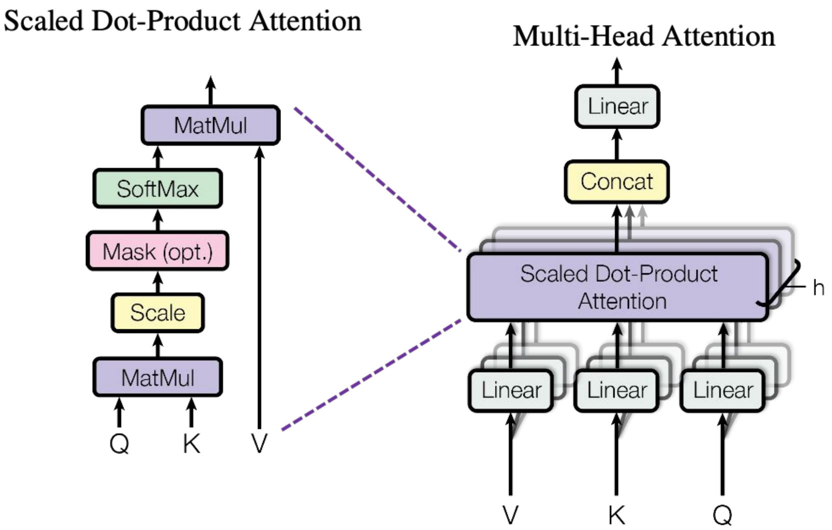
- Introduced in the 2017 paper "Attention is All You Need"
- Relies on the self-attention mechanism
- Encoder/decoder parts that are used in a lot of models
- State of the art results on machine translation tasks

Overview of the Transformer

- Introduced in the 2017 paper "Attention is All You Need"
- Relies on the self-attention mechanism
- Encoder/decoder parts that are used in a lot of models
- State of the art results on machine translation tasks



Attention mechanism



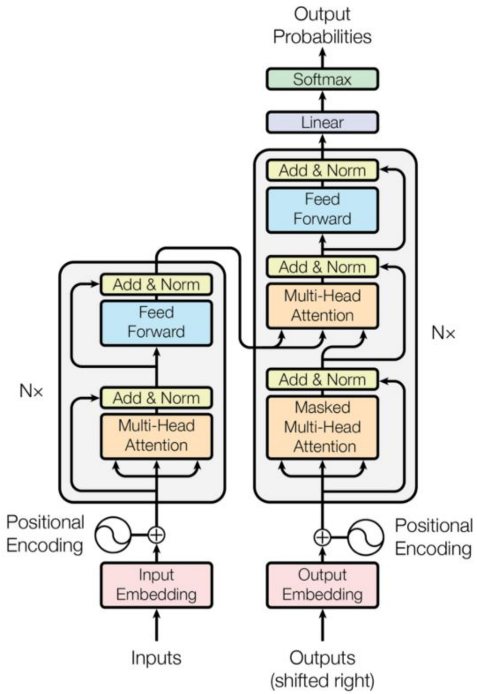
- **Q**uery, **K**ey, **V**alue

- Computationally efficient with matrices

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

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

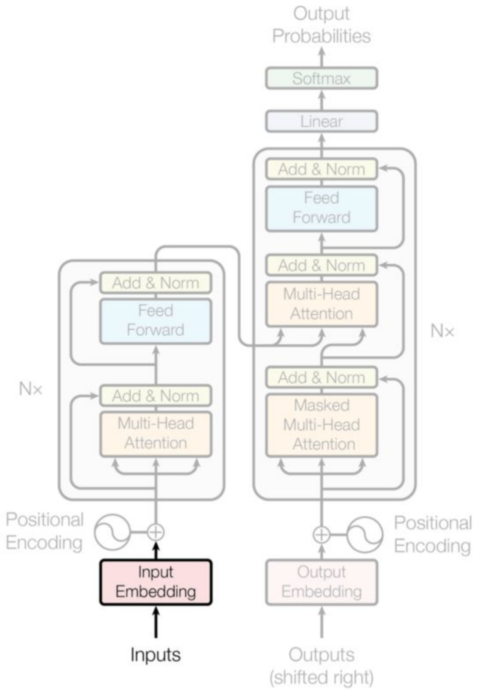
Transformer architecture



- **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

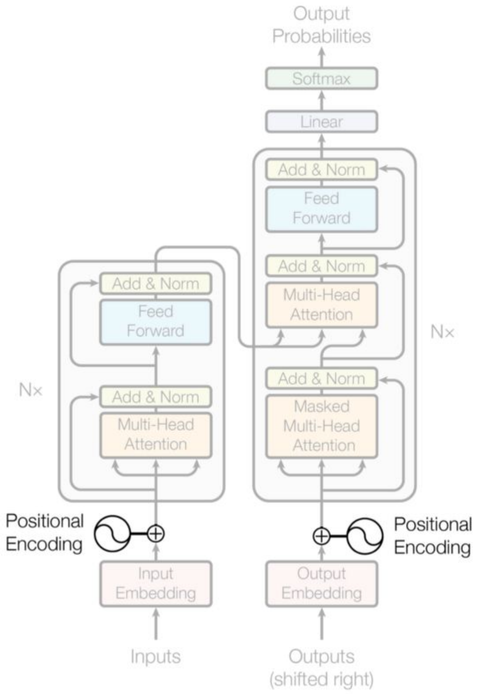
- Text is "tokenized"
- Learned embeddings for tokens

Parameters

- V: vocabulary size
- d_model: embedding dimensions

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

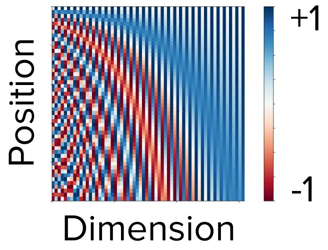
... with a trick!

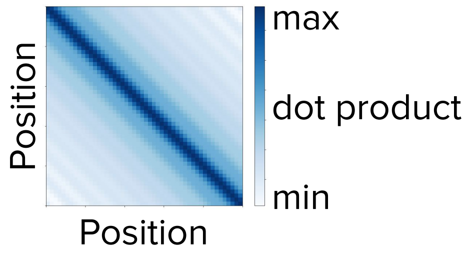


Positional encoding

Idea:

- From a convolutional seq2seq 2017 paper
- Add **position information** to inputs
- Can be either learned or hardcoded

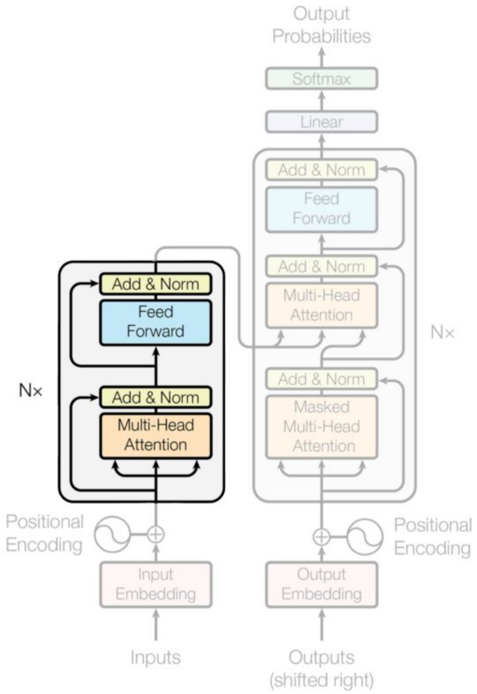




Goal: let model understand relative input position

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

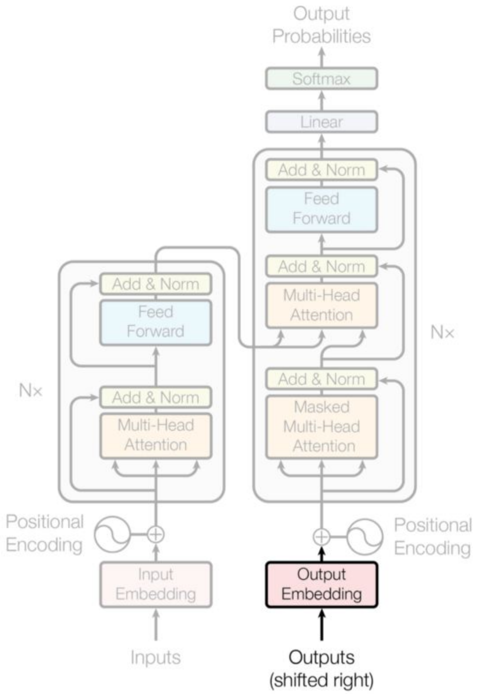
- Encoder-Encoder attention / self-attention
- Feed Forward Neural Network
- Normalization layer

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

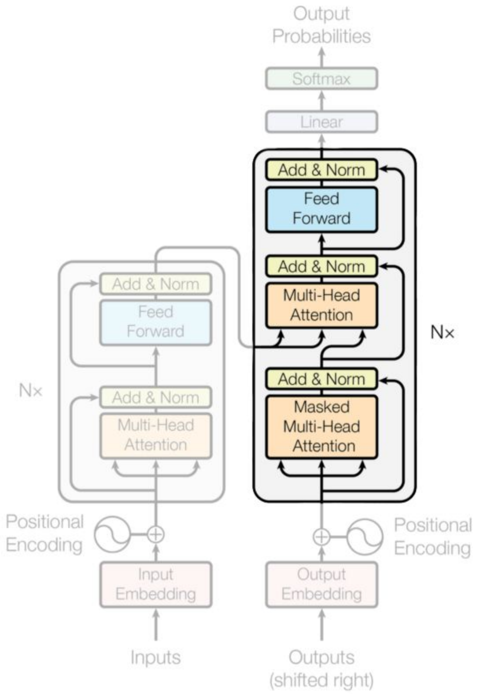
- Learned embeddings for output tokens
- In practice, will start with `[BOS]` during translation

Parameters

- V: vocabulary size
- d_model: embedding dimensions

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

Decoder



Overview

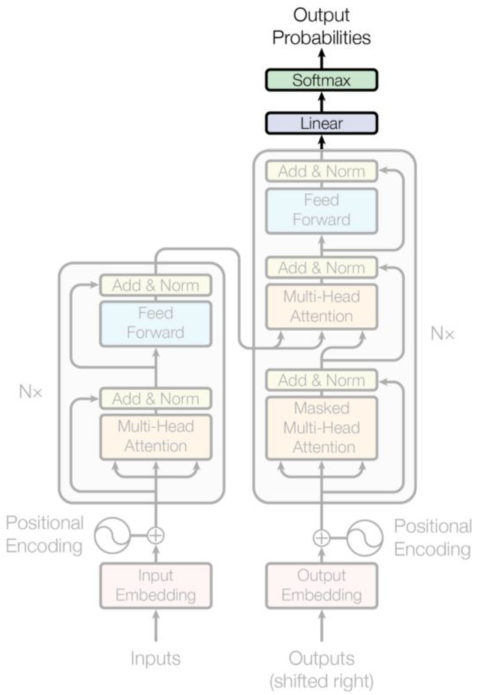
- Decoder-Decoder attention / self-attention
- Encoder-Decoder attention
- Feed Forward Neural Network
- Normalization layer

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.

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

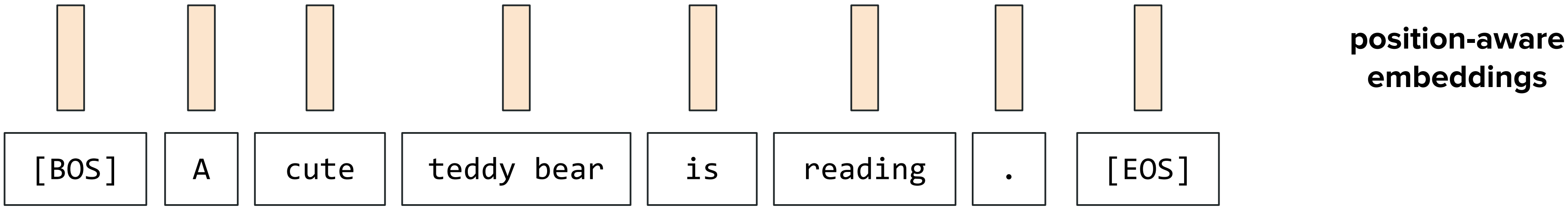
[BOS]	A	cute	teddy bear	is	reading	.	[EOS]

Stitching all the pieces together with an example

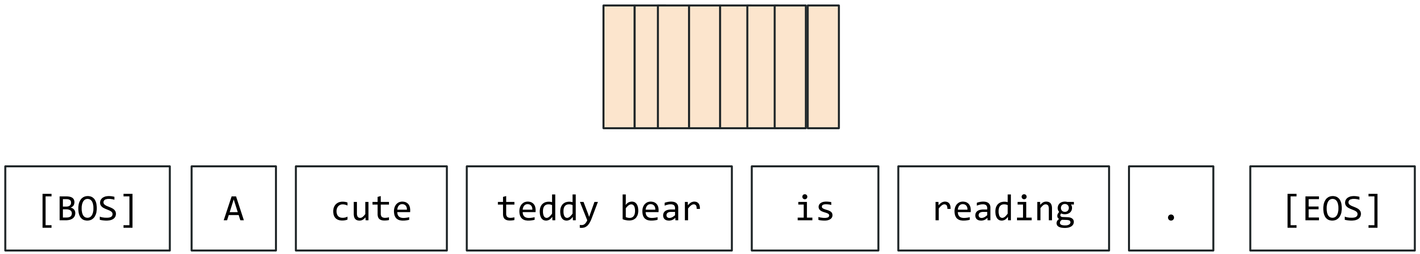
Stitching all the pieces together with an example

Stitching all the pieces together with an example

Stitching all the pieces together with an example

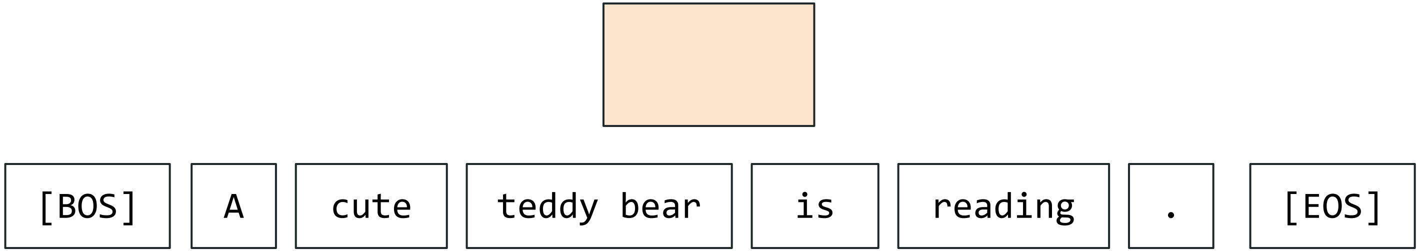


Stitching all the pieces together with an example



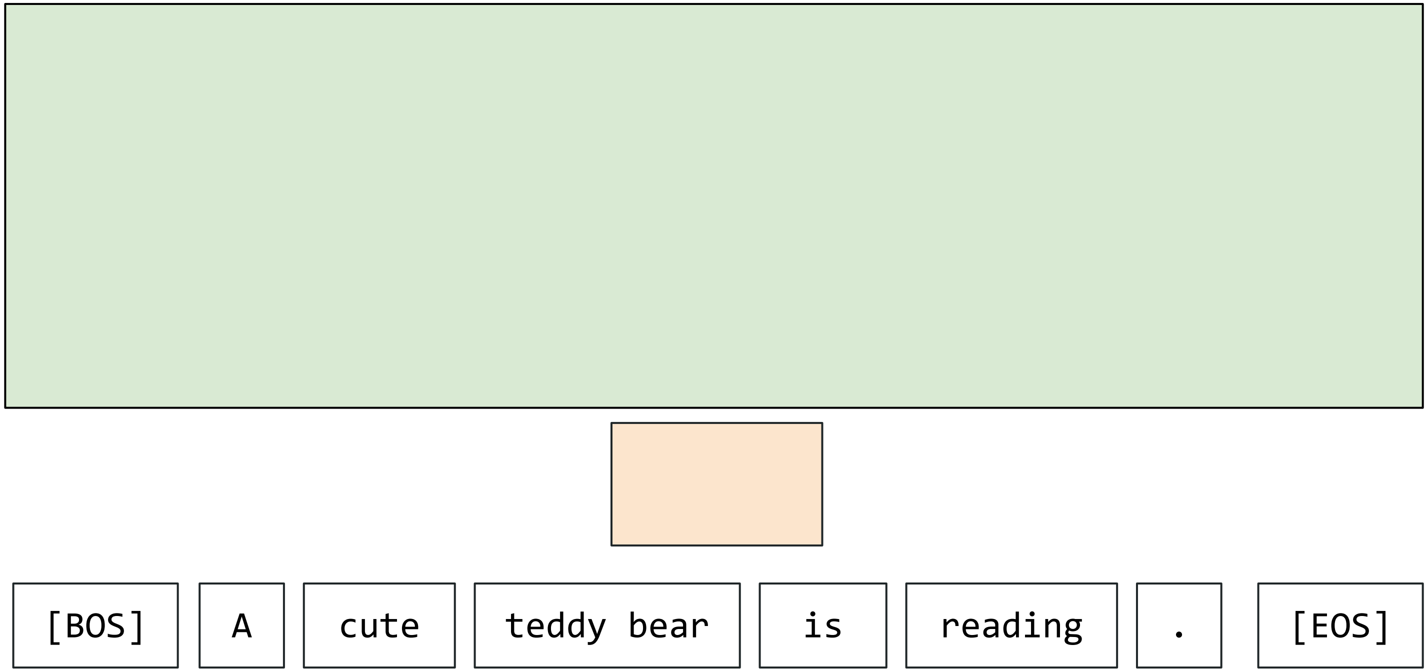
position-aware embeddings matrix

Stitching all the pieces together with an example



position-aware embeddings matrix

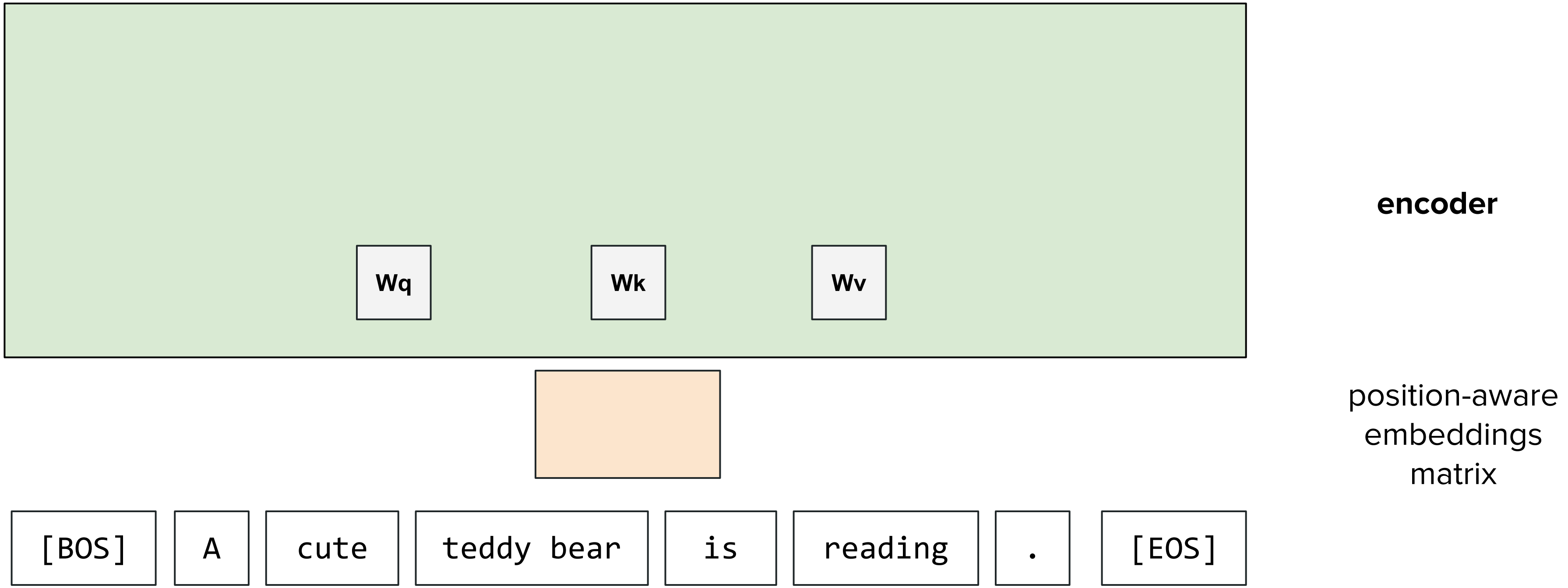
Stitching all the pieces together with an example



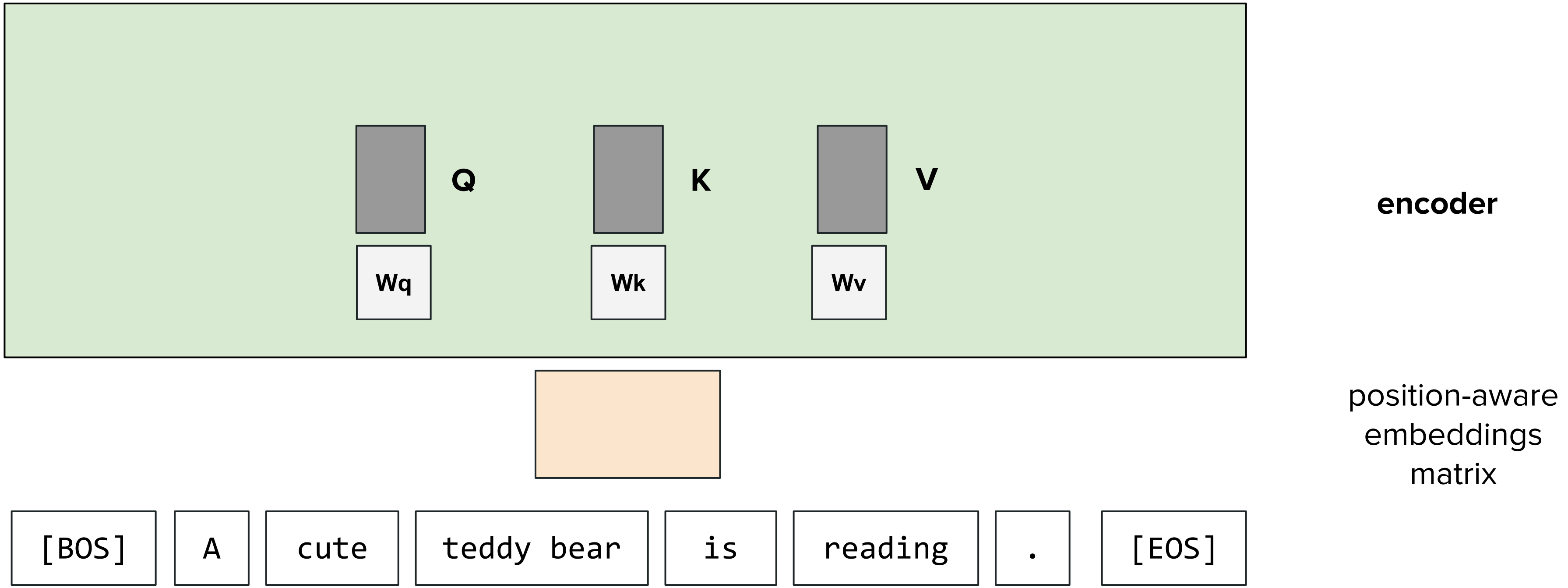
encoder

position-aware
embeddings matrix

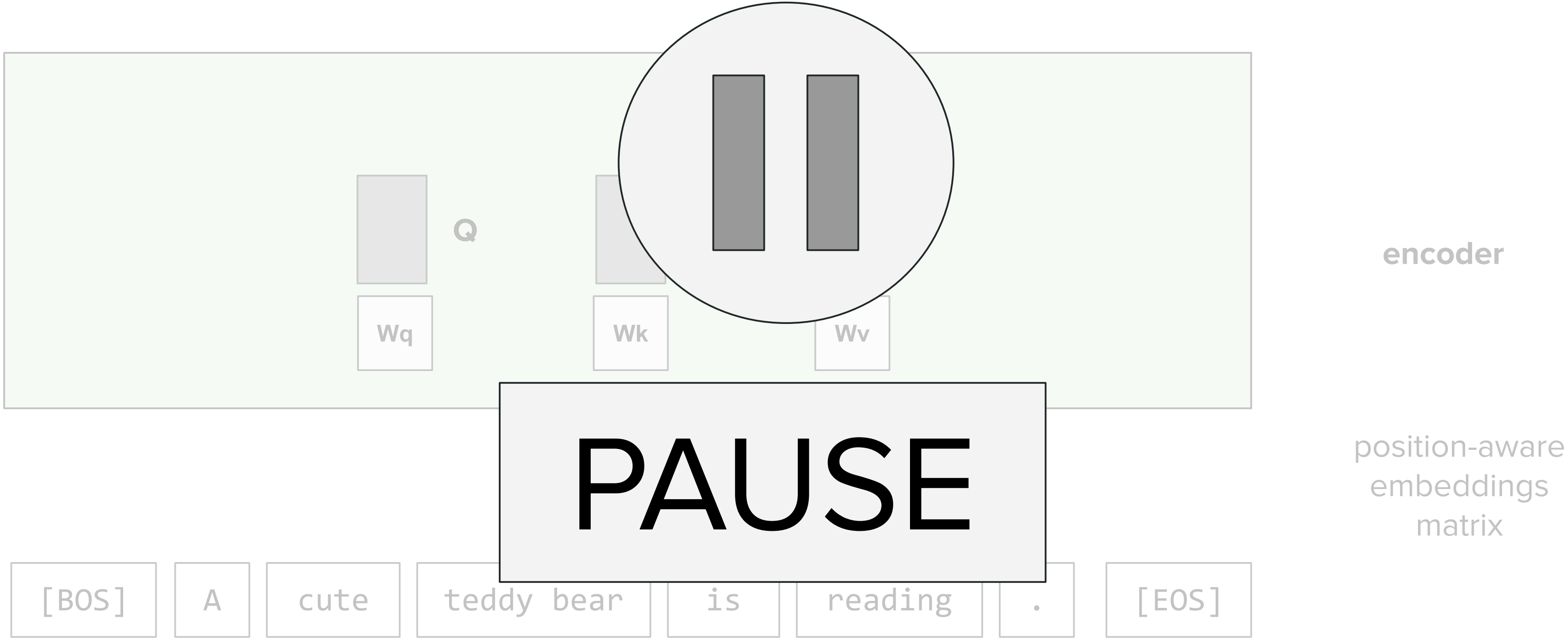
Stitching all the pieces together with an example



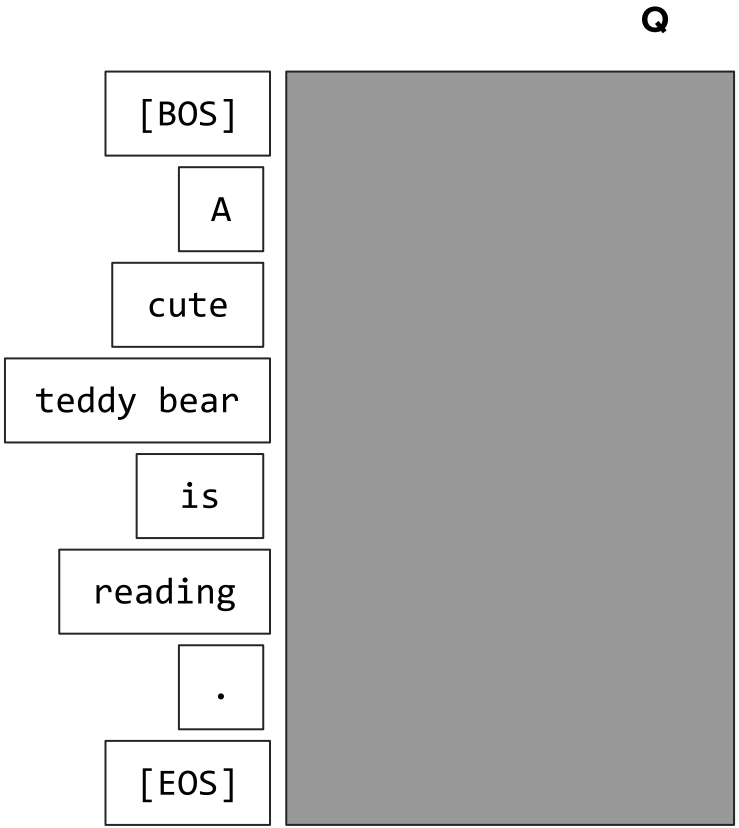
Stitching all the pieces together with an example



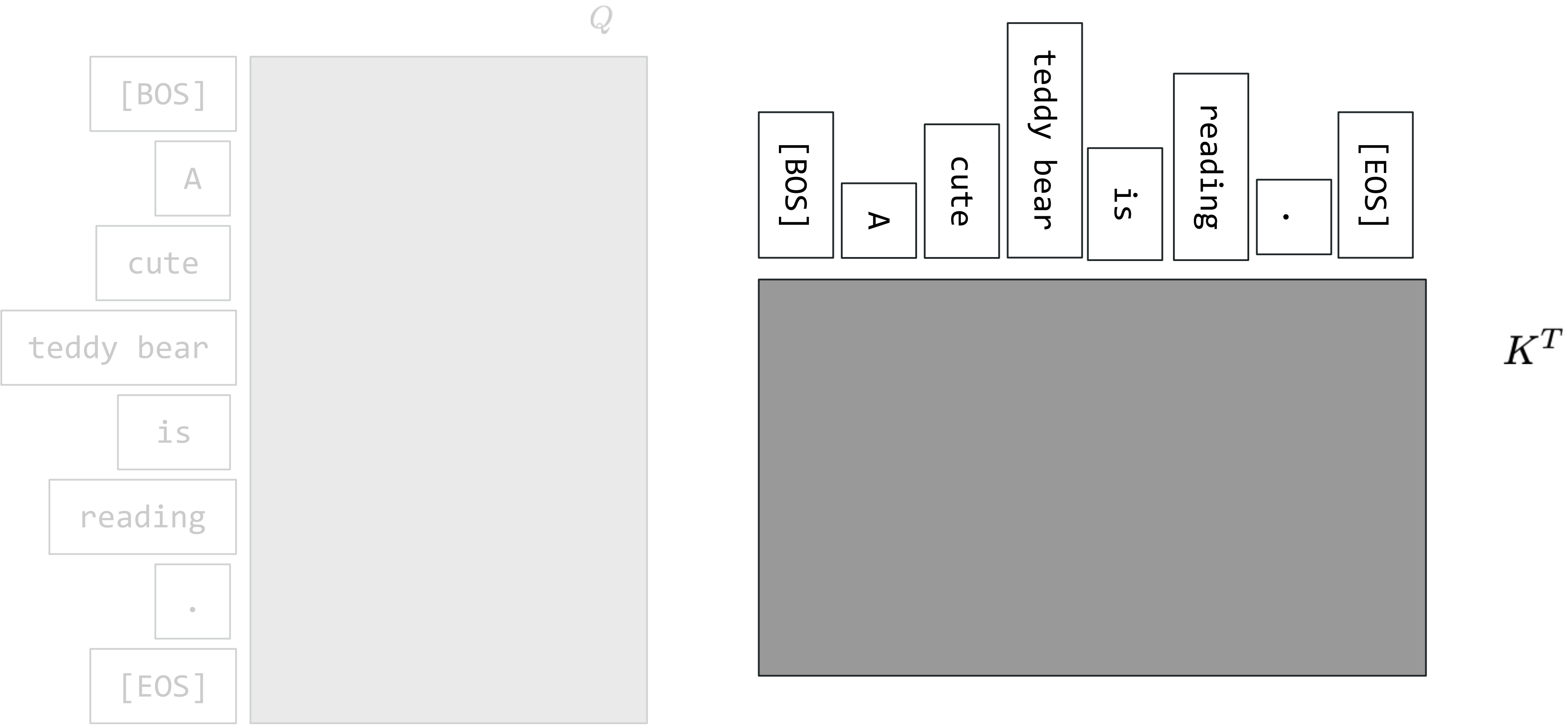
Stitching all the pieces together with an example



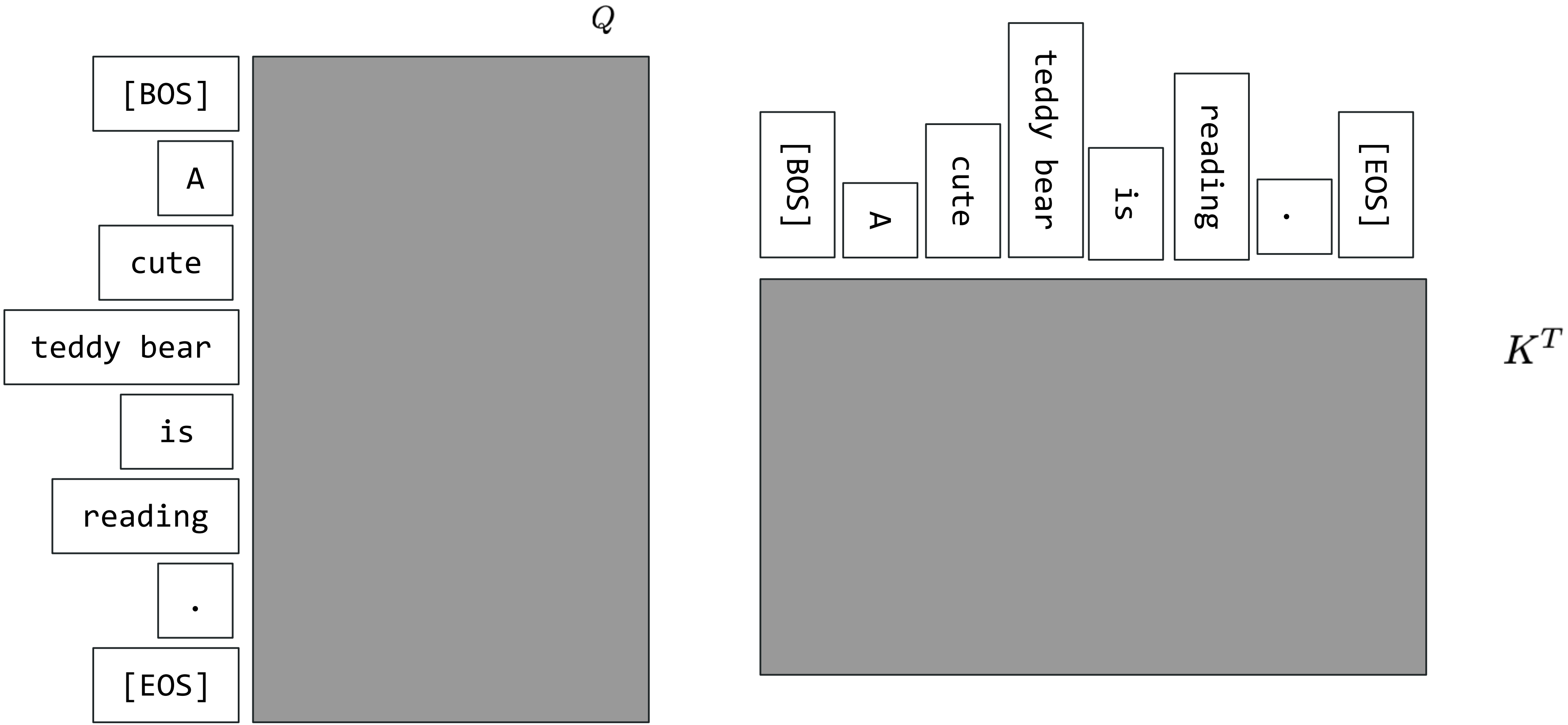
Stitching all the pieces together with an example



Stitching all the pieces together with an example



Stitching all the pieces together with an example



Stitching all the pieces together with an example

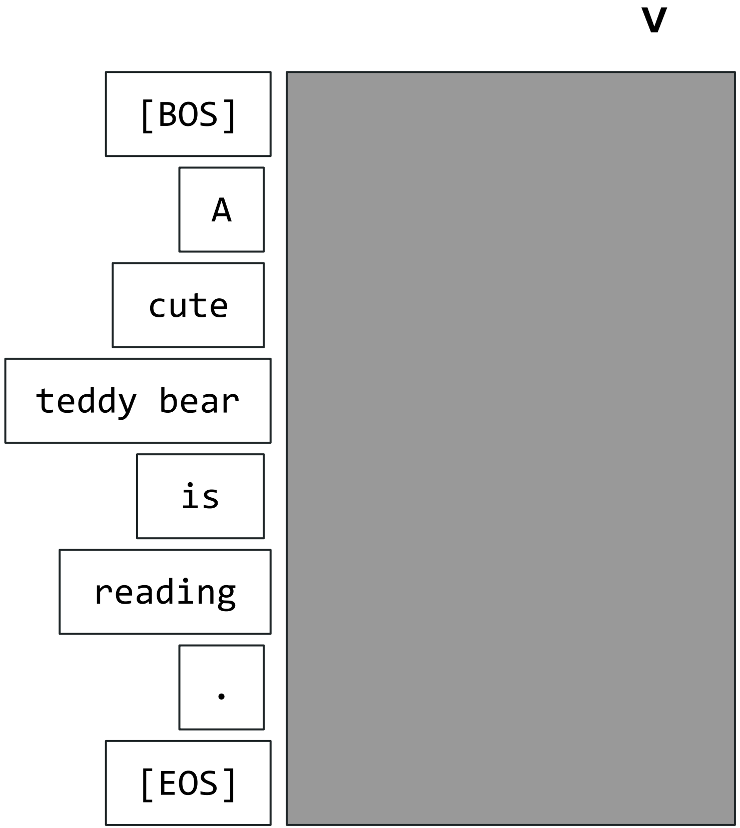
$$
\begin{matrix}
\langle q_{\text{[BOS]}}, k_{\text{[BOS]}} \rangle & \langle q_{\text{[BOS]}}, k_{\text{A}} \rangle & \langle q_{\text{[BOS]}}, k_{\text{cute}} \rangle & \cdots \\
\langle q_{\text{A}}, k_{\text{[BOS]}} \rangle & \langle q_{\text{A}}, k_{\text{A}} \rangle & & \\
\langle q_{\text{cute}},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

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

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

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

$$\vdots$$

$$QK^TV$$

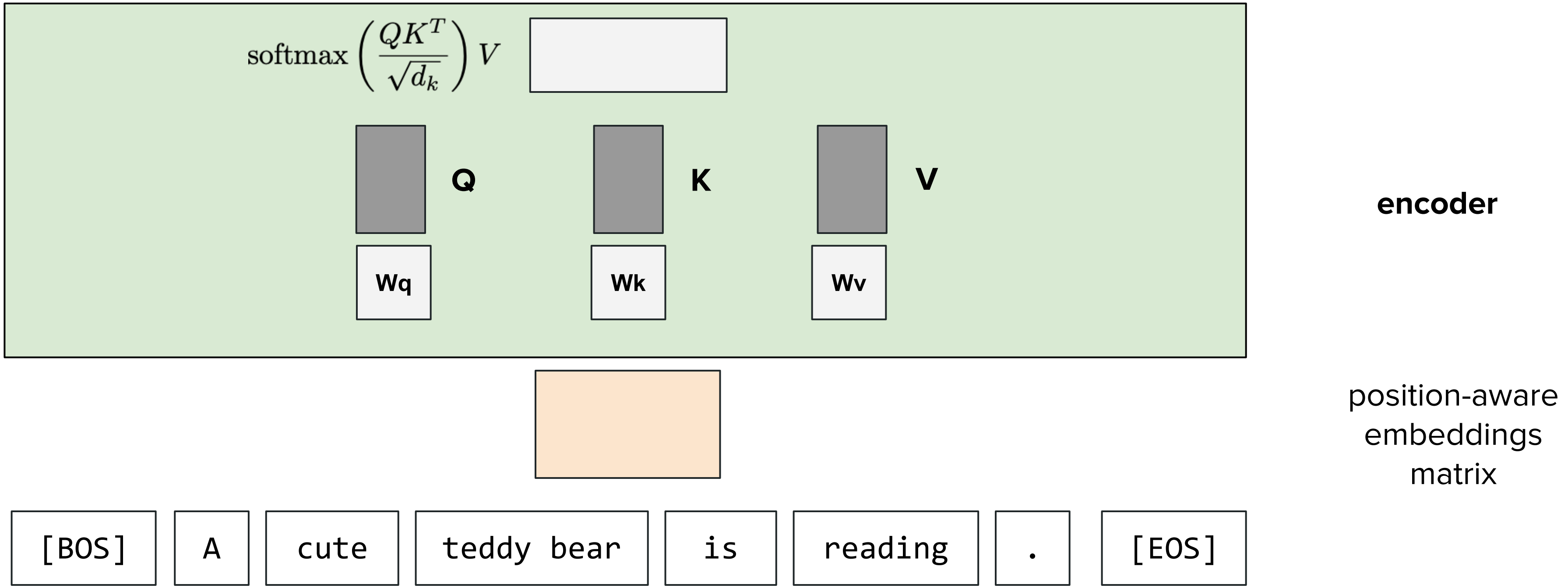
Stitching all the pieces together with an example

weighted average of values

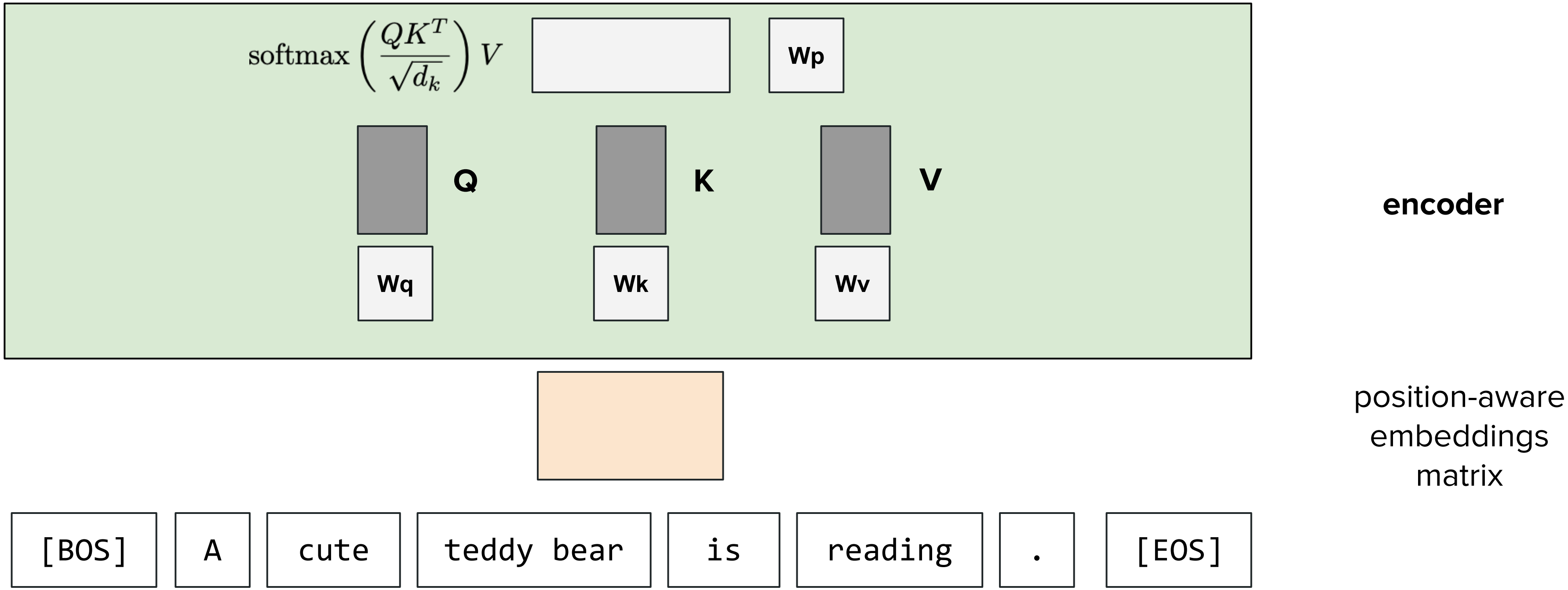
with weights being a function of $\langle q, k \rangle$

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

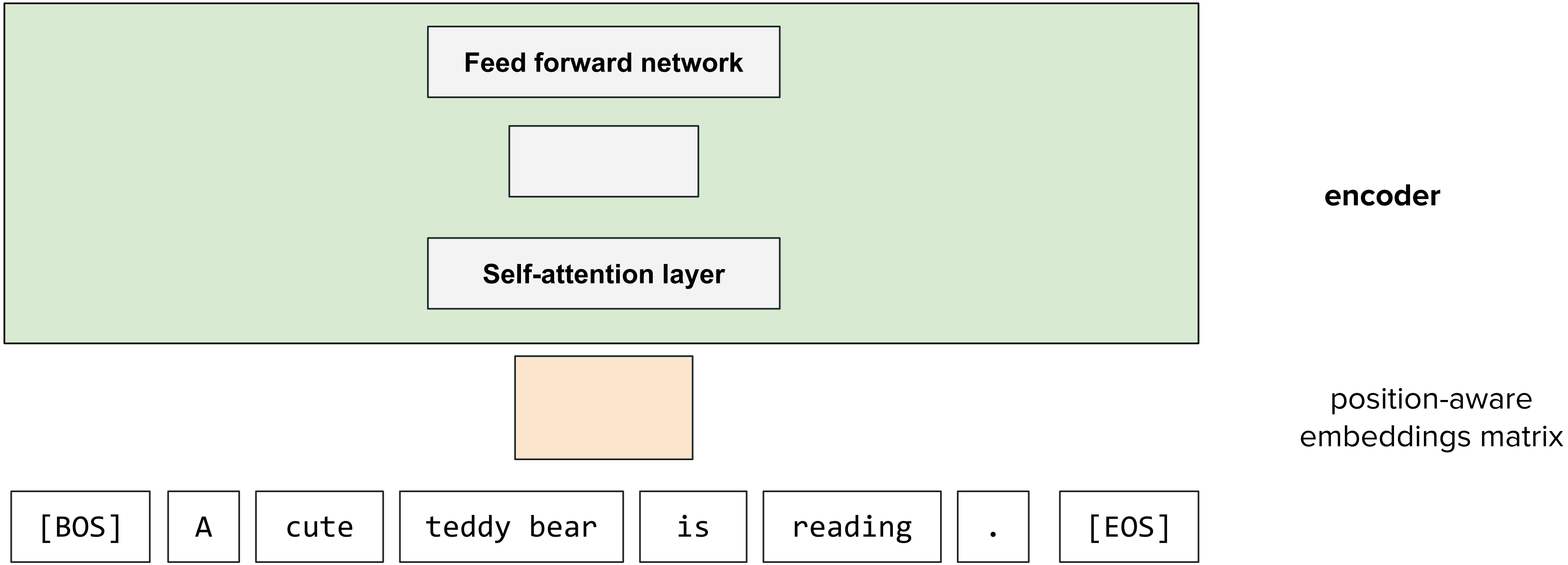
Stitching all the pieces together with an example



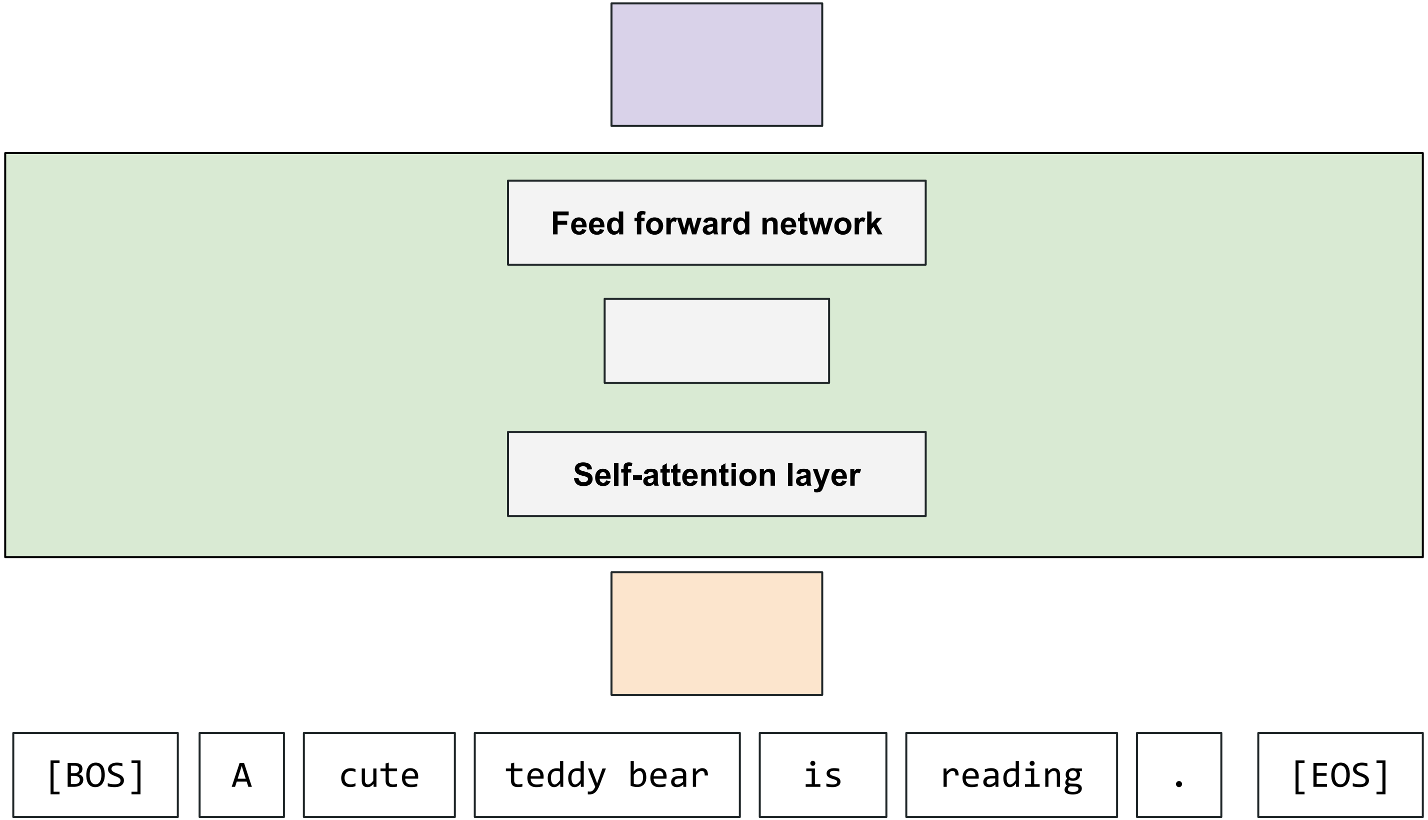
Stitching all the pieces together with an example



Stitching all the pieces together with an example



Stitching all the pieces together with an example

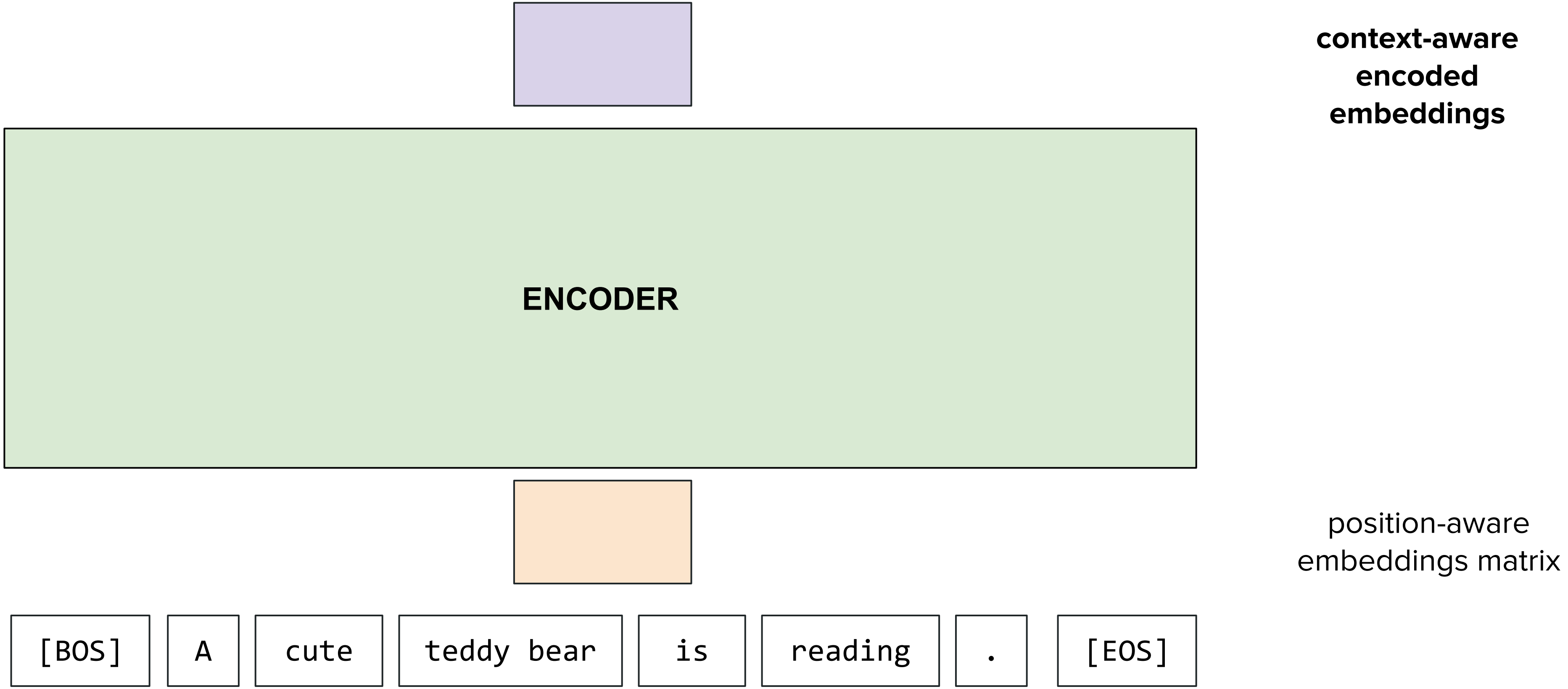


context-aware encoded embeddings

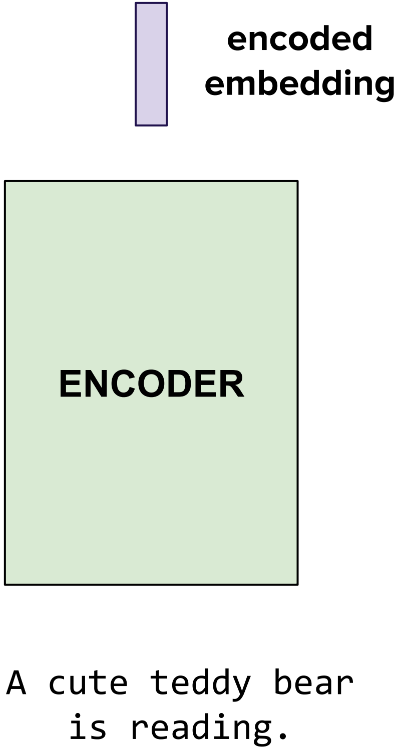
encoder

position-aware embeddings matrix

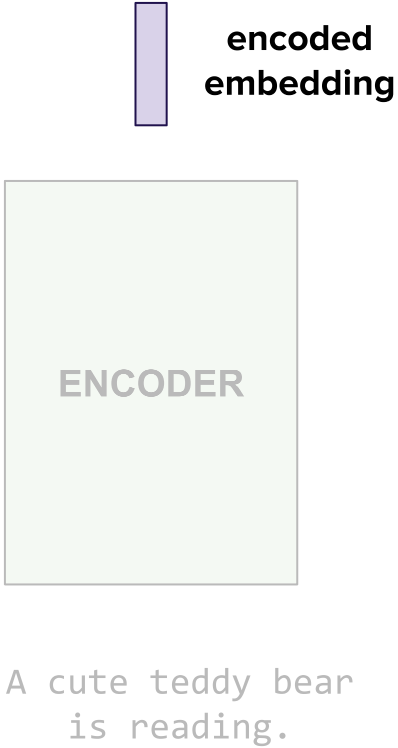
Stitching all the pieces together with an example



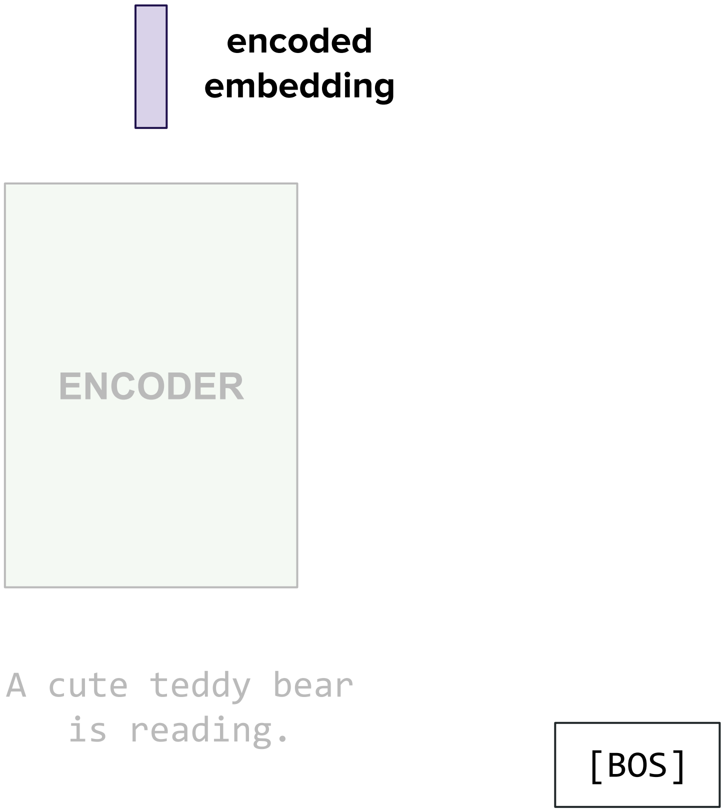
Stitching all the pieces together with an example



Stitching all the pieces together with an example

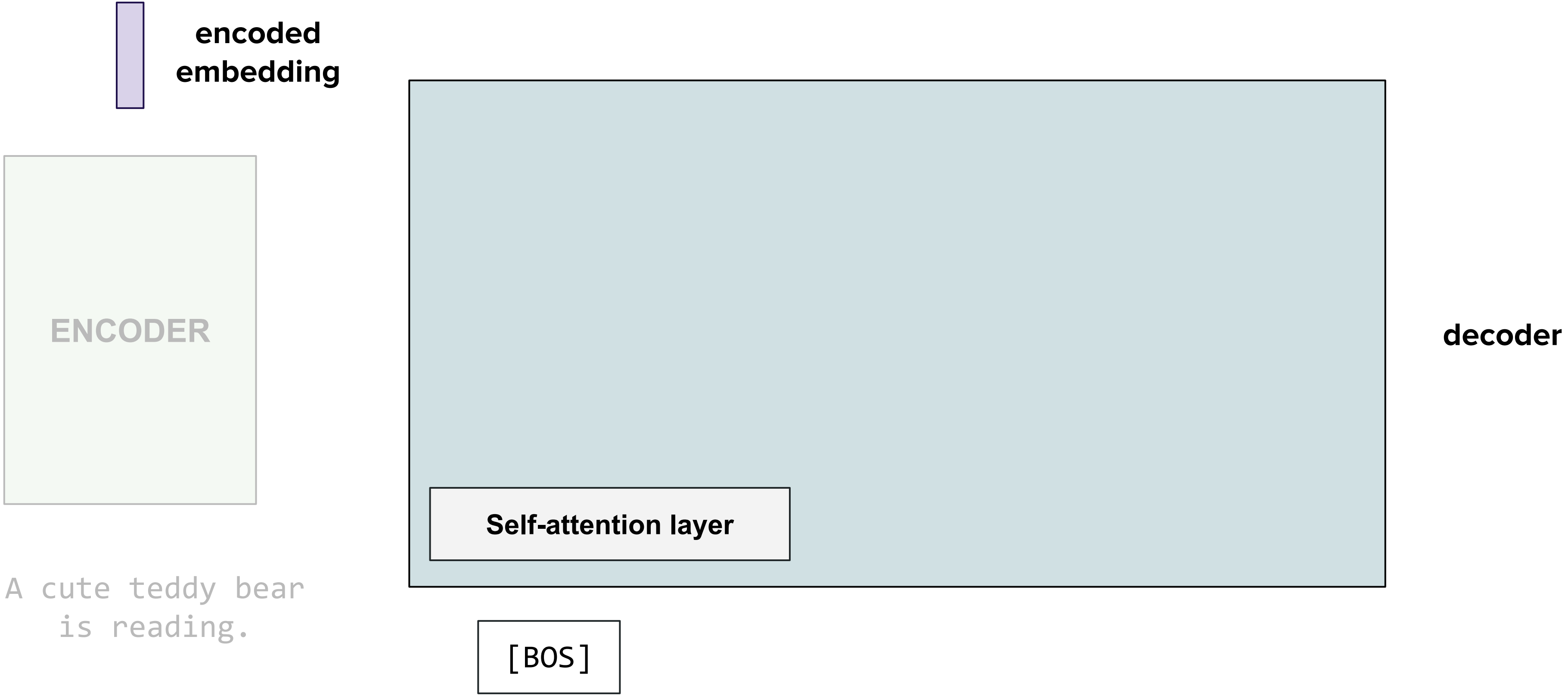


Stitching all the pieces together with an example

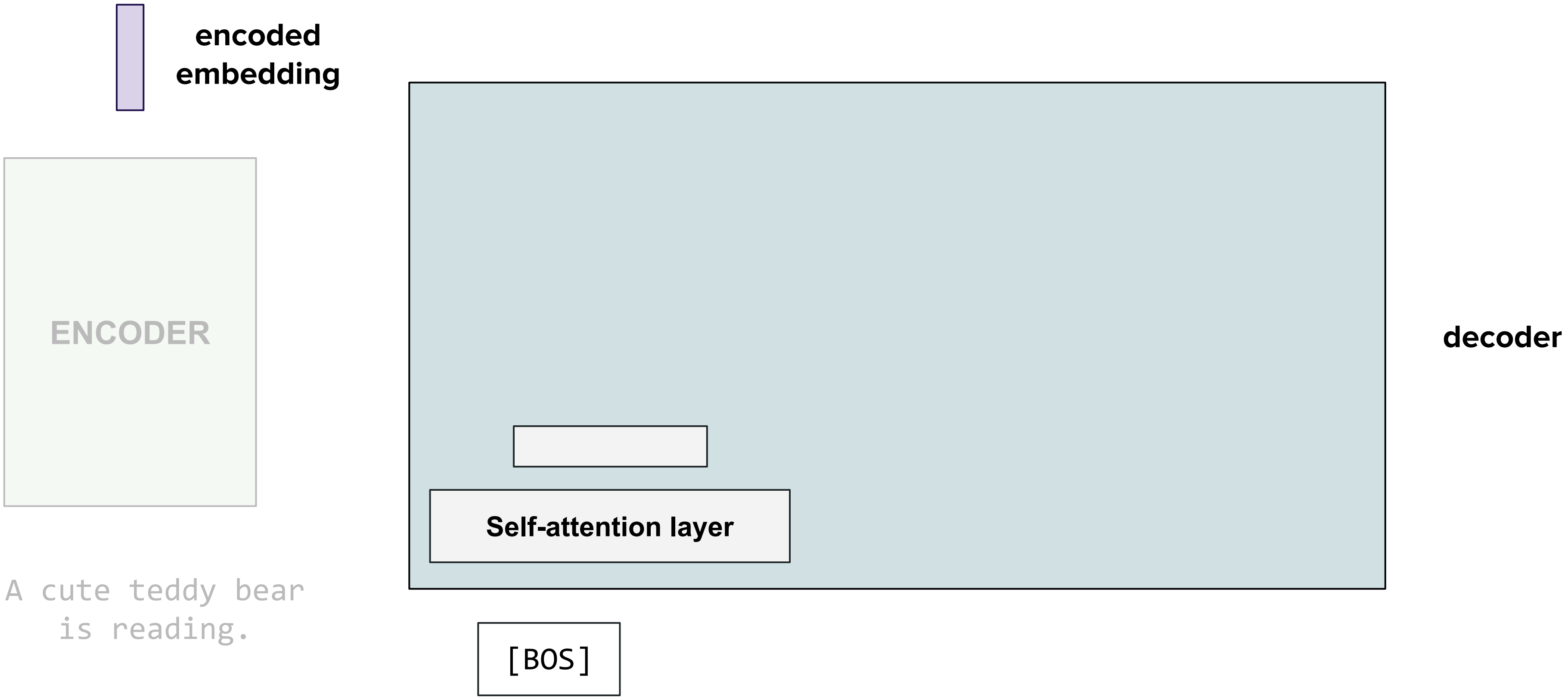


Stitching all the pieces together with an example

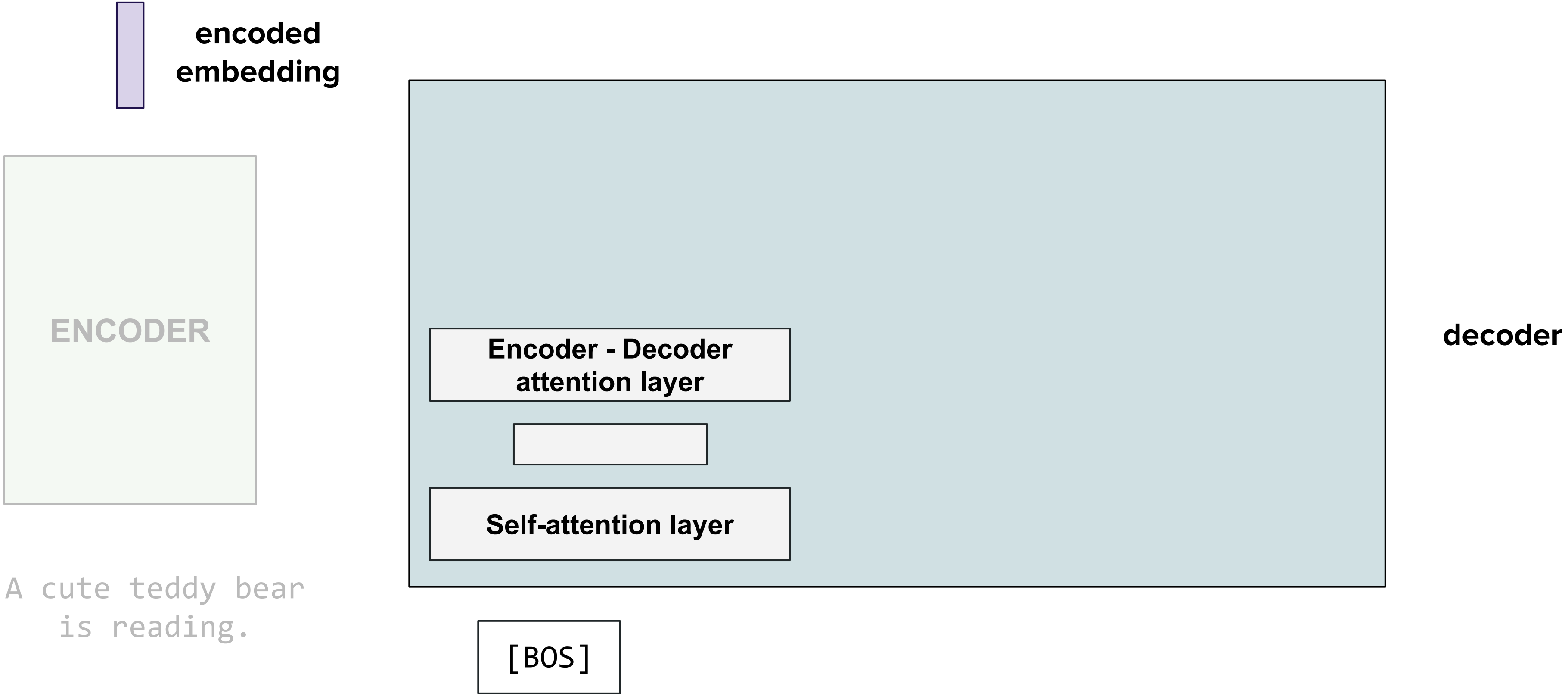
Stitching all the pieces together with an example



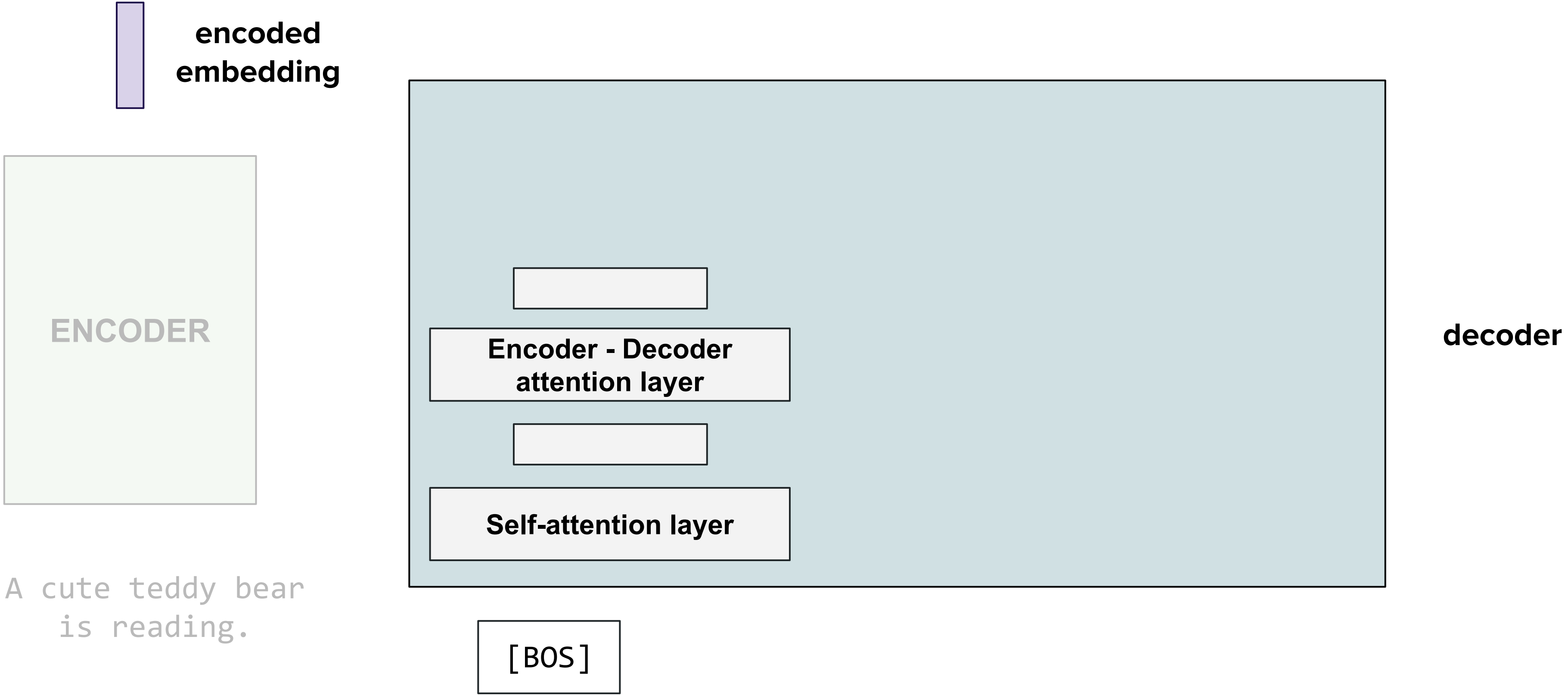
Stitching all the pieces together with an example



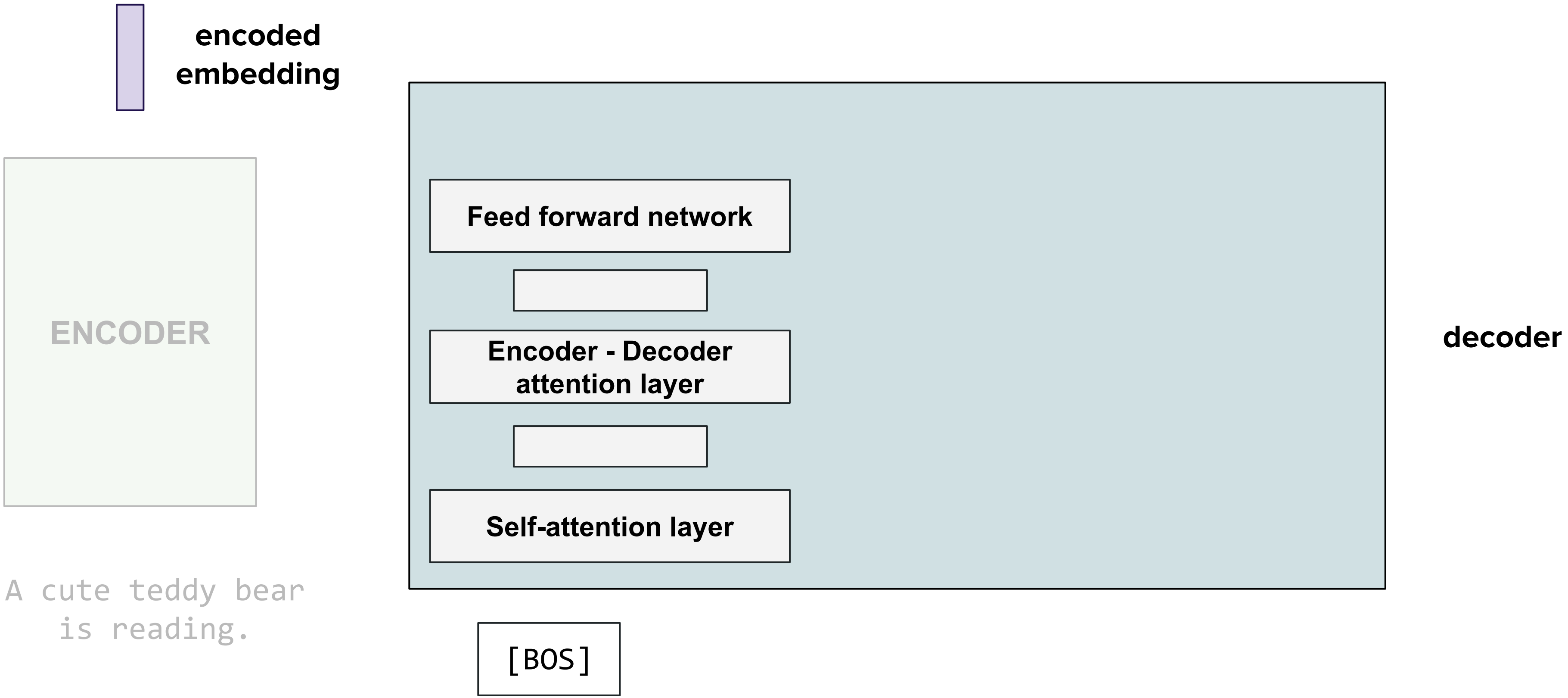
Stitching all the pieces together with an example



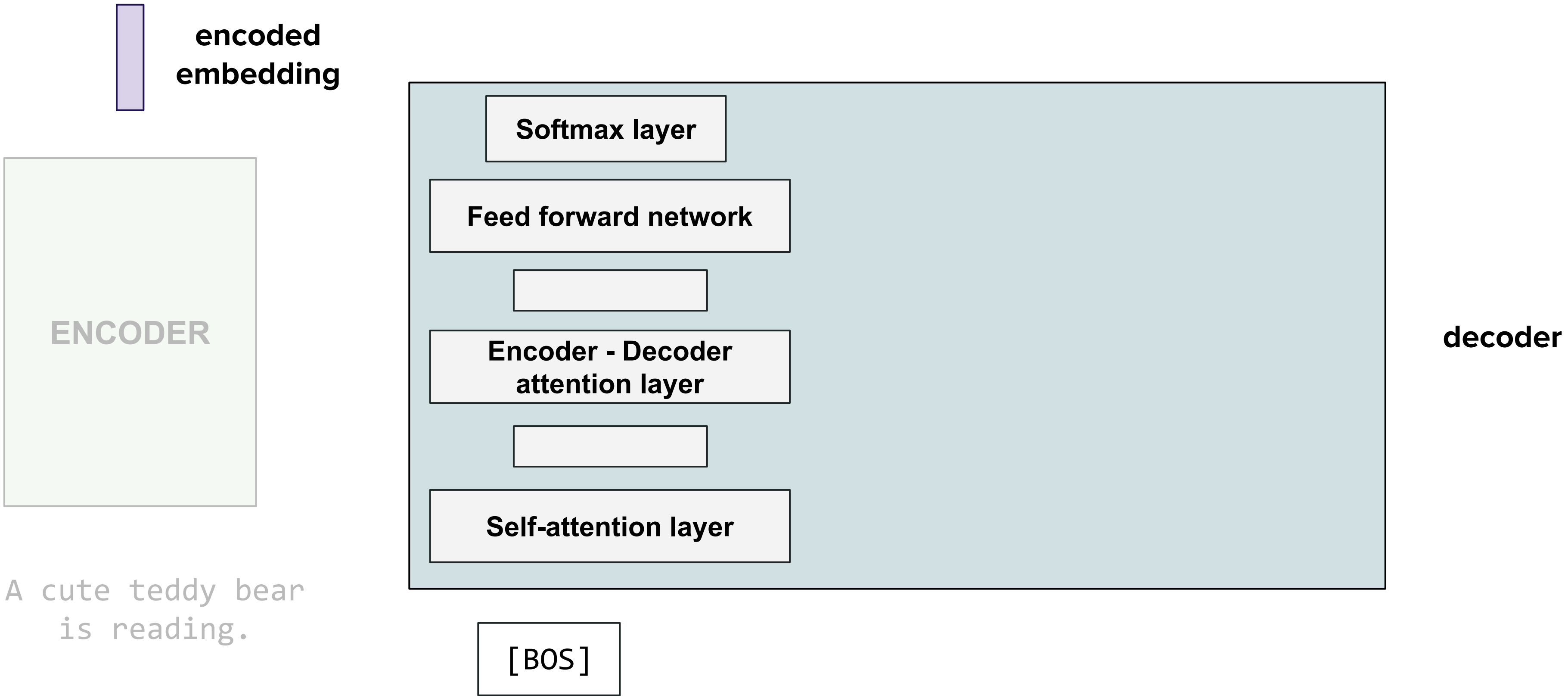
Stitching all the pieces together with an example



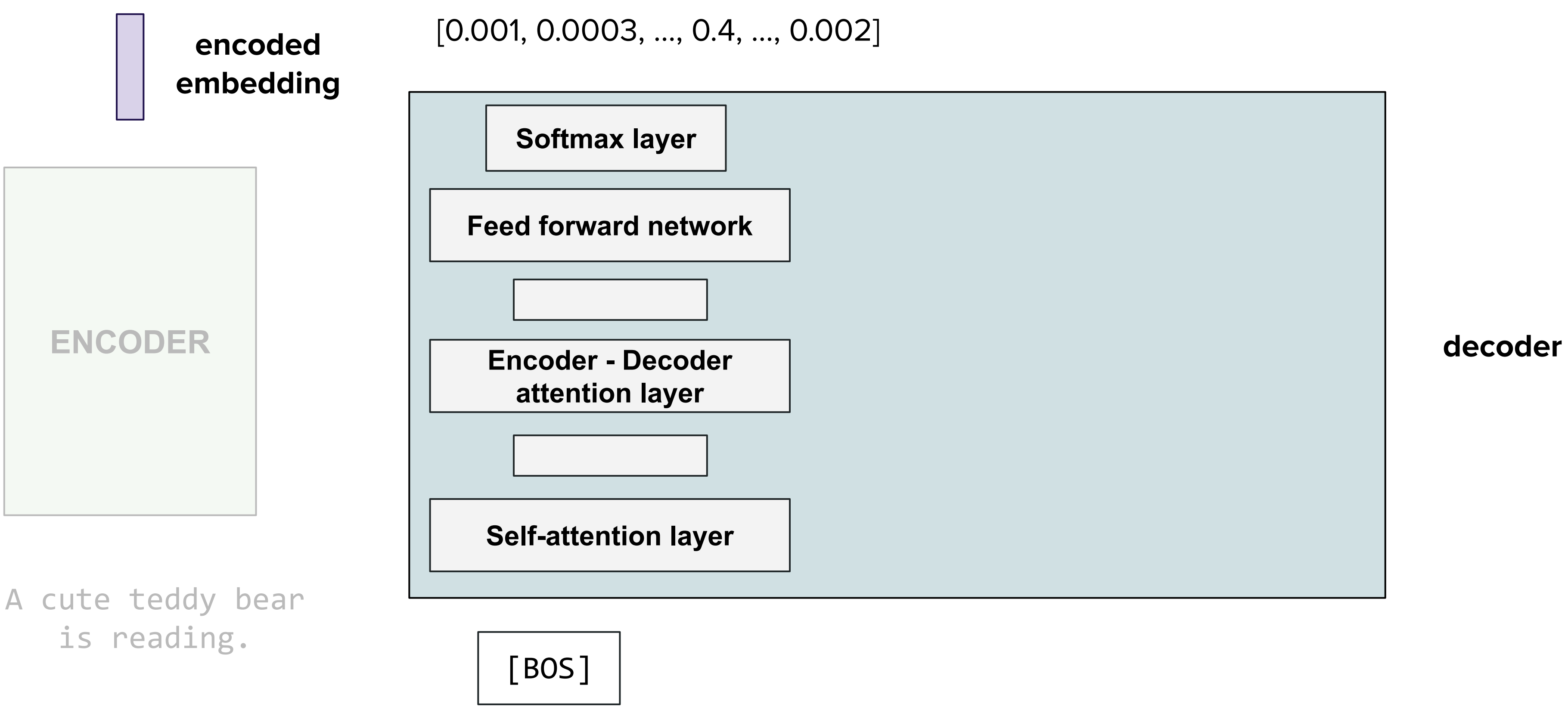
Stitching all the pieces together with an example



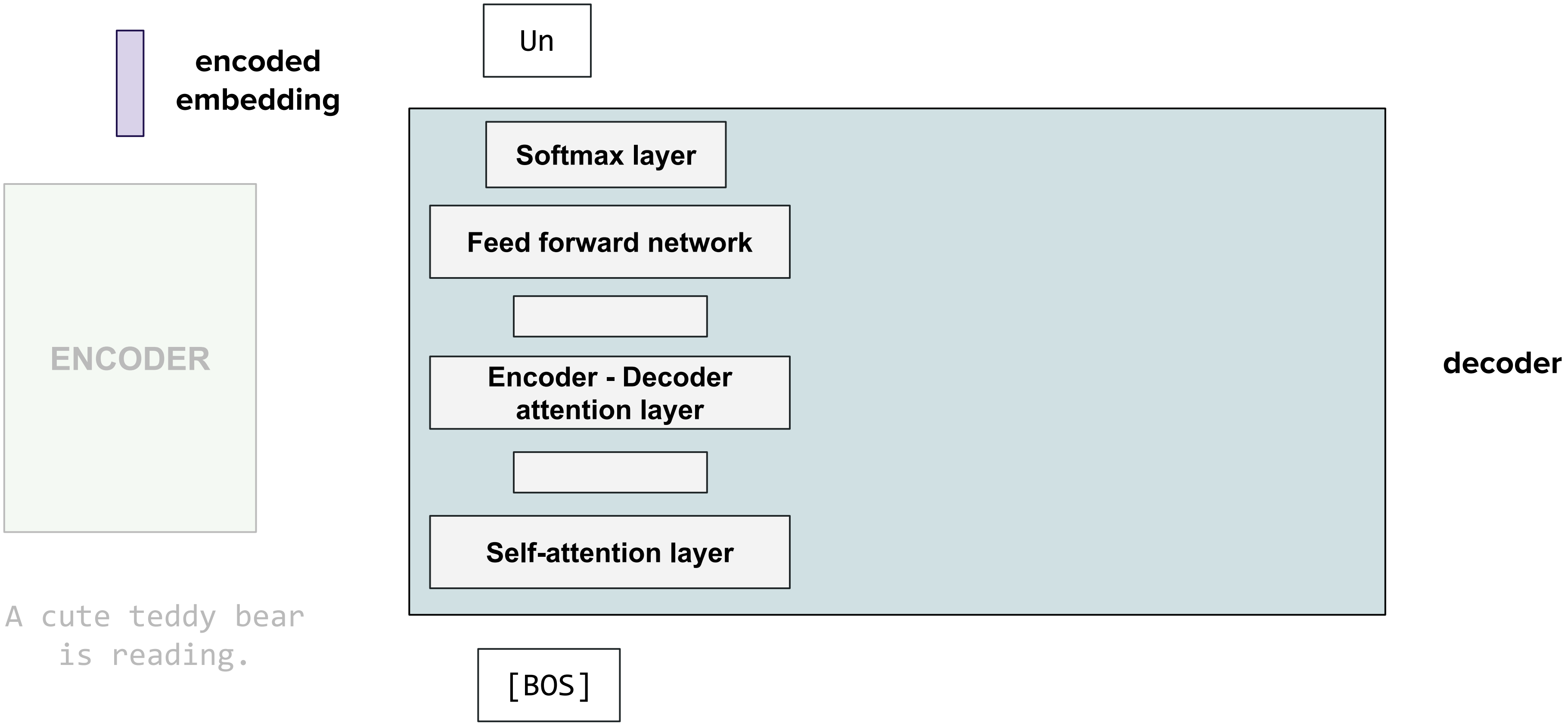
Stitching all the pieces together with an example



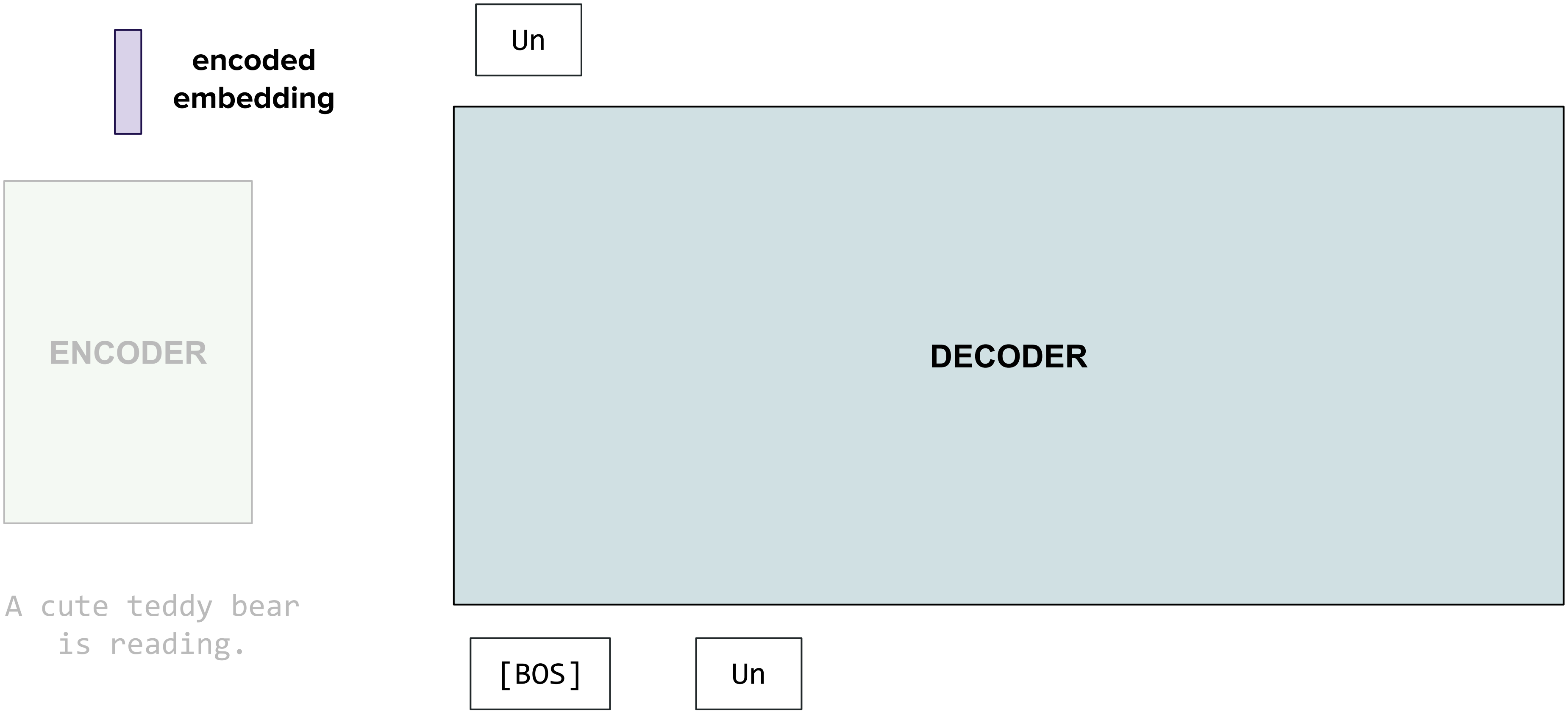
Stitching all the pieces together with an example



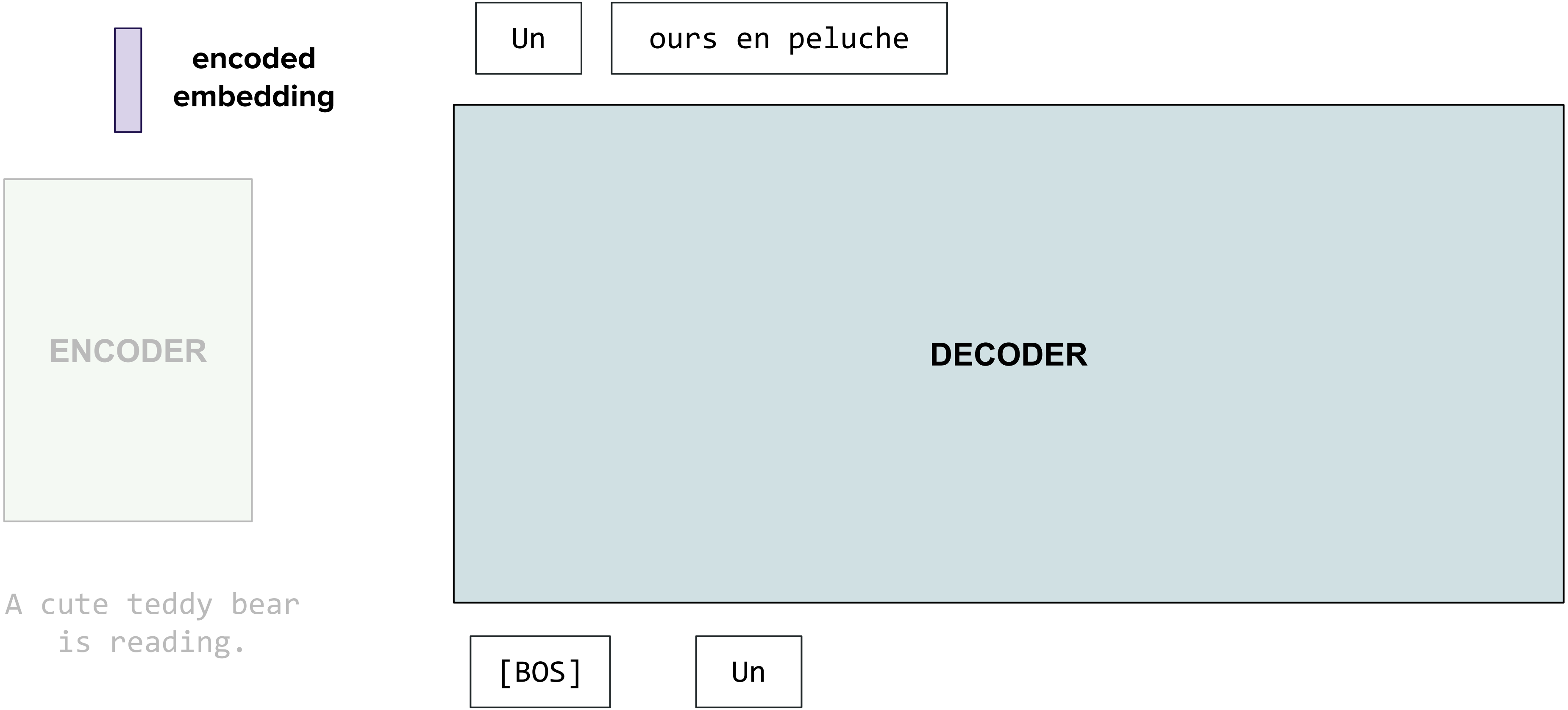
Stitching all the pieces together with an example



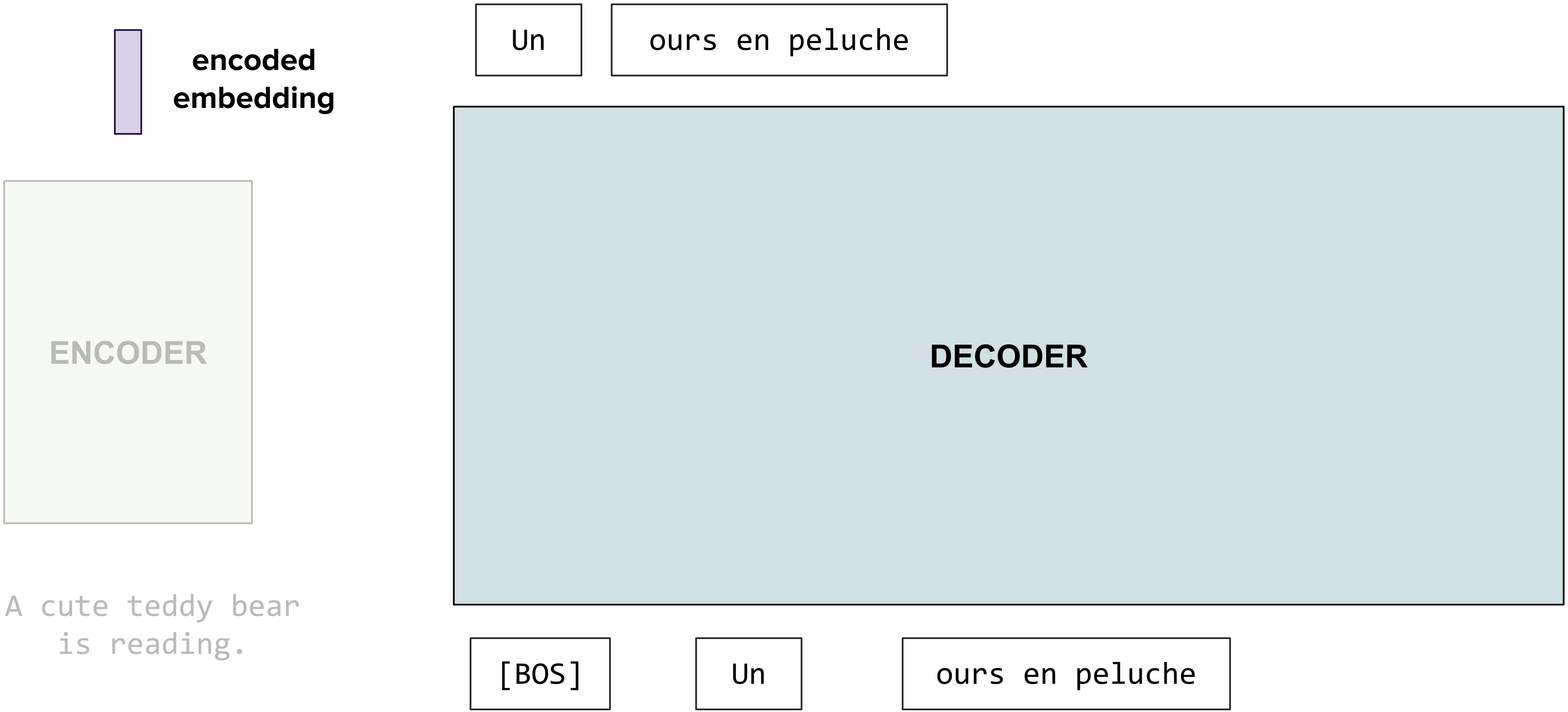
Stitching all the pieces together with an example



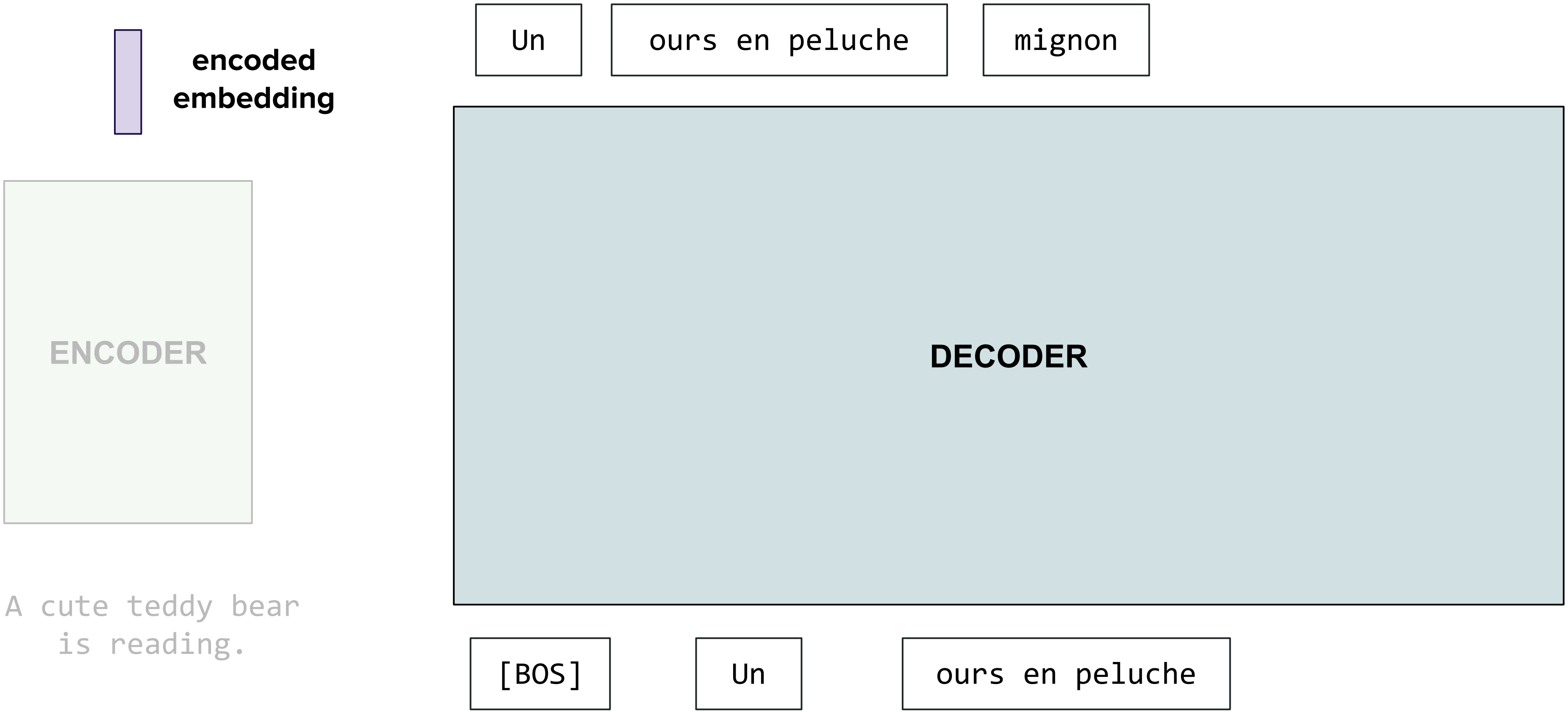
Stitching all the pieces together with an example



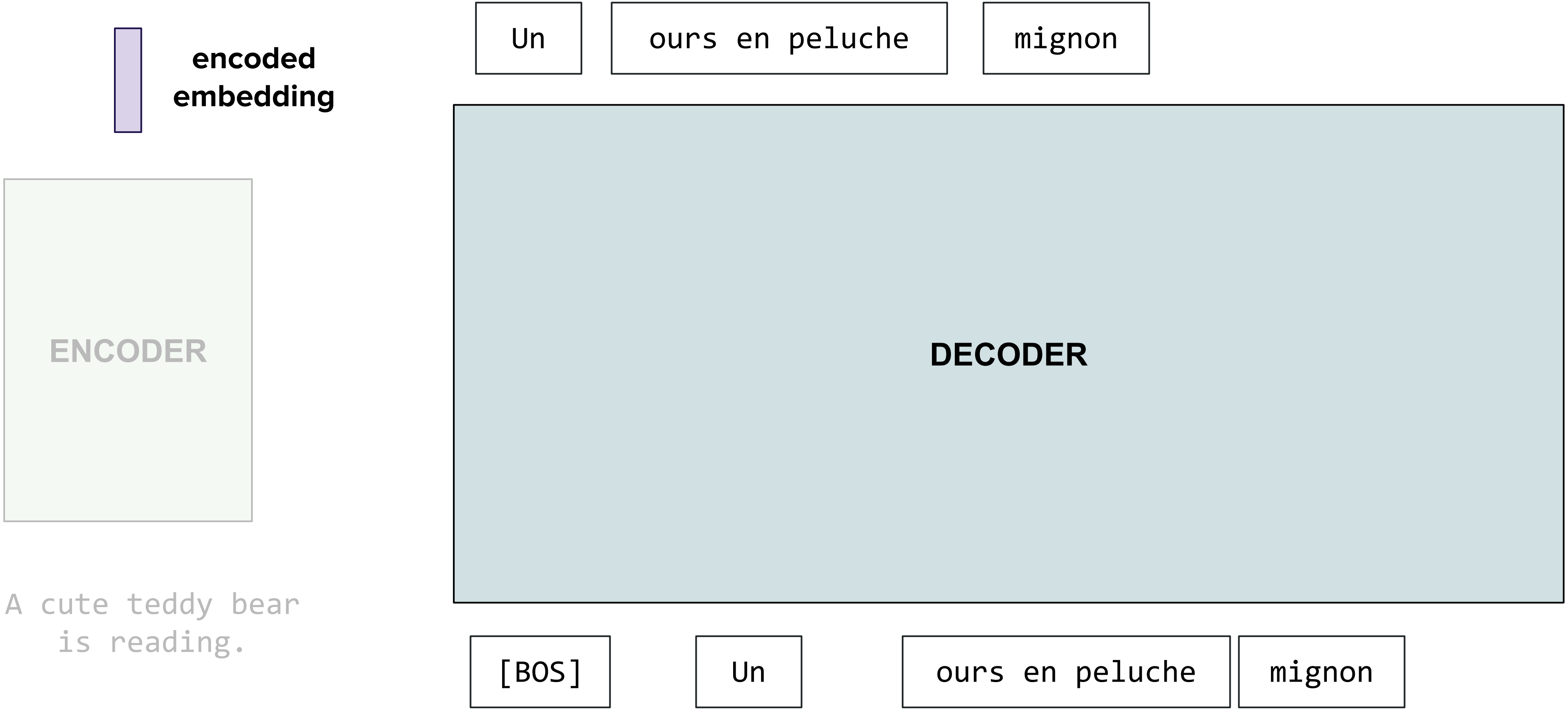
Stitching all the pieces together with an example



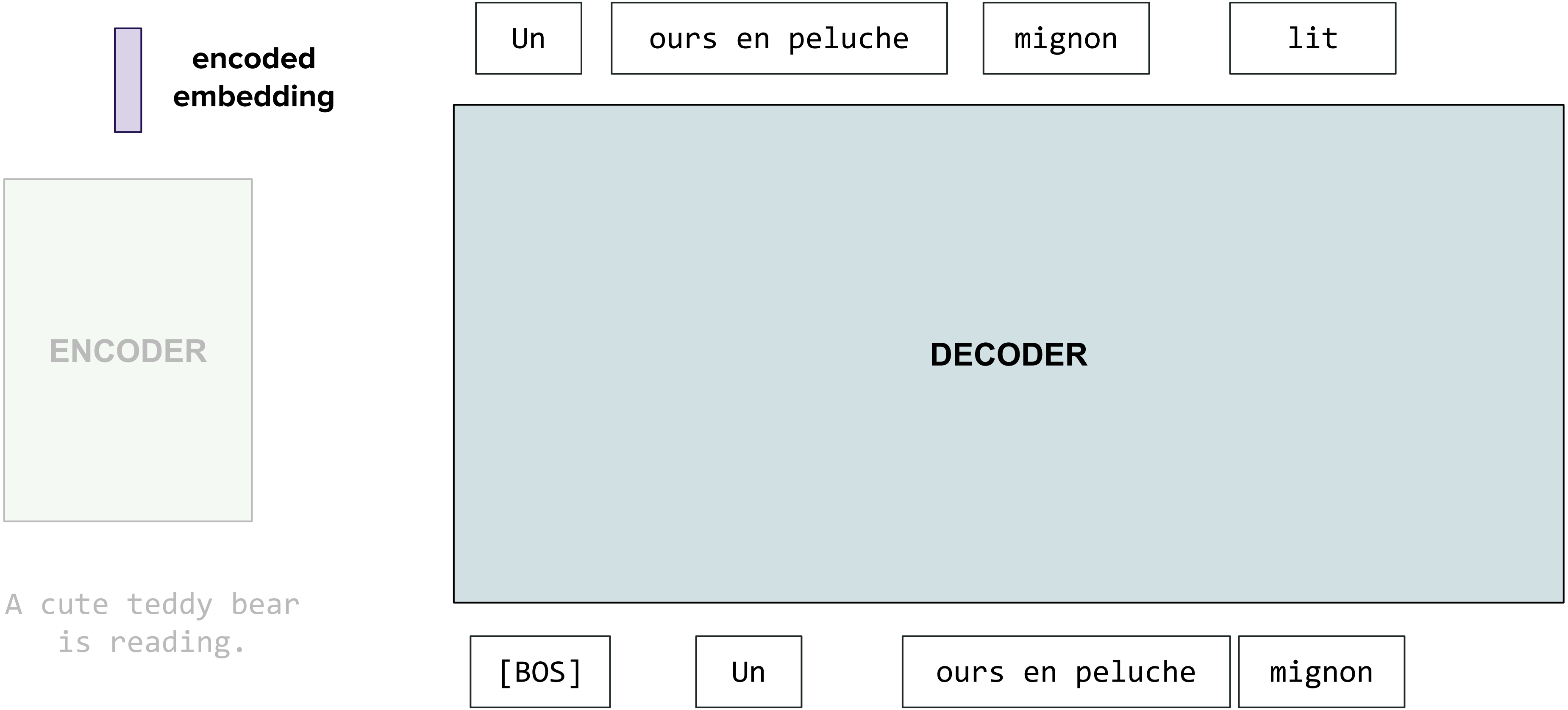
Stitching all the pieces together with an example



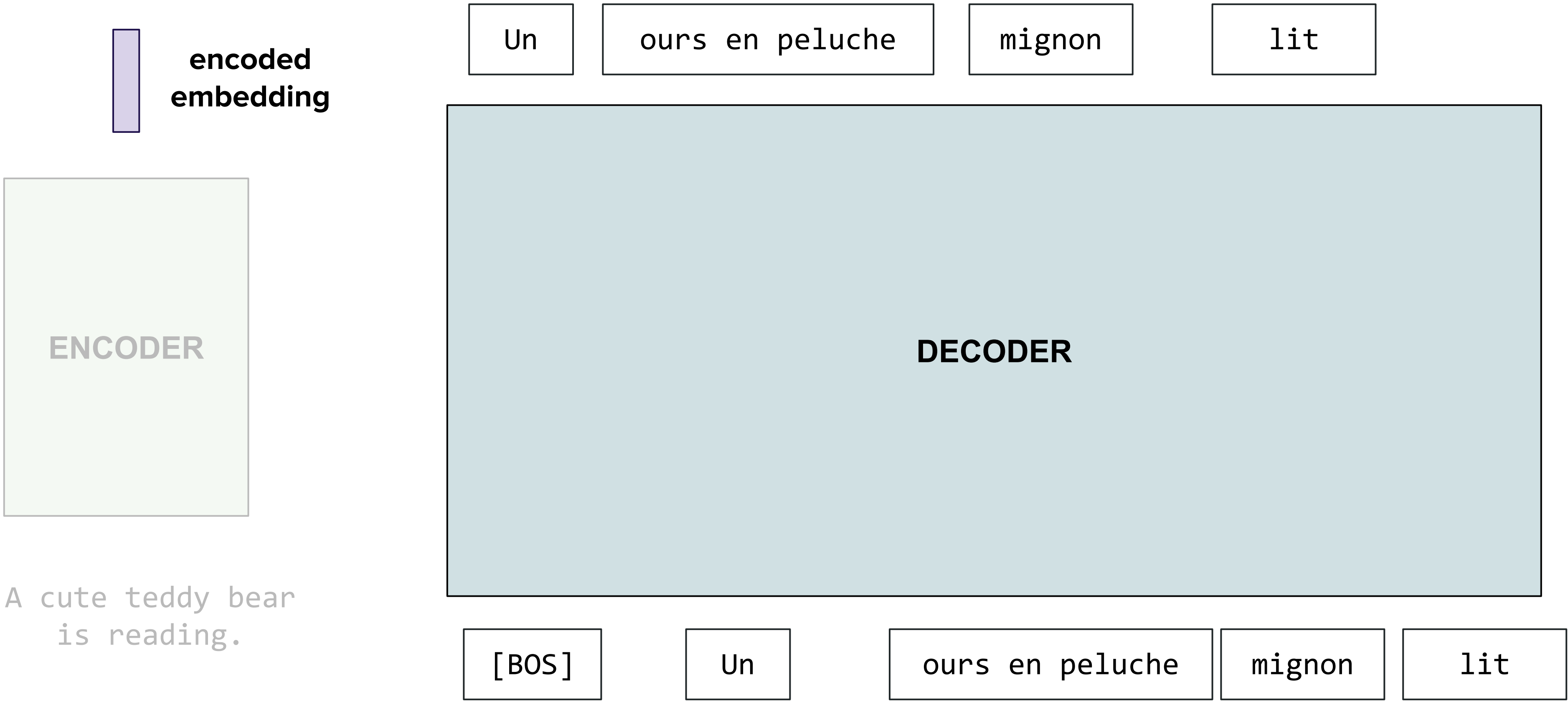
Stitching all the pieces together with an example



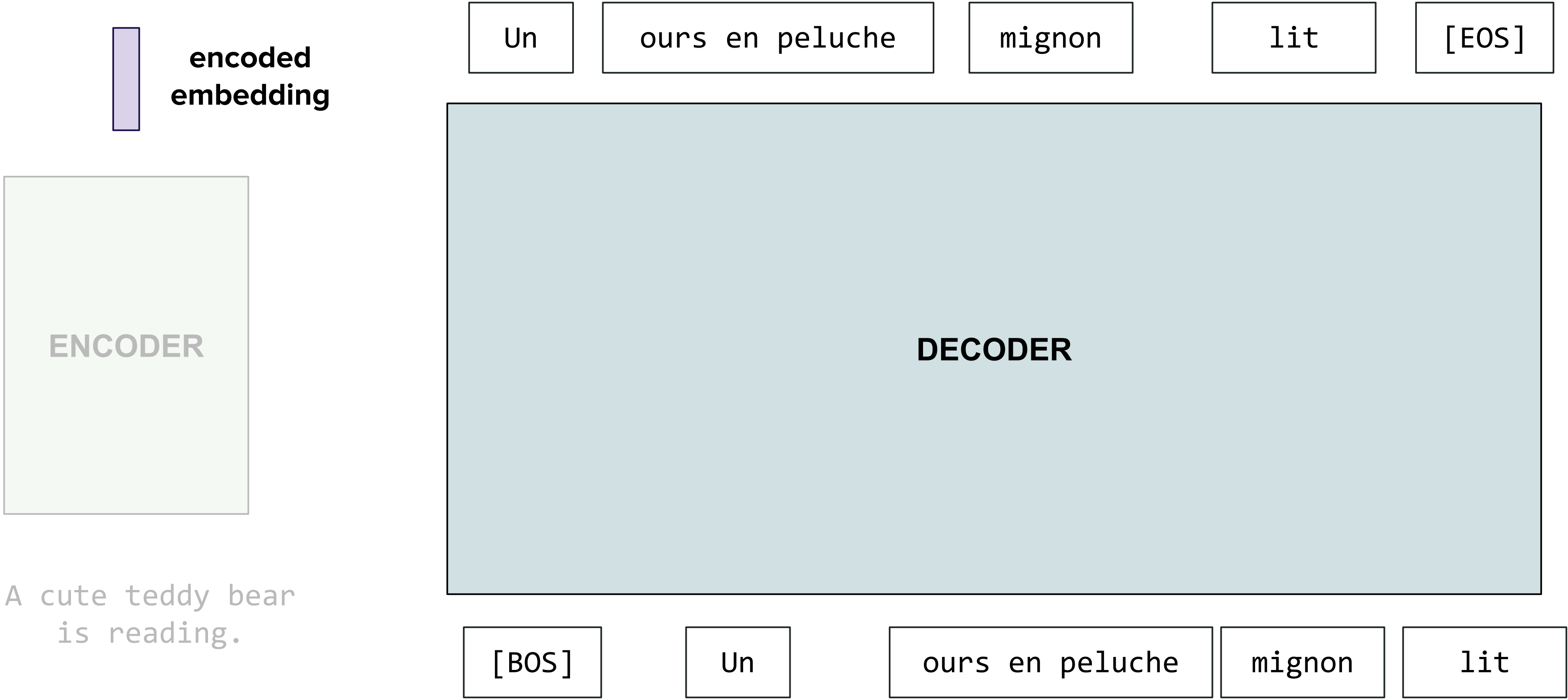
Stitching all the pieces together with an example



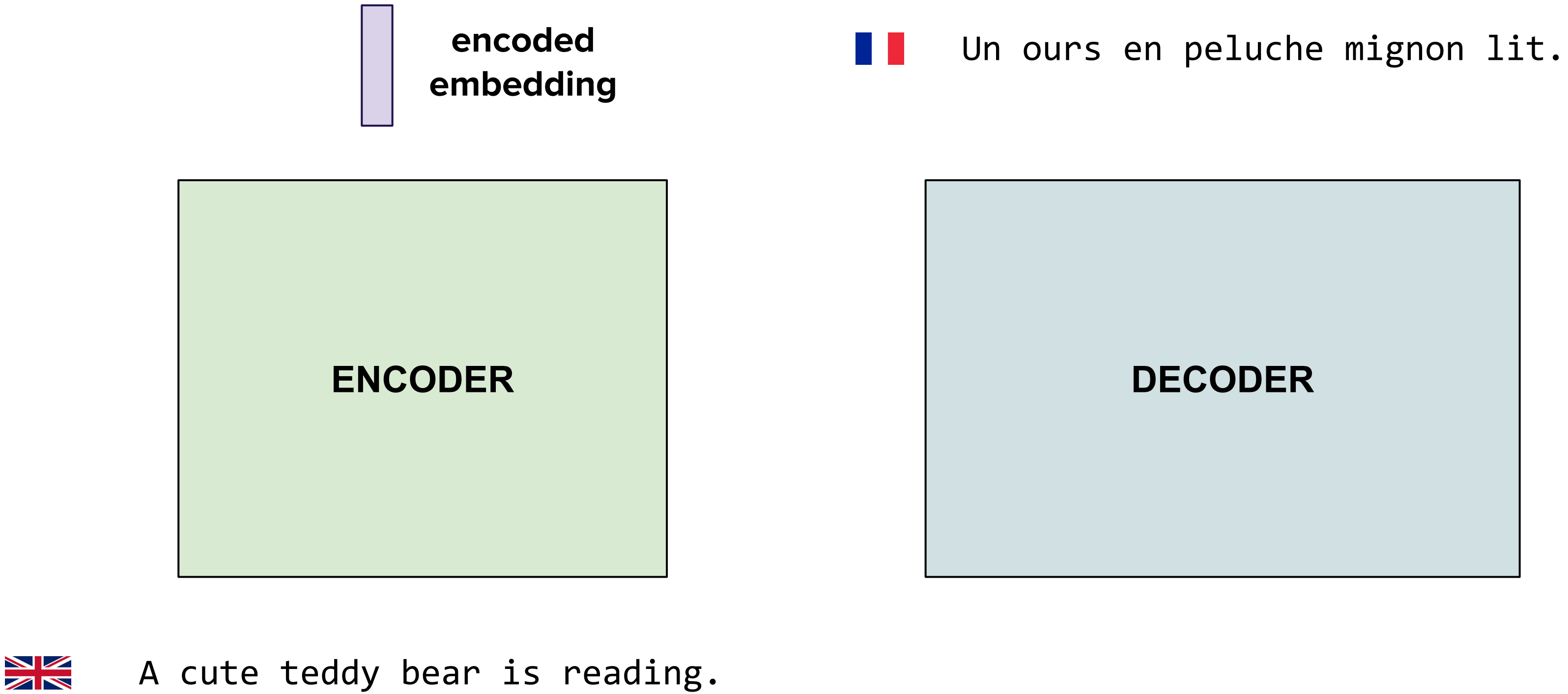
Stitching all the pieces together with an example



Stitching all the pieces together with an example



Stitching all the pieces together with an example



See you on Friday!