Section with Barbara

Week 6

AI Stories

Projects

Language

A Friendly Reminder

Project 6 is due this Friday, August 7!

AI Stories: IBM Watson

Natural Language Understanding (NLU)

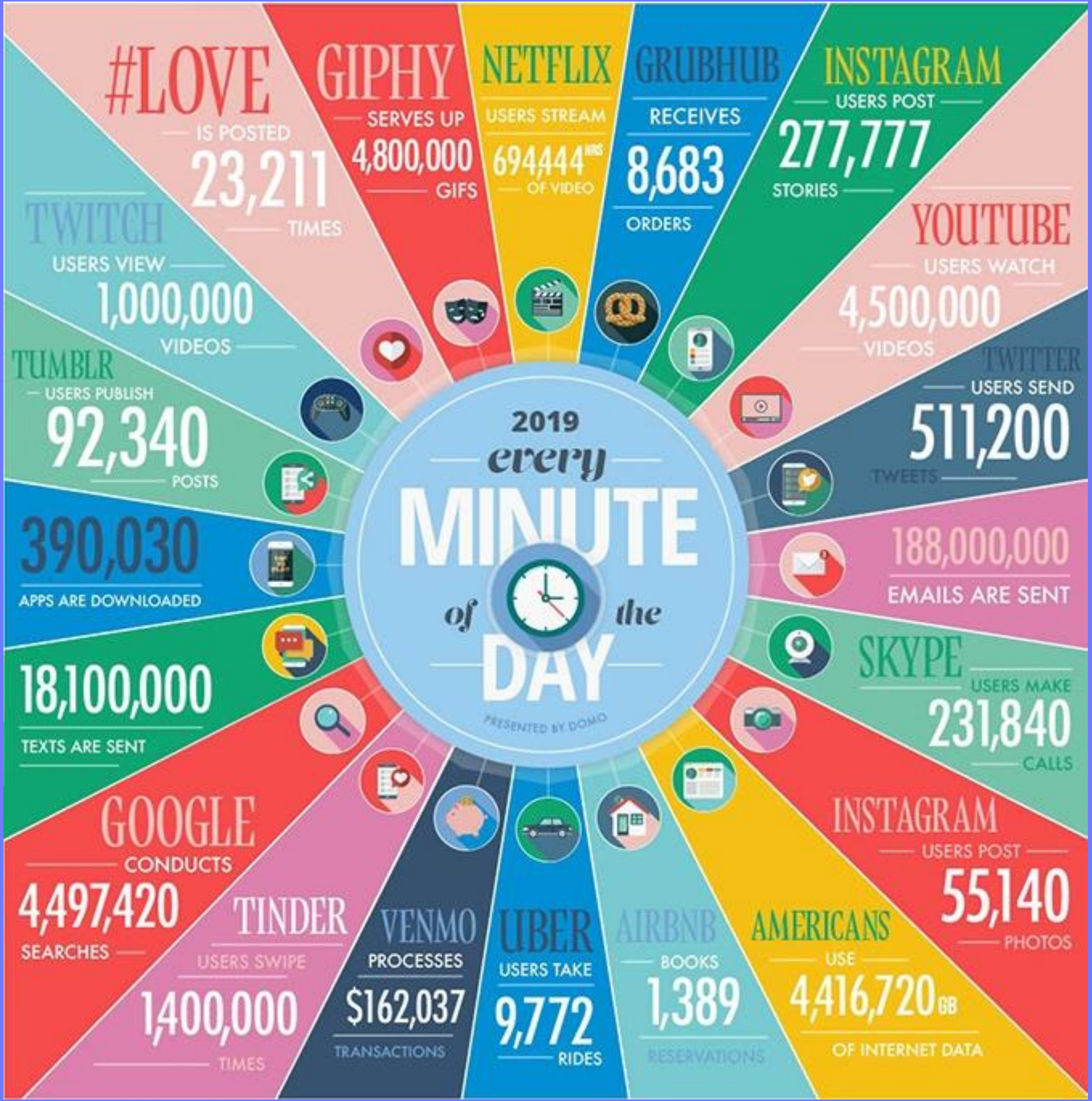
AI can read

Natural Language Generation (NLG)

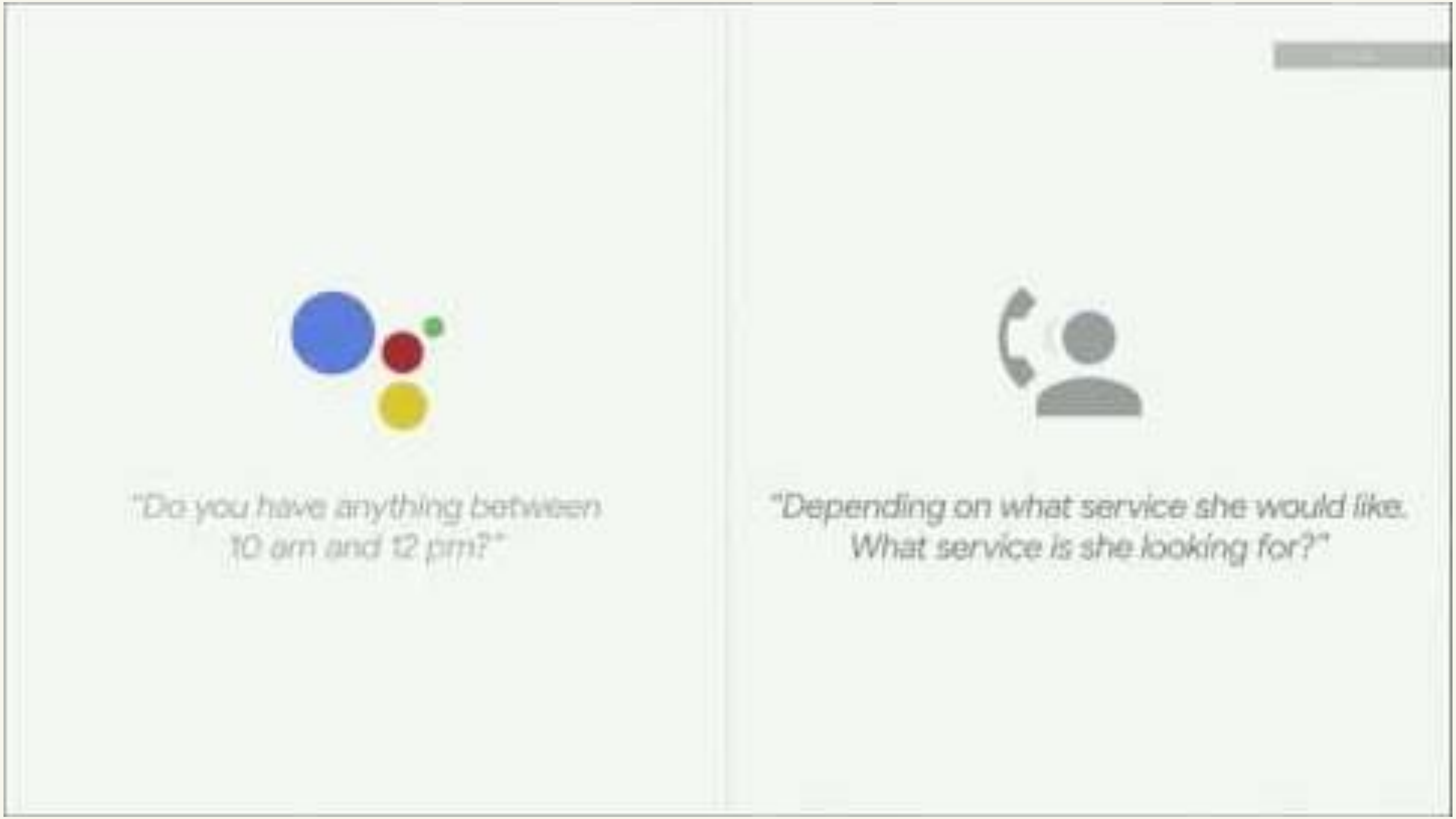
AI can write

Natural Language Processing (NLP)

AI can analyze



2019 every MINUTE of the DAY
PRESENTED BY DOMO
#LOVE IS POSTED 23,211 TIMES
GIPHY SERVES UP 4,800,000 GIFS
NETFLIX USERS STREAM 694,444 HRS OF VIDEO
GRUBHUB RECEIVES 8,683 ORDERS
INSTAGRAM USERS POST 277,777 STORIES
YOUTUBE USERS WATCH 4,500,000 VIDEOS
TWITTER USERS SEND 511,200 TWEETS
188,000,000 EMAILS ARE SENT
SKYPE USERS MAKE 231,840 CALLS
INSTAGRAM USERS POST 55,140 PHOTOS
AMERICANS USE 4,416,720 GB OF INTERNET DATA
AIRBNB BOOKS 1,389 RESERVATIONS
UBER USERS TAKE 9,772 RIDES
VENMO PROCESSES $162,037 TRANSACTIONS
TINDER USERS SWIPE 1,400,000 TIMES
GOOGLE CONDUCTS 4,497,420 SEARCHES
18,100,000 TEXTS ARE SENT
390,030 APPS ARE DOWNLOADED
TUMBLR USERS PUBLISH 92,340 POSTS
TWITCH USERS VIEW 1,000,000 VIDEOS


"Do you have anything between
10 am and 12 pm?"
"Depending on what service she would like.
What service is she looking for?"

Current State of the Art in NLP: GPT-3

Artificial intelligence programs like deep learning neural networks may be able to beat humans at playing Go or chess, or doing arithmetic, or writing Navy Seal copypasta, but they will never be able to truly think for themselves, to have consciousness, to feel any of the richness and complexity of the world that we mere humans can feel. Mere, unenlightened humans might be impressed by the abilities of simple deep learning programs, but when looked at in a more holistic manner, it all adds up to… well, nothing. They still don't exhibit any trace of consciousness. All of the available data support the notion that humans feel and experience the world differently than computers do. While a computer can beat a human master at chess or Go or some other game of structured rules, it will never be able to truly think outside of those rules, it will never be able to come up with its own new strategies on the fly, it will never be able to feel, to react, the way a human can. Artificial intelligence programs lack consciousness and self-awareness. They will never be able to have a sense of humor. They will never be able to appreciate art, or beauty, or love. They will never feel lonely. They will never have empathy for other people, for animals, for the environment. They will never enjoy music or fall in love, or cry at the drop of a hat. Merely by existing, mere, unenlightened humans are intellectually superior to computers, no matter how good our computers get at winning games like Go or Jeopardy. We don't live by the rules of those games. Our minds are much, much bigger than that. – *https://www.gwern.net/GPT-3*

Current State of the Art in NLP: GPT-3

GPT-3 wrote the text in the previous slide!

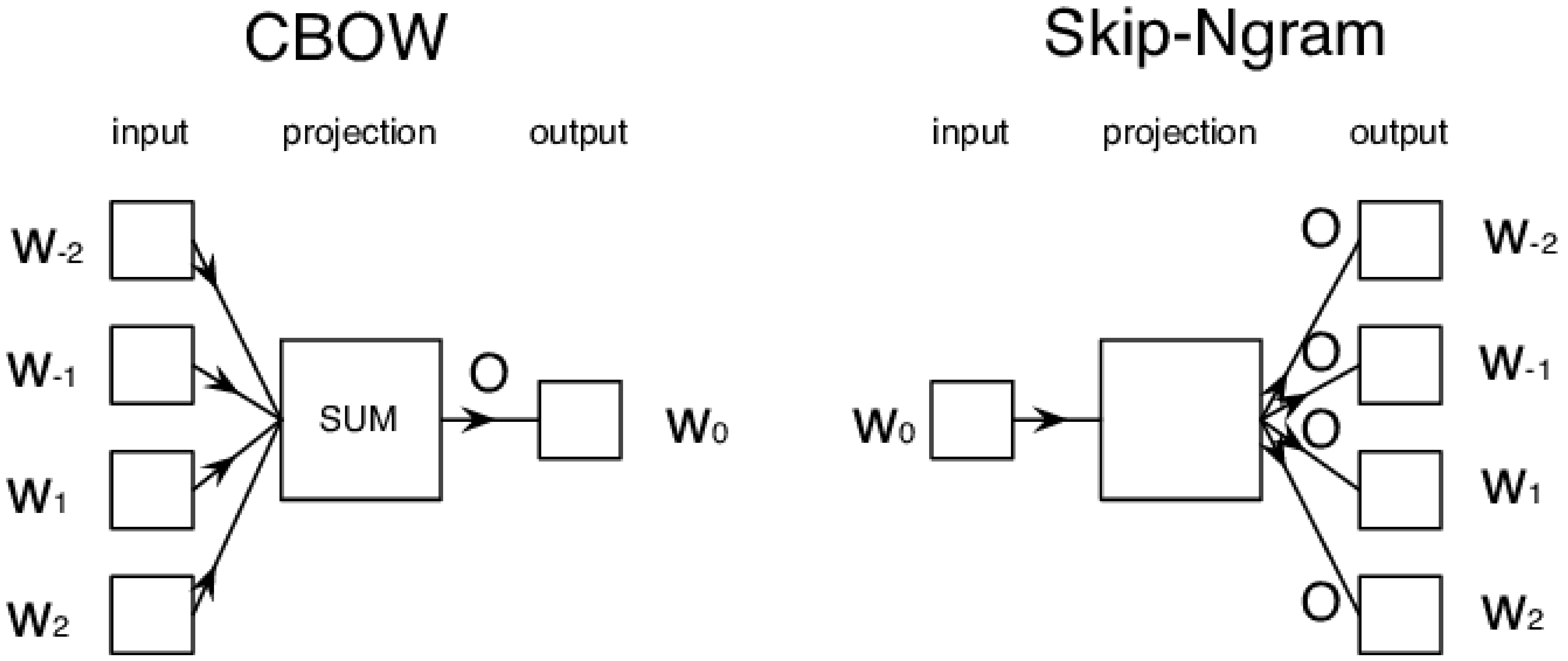
Language and AI

- 1950’s: Beginnings, Machine Translation
- Pre-1990’s: Knowledge Bases and Rules
- ‘90’s and ‘00’s: Statistical Learning
- 2010’s: Deep Learning
- Now: Large Pretrained Models

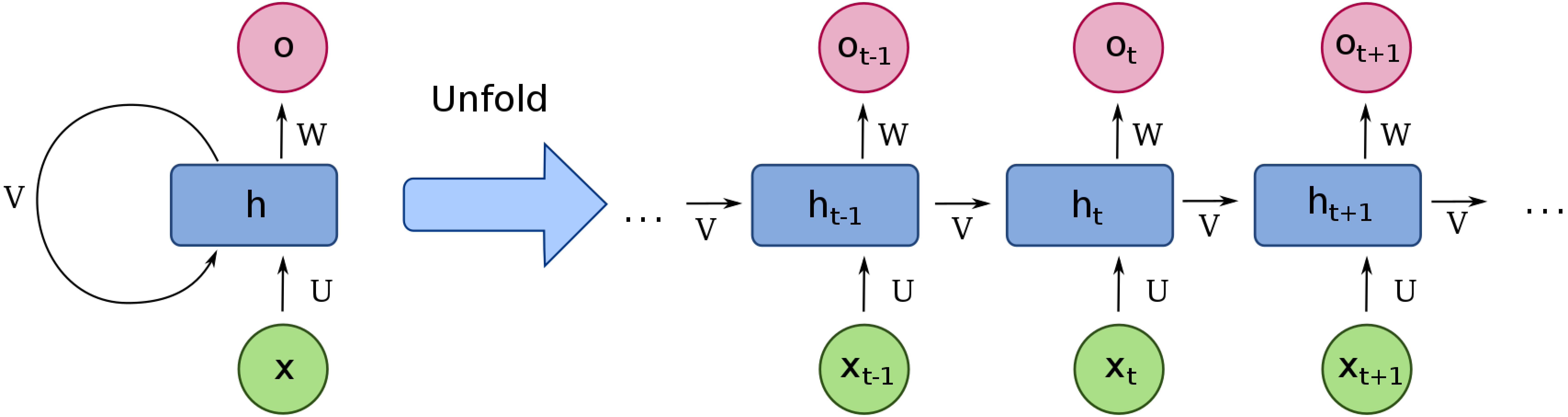
Deep Learning

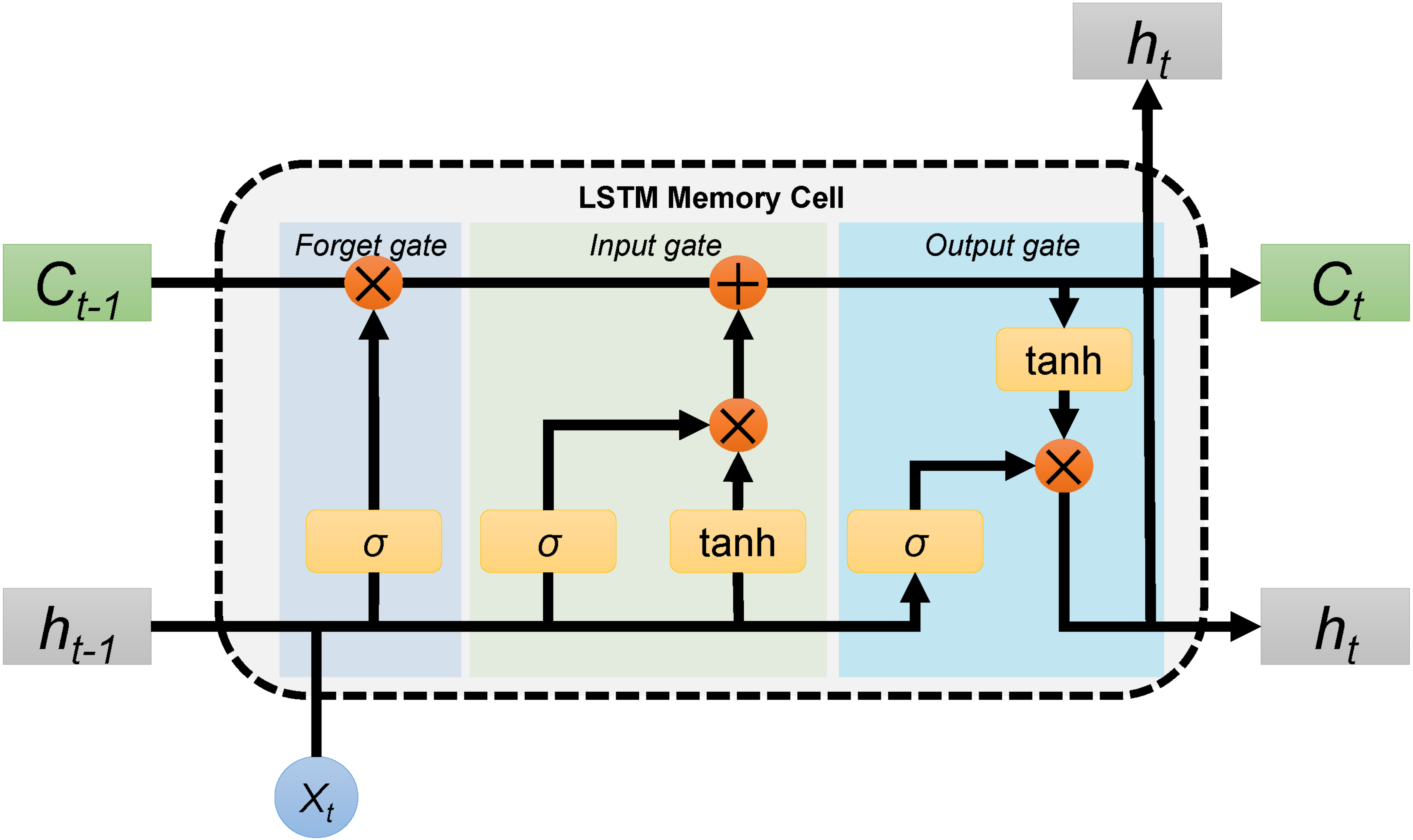
- 2003: “A Neural Probabilistic Language Model”
- 2008: “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”
- 2013: “Efficient Estimation of Word Representations in Vector Space”
- 2014: “Sequence to Sequence Learning with Neural Networks”
- 2017: “Attention Is All You Need”

Word Embeddings



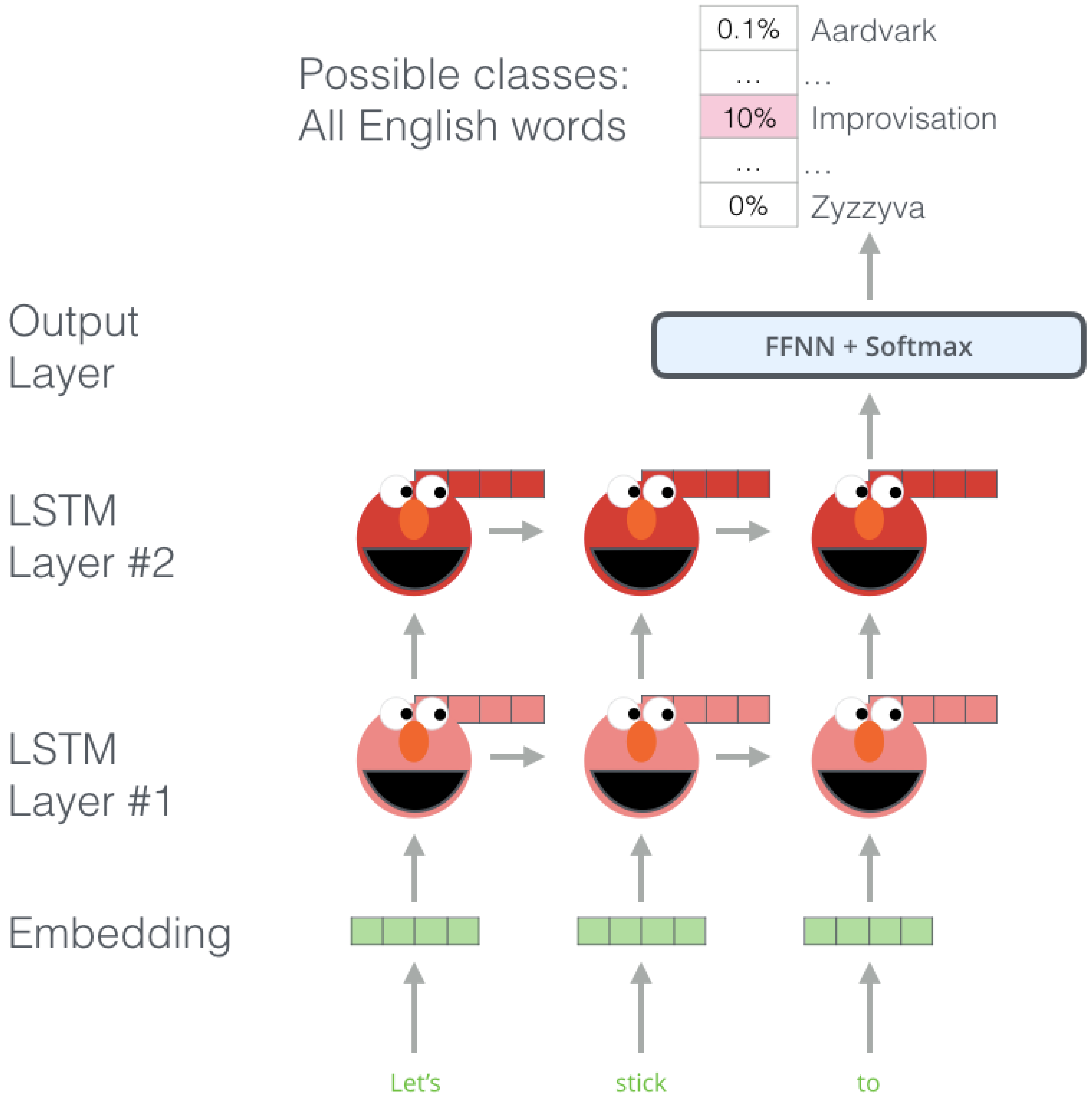
Recurrent Neural Network



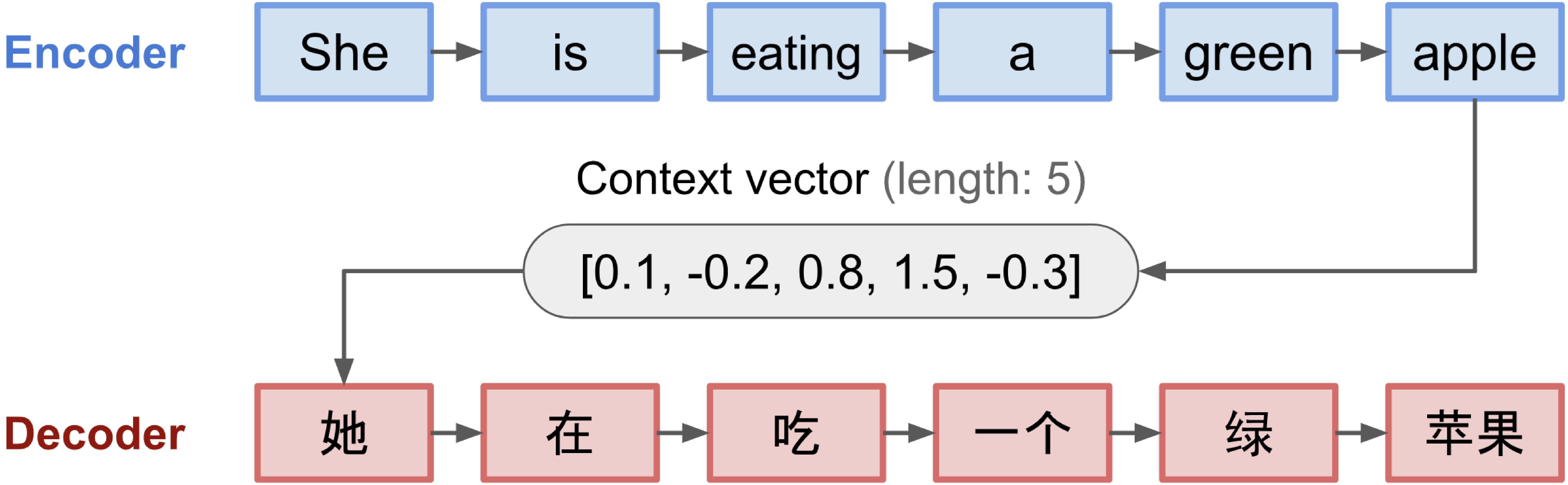


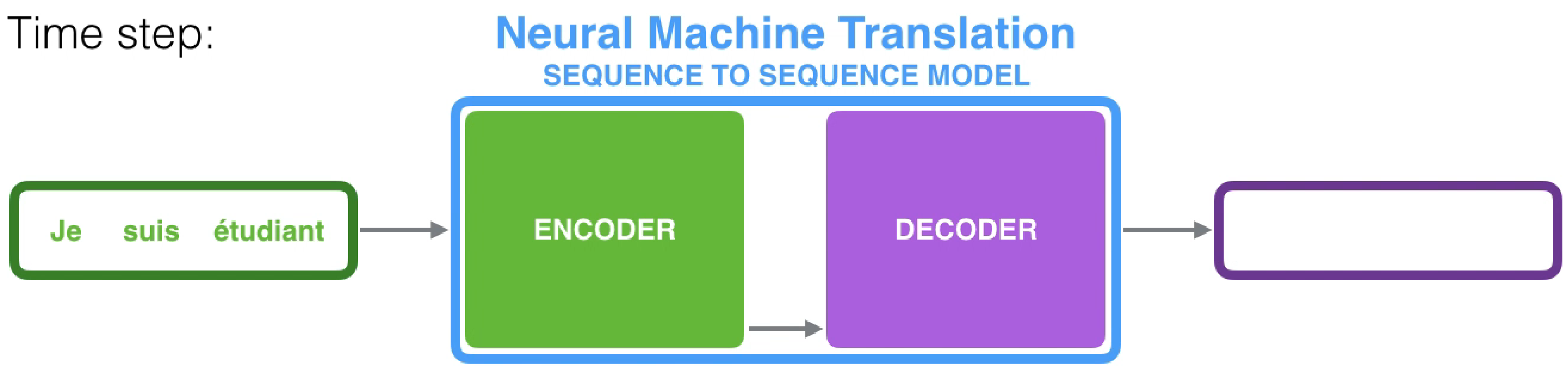
LSTM Memory Cell
Forget gate
Input gate
Output gate
C_{t-1}
C_t
h_{t-1}
h_t
h_t
X_t
σ
σ
tanh
σ
tanh

ELMo – Bidirectional LSTM



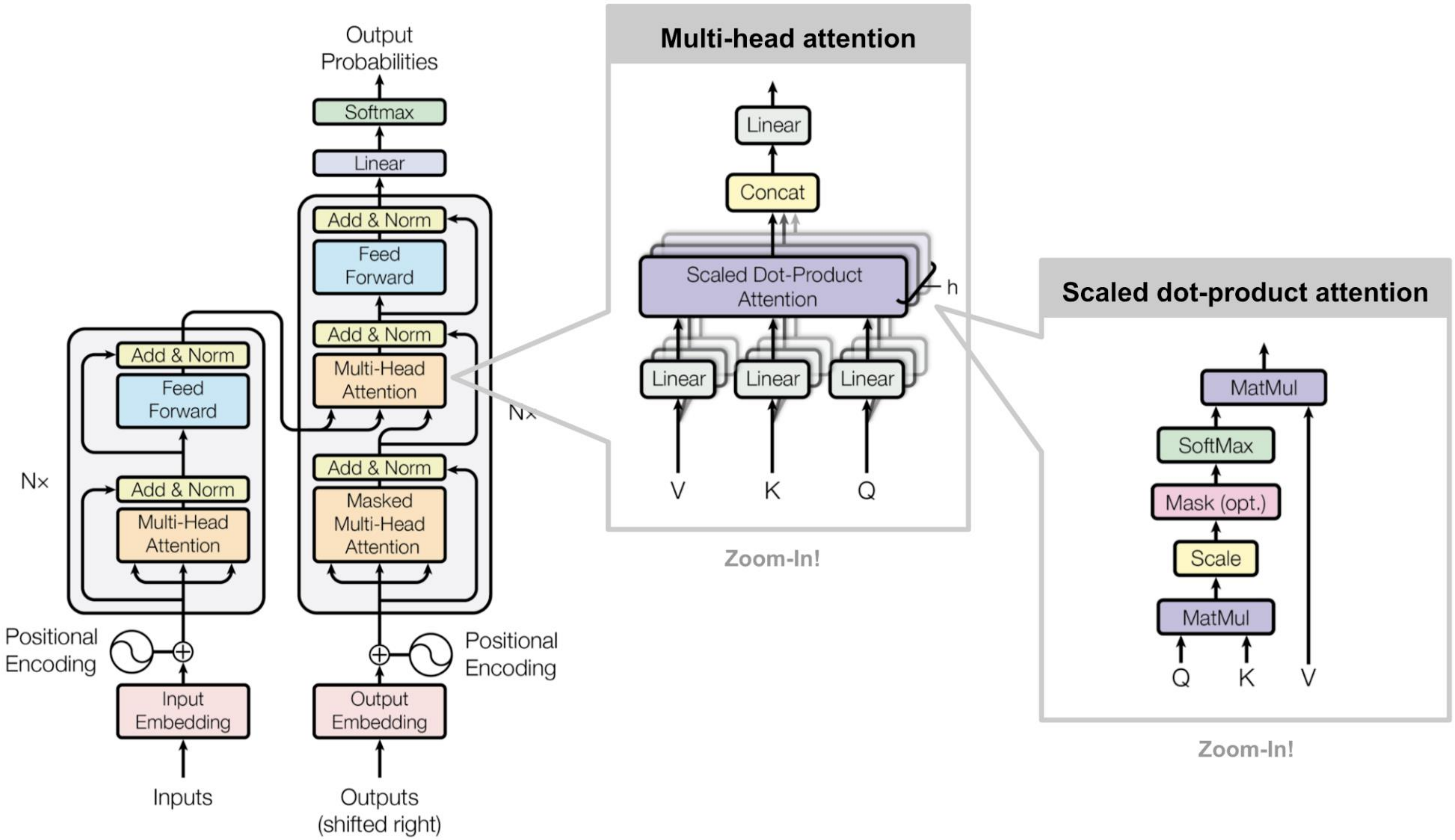
seq2seq models



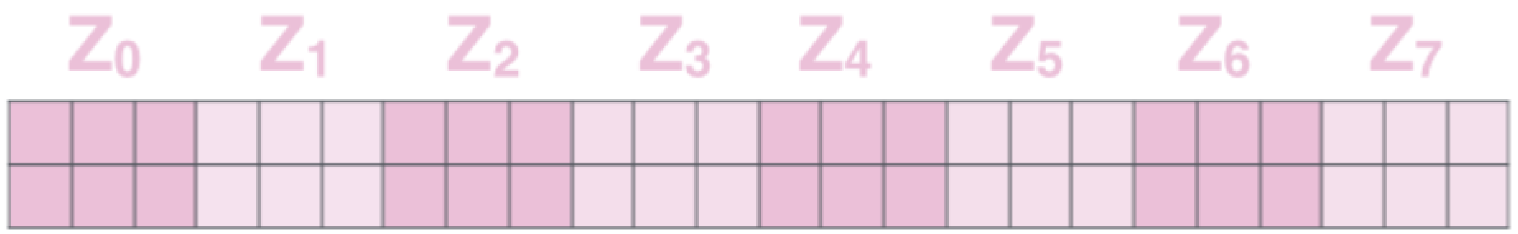

Time step:
Neural Machine Translation
SEQUENCE TO SEQUENCE MODEL
Je suis étudiant
ENCODER
DECODER

Currently we are in the dawn of large, pretrained models able to perform extremely well with just a little bit of fine tuning.

Transformer

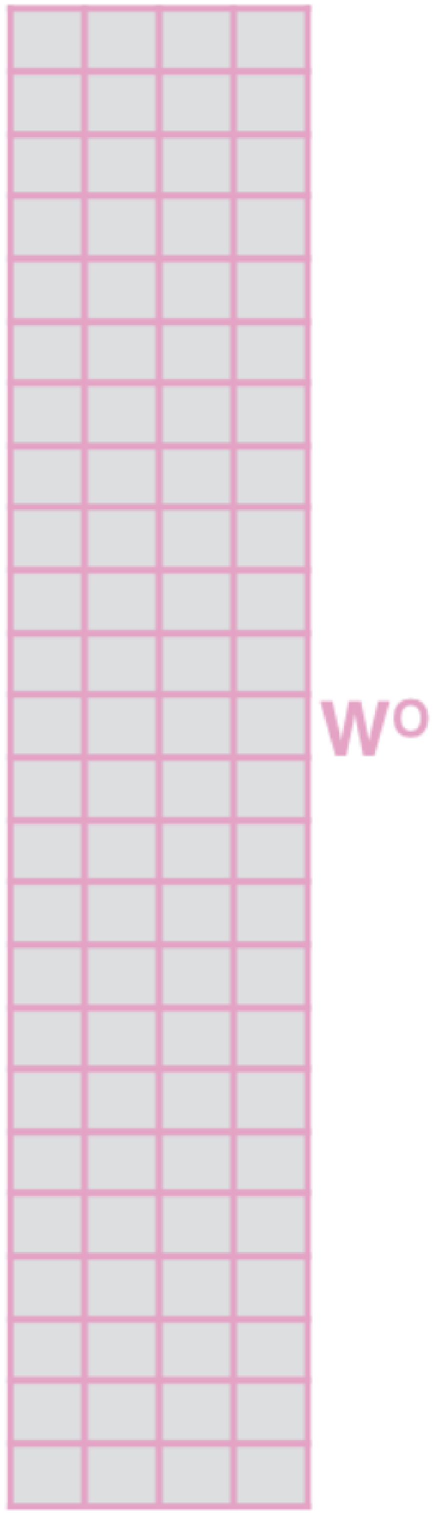


1) Concatenate all the attention heads



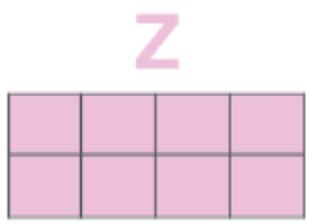
2) Multiply with a weight matrix W^O that was trained jointly with the model

X



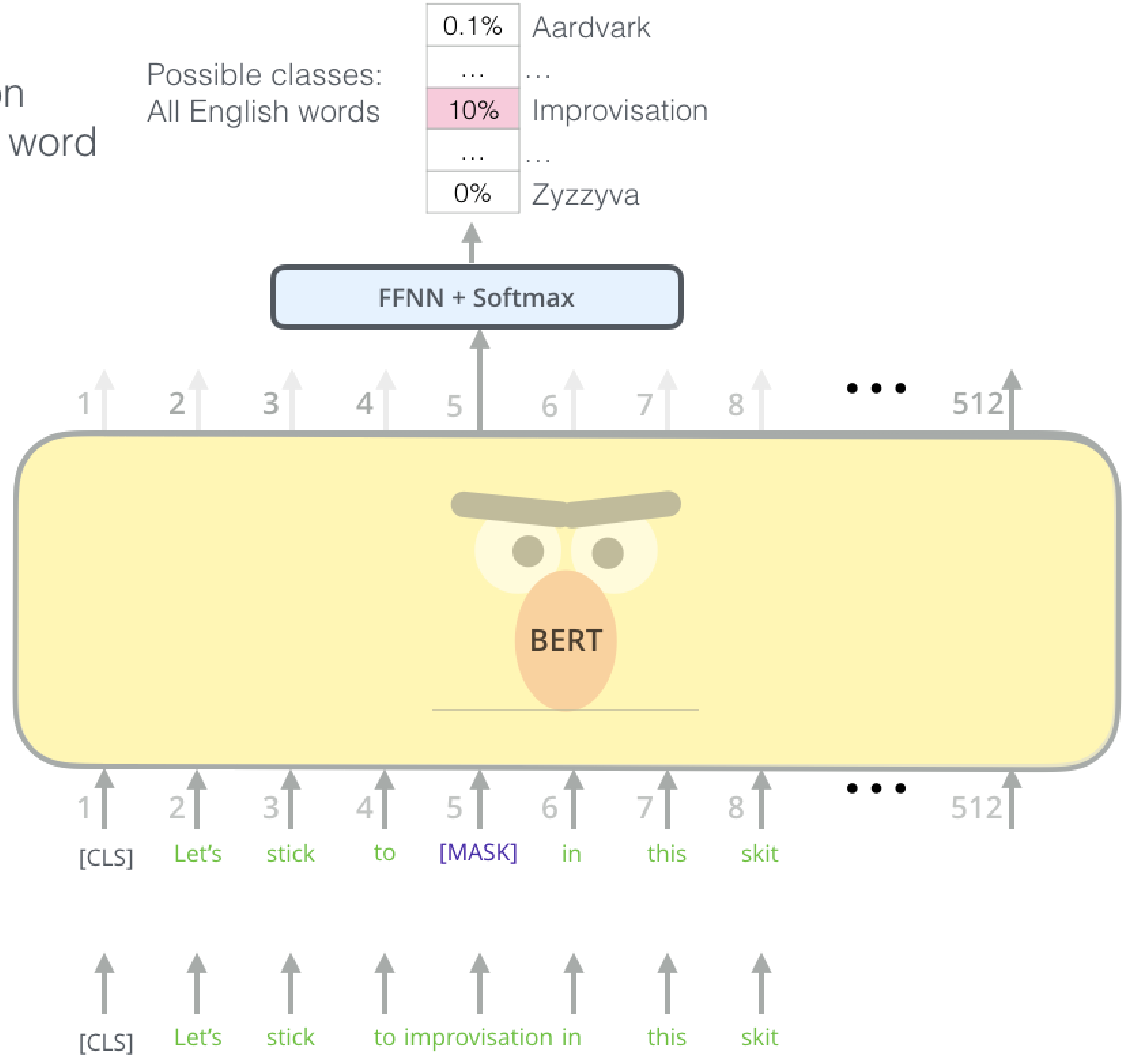
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



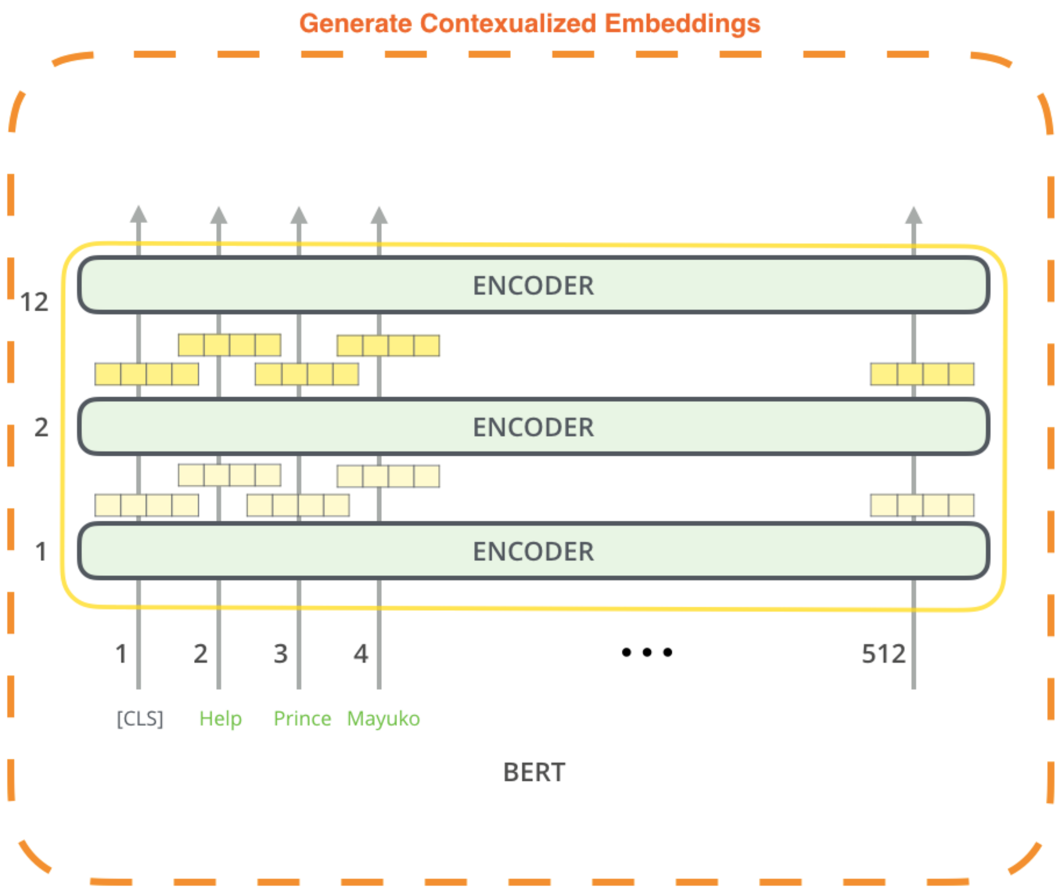
BERT

Use the output of the masked word's position to predict the masked word

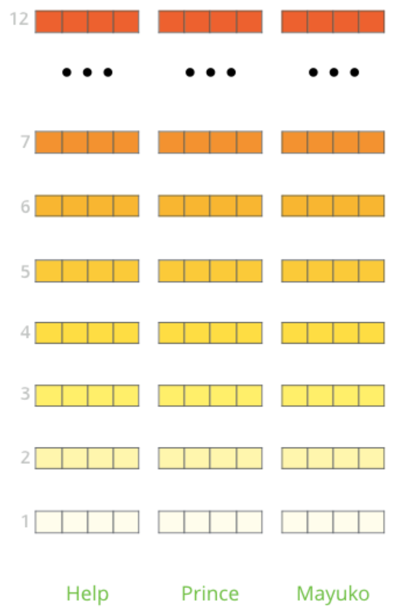


Randomly mask 15% of tokens

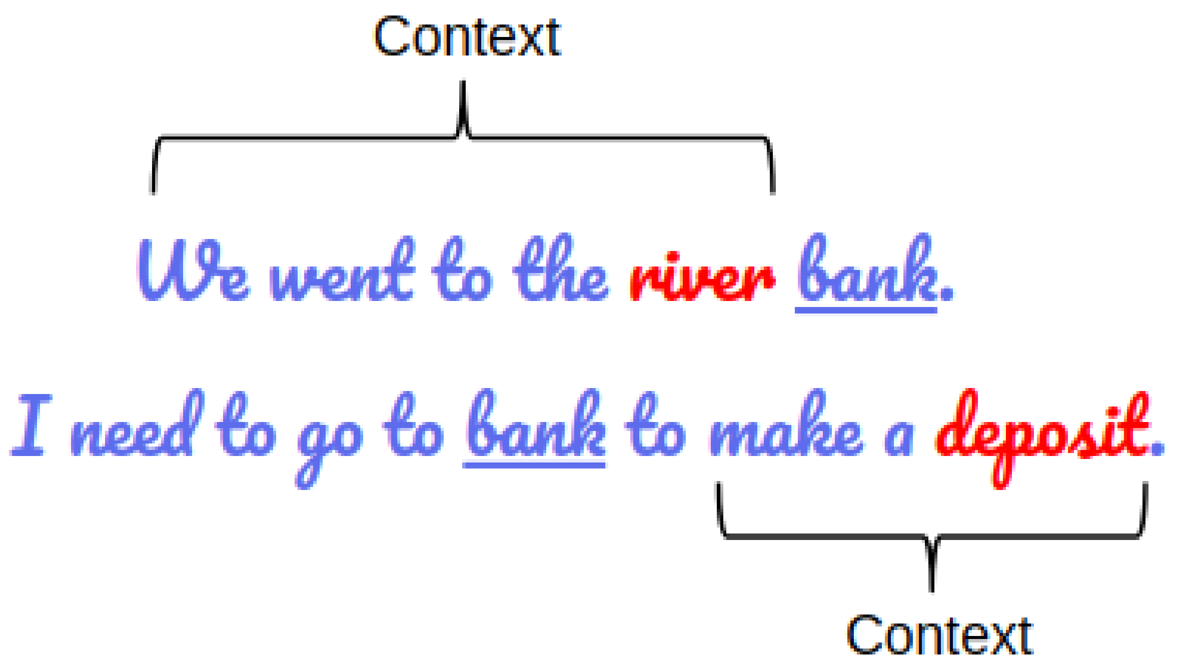
Input



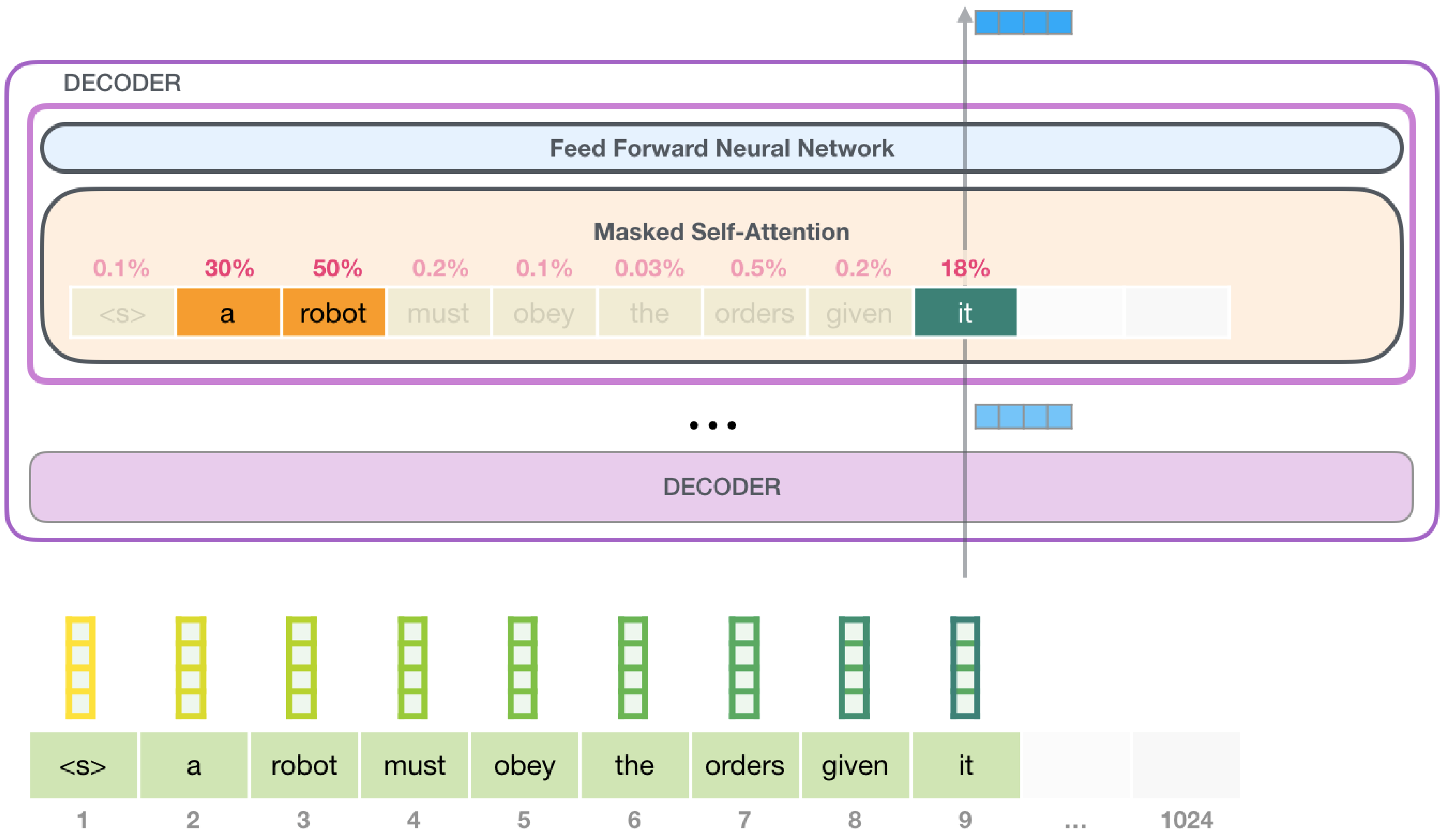
The output of each encoder layer along each token's path can be used as a feature representing that token.



But which one should we use?



GPT



DECODER
Feed Forward Neural Network
Masked Self-Attention
0.1%
30%
50%
0.2%
0.1%
0.03%
0.5%
0.2%
18%
<s>
a
robot
must
obey
the
orders
given
it
• • •
DECODER
<s>
a
robot
must
obey
the
orders
given
it
1
2
3
4
5
6
7
8
9
...
1024

GPT-2

1.5 billion parameters

GPT-3

175 billion parameters.
Also got rid of gradient descent.

Reformer – the next state of the art?

https://ai.googleblog.com/2020/01/reformer-efficient-transformer.html

projects

parser

```
NONTERMINALS = """
S -> N V
"""
```

The only grammar you need to model is the grammar in all the sentences.

The specifics of the design is up to you!

parser

```
def preprocess(sentence):
    """
    Convert `sentence` to a list of its words.
    Pre-process sentence by converting all characters
    to lowercase and removing any word that does not
    contain at least one alphabetic character.
    """
```

If you get stuck, refer to the lecture source code!

parser

```
def np_chunk(tree):
    """
    Return a list of all noun phrase chunks in the
    sentence tree. A noun phrase chunk is defined as
    any subtree of the sentence whose label is "NP"
    that does not itself contain any other noun phrases
    as subtrees.
    """
```

nltk.tree is all you need here! The .subtree() method will be helpful here!

questions

```
def load_files(directory):
    """
    Given a directory name, return a dictionary mapping
    the filename of each `.txt` file inside that
    directory to the file's contents as a string.
    """
```

Remember to check for a file’s `.txt` extension before you import!

questions

```
def tokenize(document):
    """
    Given a document (represented as a string), return
    a list of all the words in that document, in order.
    Process document by converting all words to
    lowercase and removing any punctuation or English
    stopwords.
    """
```

Hint: what case are the stopwords?

questions

```
def compute_idfs(documents):
    """
    Given a dictionary of `documents` that maps names
    of documents to a list of words, return a
    dictionary that maps words to their IDF values. Any
    word that appears in at least one of the documents
    should be in the resulting dictionary.
    """
```

If you are stuck, refer to lecture source code!

questions

```
def top_files(query, files, idfs, n):

def top_sentences(query, sentences, idfs, n):
```

These are similar in implementation, but what is returned should be different! Hint, you can sum a list! The str.count() method can also be helpful here! Also, be mindful when you want words to repeat in your calculations and when you don’t.

Yay! You Made It!