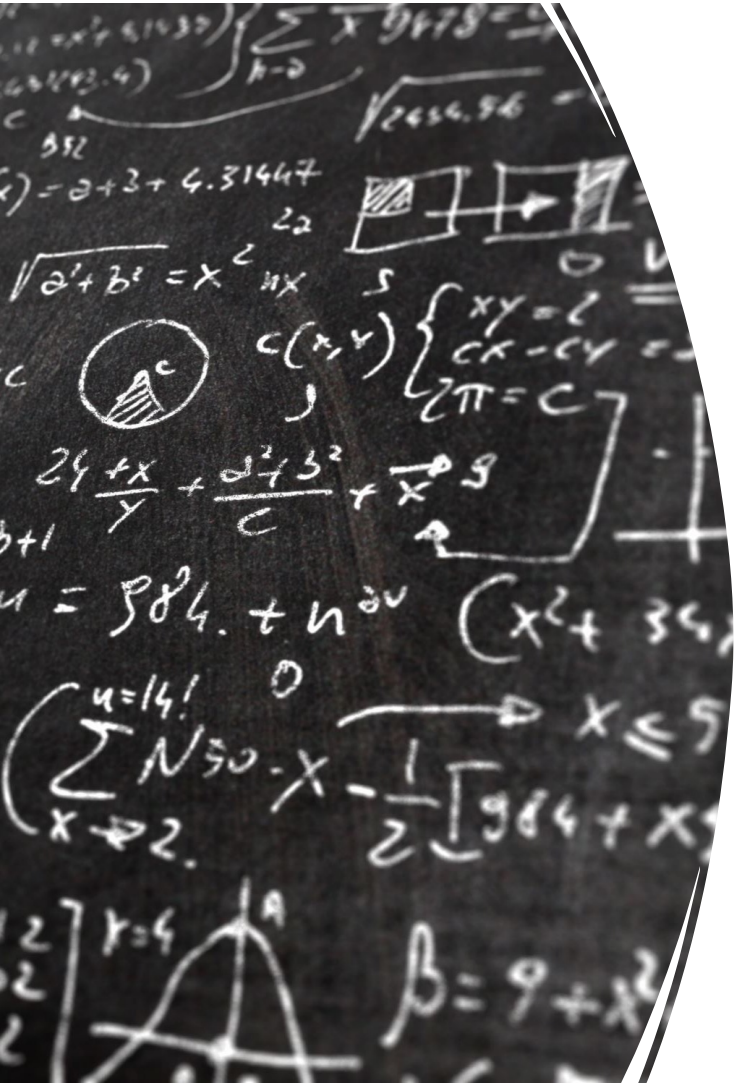


CS 383 – Computational Text Analysis

Lecture 17
Transformer models
Pre-training & Fine-tuning

Adam Poliak

03/22/2023

Slides adapted from Daniel Khashabi, Chris Manning

Announcements

- Final project ideation
  - Mandating partner, must work with a partner!
  - Due this Friday 03/25
  - I'm deleting all single-person submissions tonight then re-opening the submission

- HW06:
  - I'm writing it from scratch (mostly)
  - Downloading tweets
  - Building classifiers
  - Asking a research question
    - Playing with directionality in an RNN
    - Change dimension sizes of hidden layers
    - Change which word embeddings to use
    - Your choice!
  - Partner
  - Will have >2 weeks

Twitter API for next HW

1. Create a Twitter developer account https://developer.twitter.com/
2. Go to https://developer.twitter.com/en/apps and log in with your Twitter user account.
3. Click “Create an app”
4. Fill out the form, and click “Create”
5. A pop up window will appear for reviewing Developer Terms. Click the “Create” button again.

Instructions from http://socialmedia-class.org/twittertutorial.html

Look here for instructions on how to use Tweepy: https://github.com/BC-COMS-2710/summer21-material/blob/master/demo/Demo13.ipynb

Midterm - Format

Multiple Choice

Short Answer

Problems to work out by hand

Midterm Topics

Unsupervised approaches:

- N-gram Language Modeling
  - Backoff, smoothing
- Document Term Matrix
- Dimensionality Reduction
- Topic Modeling – LDA

Supervised Machine Learning:

- Classification & Regression
- SGD
- Neural Networks
  - FNN
  - Backprop
- RNNs
- Attention

Midterm

- Create a computation graph and update weights based on a loss
- Use Gibbs sampling to update topic assignments
- Some more but not sure yet

Machine Learning in a nutshell

In a ML model, what are we training?

- **Parameters!**

How do we train parameters in supervised learning?

train parameters == figure out values for the parameters

- Update weights by using them to make predictions and seeing **how far off our predictions** are
  - **Loss function!**

Algorithm to learn weights?

- **SGD**
- Others exist but not covering them

Outline

Attention & Self-attention recap

Transformer

Pytorch demo (if time)

Attention

What problem does it solve?

- Bottleneck from having single sentence representation

How does it work?

- Instead of looking at just the sentence representation, combine it with a new attention vector for each prediction

Attention vector/output:

- A weighted average off the outputs of the hidden layer in the encoder

Self-attention

What problem does it solve?

- Static word representations

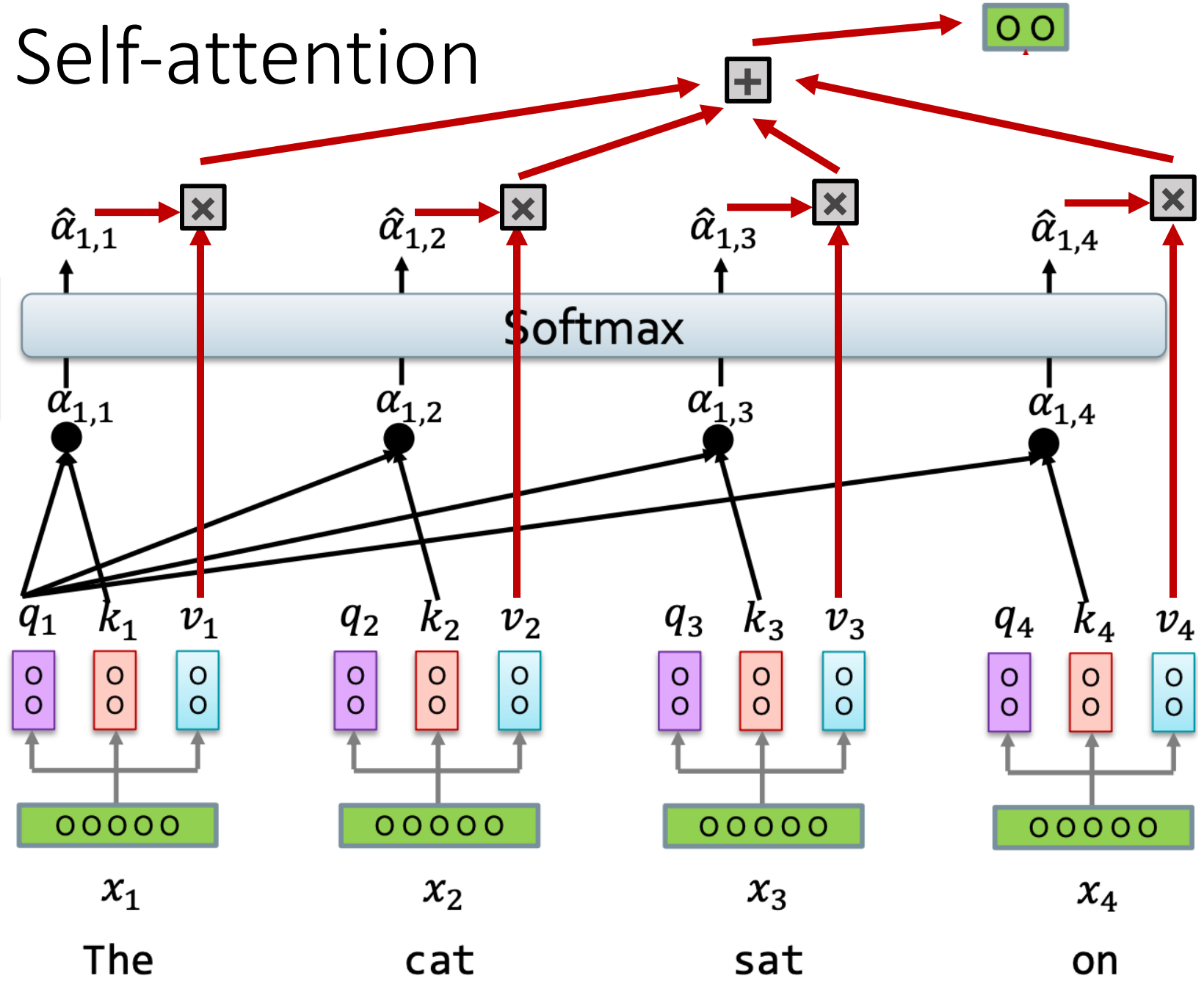
How does it work?

- Uses query's and key's to determine how much to attend to other words
  - Uses the specific word's query vector and the other words' key vector
  - Uses value vector to represent other words

Attention vector/output:

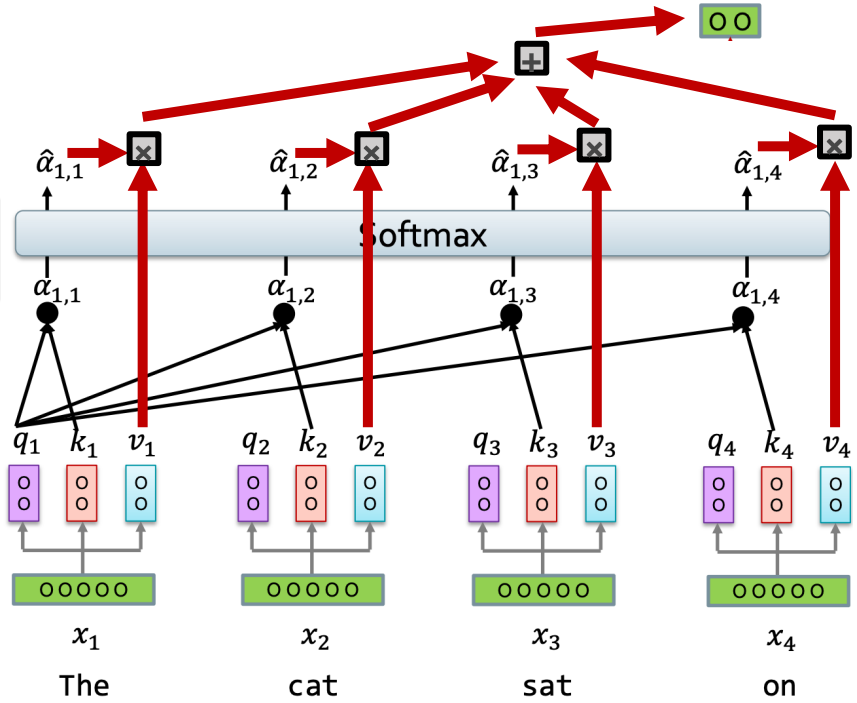
- A weighted average off the other words' value vectors
- How do we get a weighted average?
  - Softmax

Self-attention



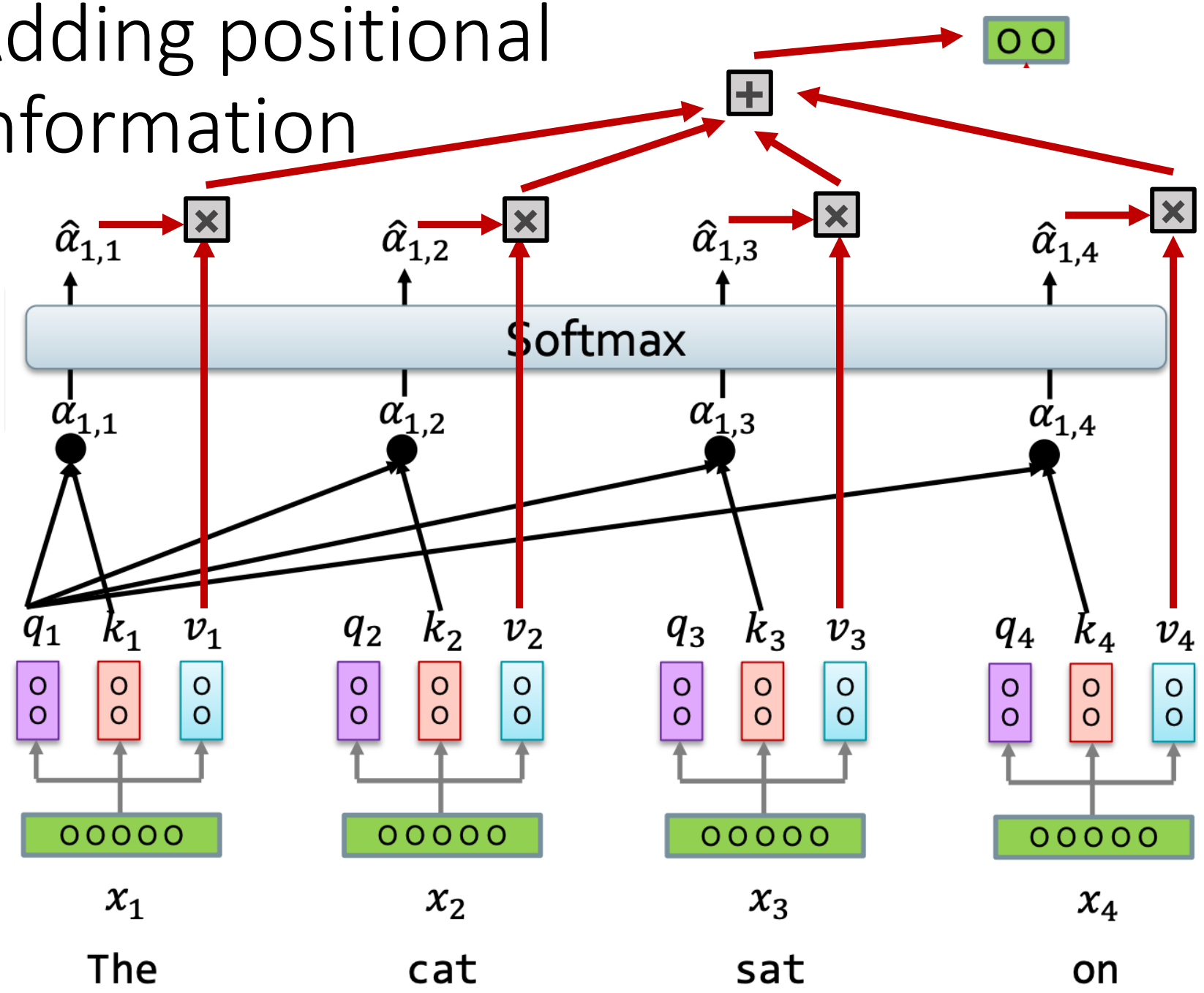
Self-attention

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

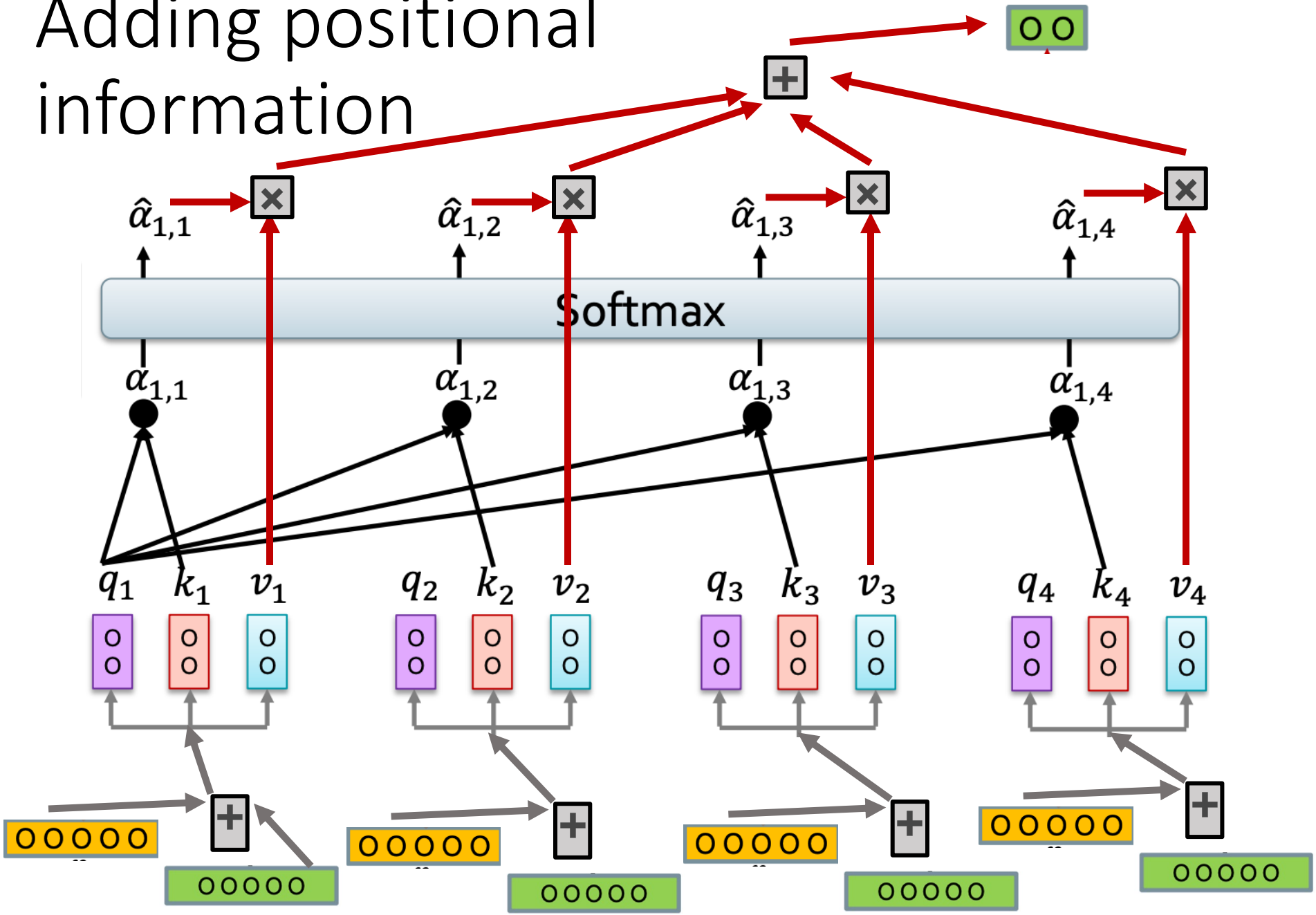


This is the main idea behind a **transformer**

Adding positional information

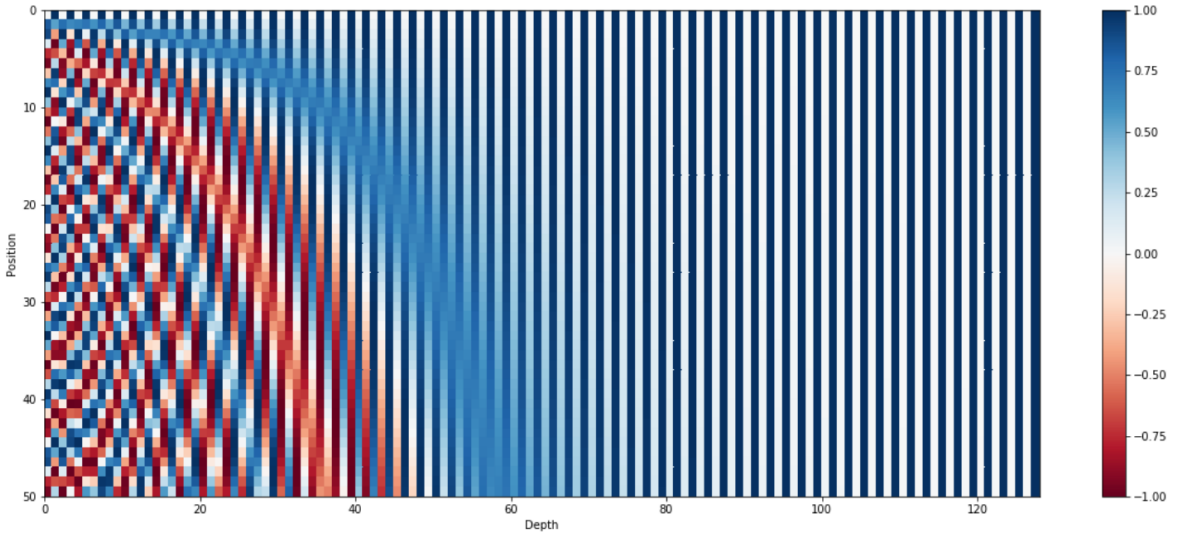


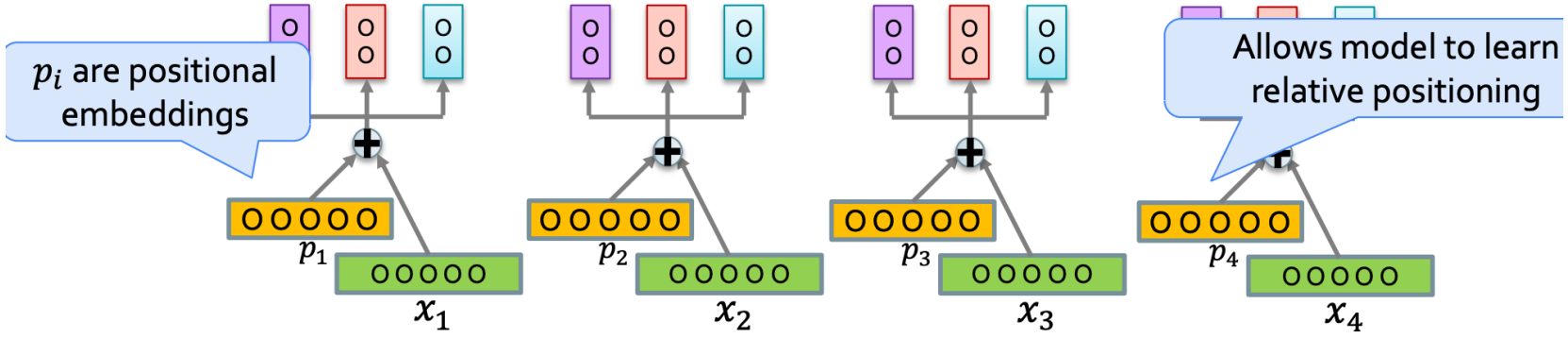
Adding positional information



Adding positional information

An approach:
Sine/Cosine encoding





Positional Representations

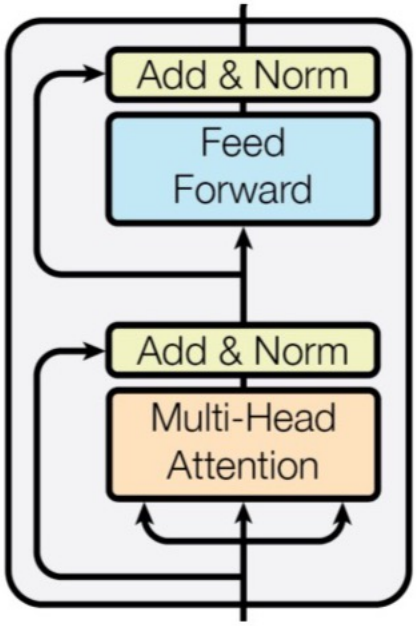
- Just go from 1 … n
  - Con:

Self Attention

Given input x

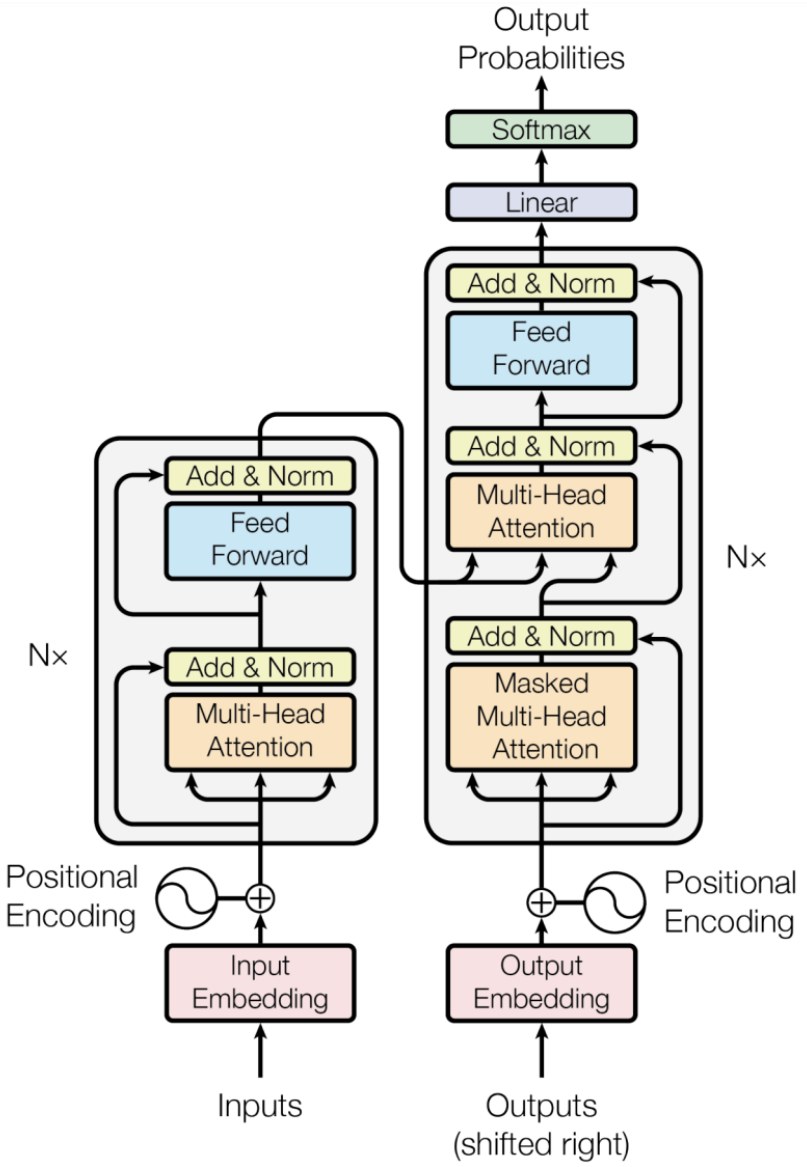
$$Q = W^q x$$
$$K = W^k x$$
$$V = W^v x$$

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



Transformer

Model form
Attention is all you need



Outline

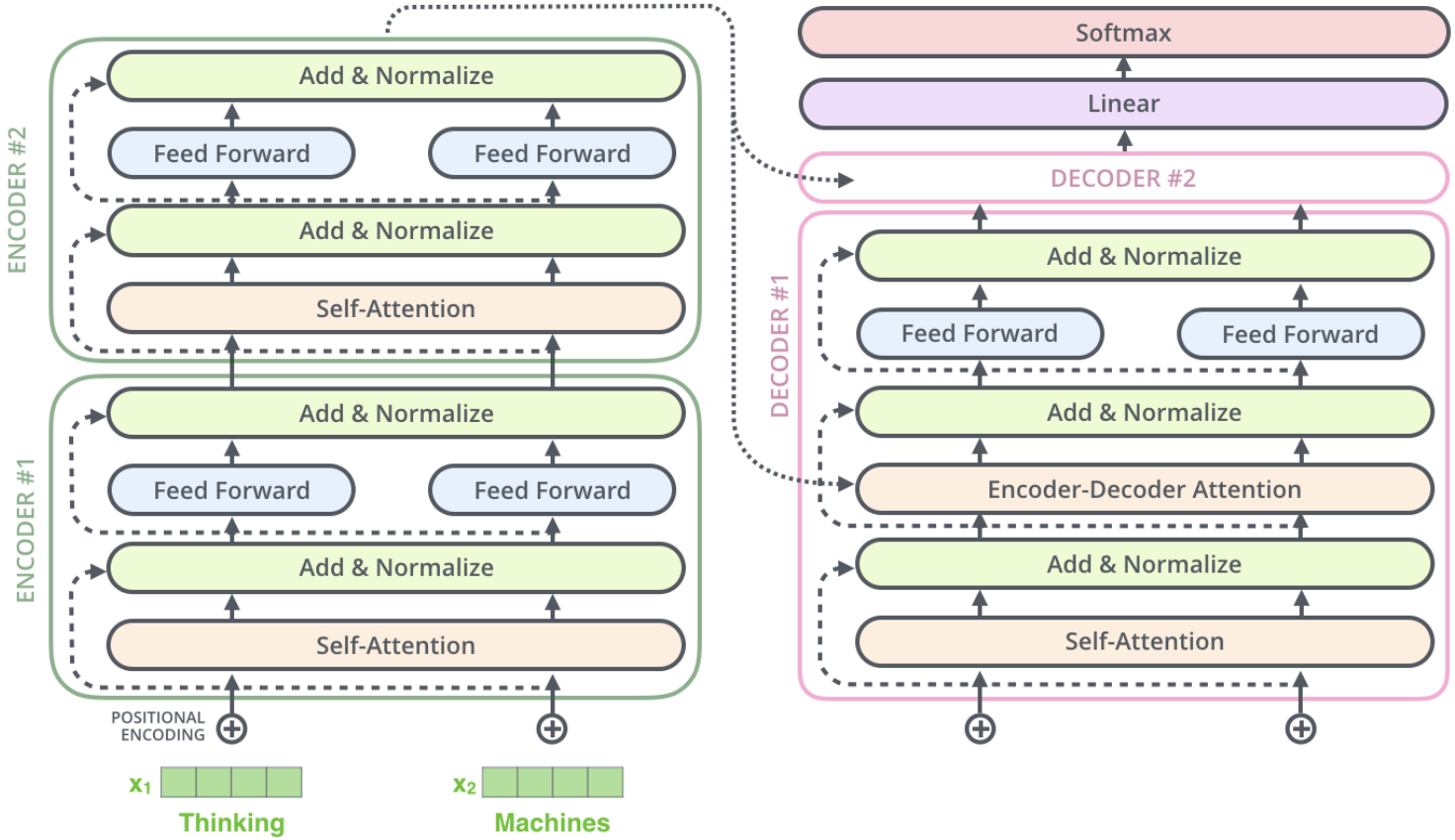
Attention & Self-attention recap

Transformer

Pytorch demo (if time)

Transformer (from *Attention is all you need* 2017)

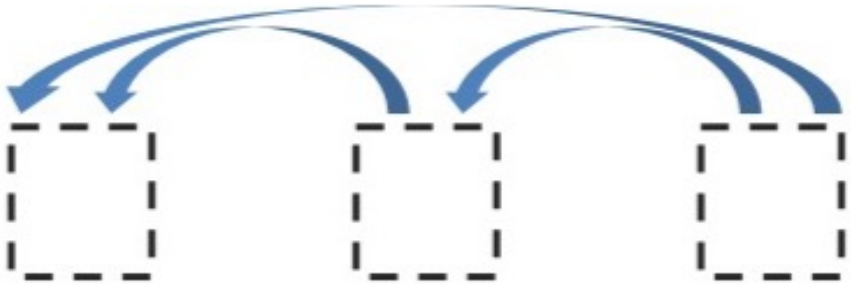
- Encoder-decoder with attention modules



3 forms of attention

Encoder-Decoder Attention

Encoder Self-Attention



MaskedDecoder Self-Attention

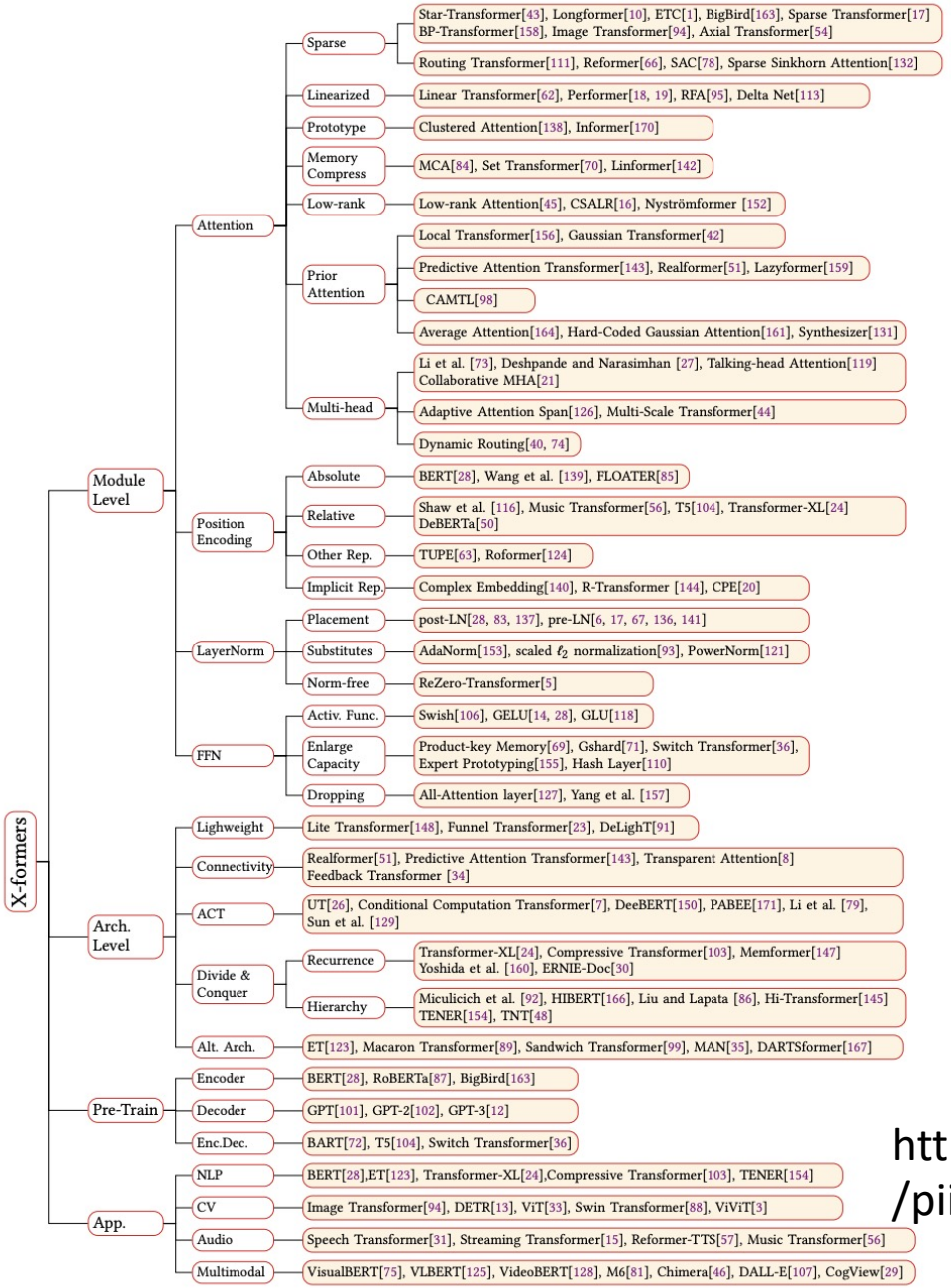


Fig. 3. Taxonomy of Transformers

https://www.sciencedirect.com/science/article/pii/S2666651022000146

BERT (2018)

Contributions:

- Bi-directional self-attention encoder rather than BiLSTM
- Brought Transformers beyond Machine Translation
  - Attention All You Need (Summer 2017)
  - BERT (October 2018)

Side note:

- Attention was introduced by Bahdanau et al. (2015) for MT

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

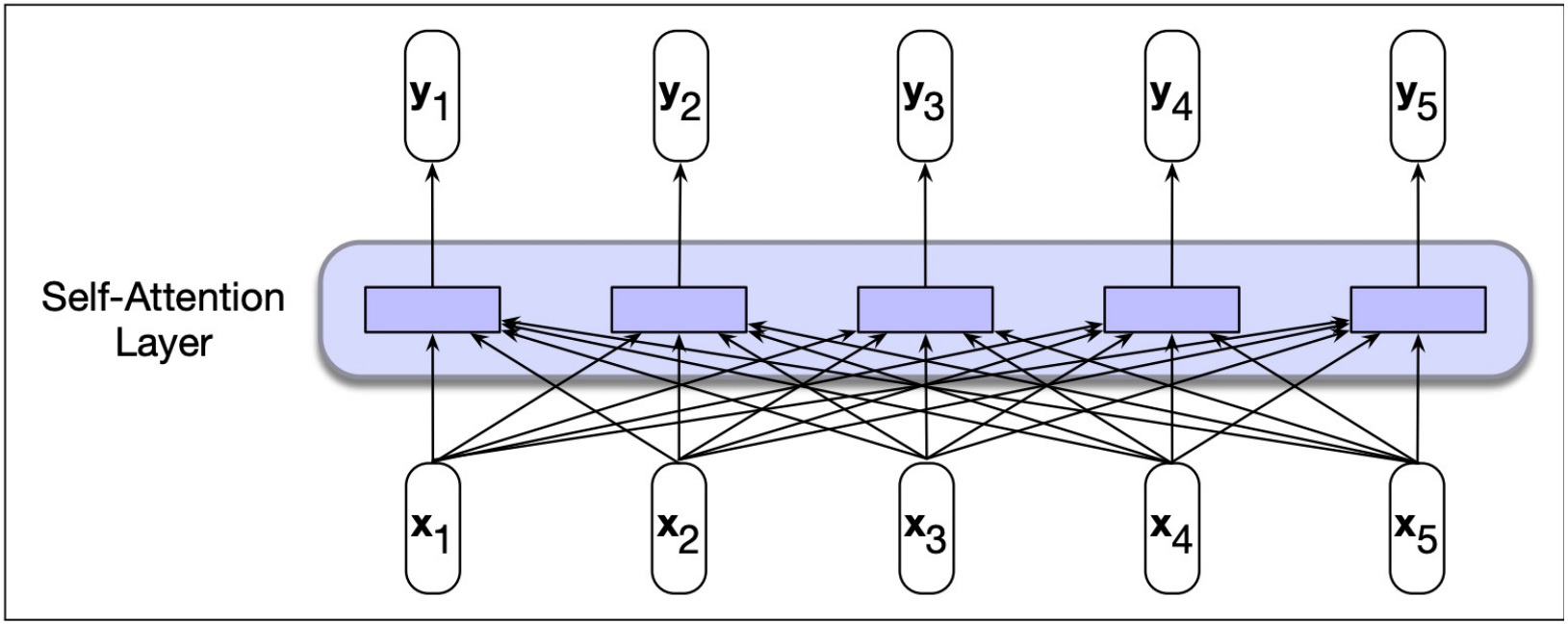
There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers

BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers

Uses just transformer encoder, not decoder

Bi-directional self-attention



Training BERT (1/2)

Since it uses both-directions, language modeling would be cheating

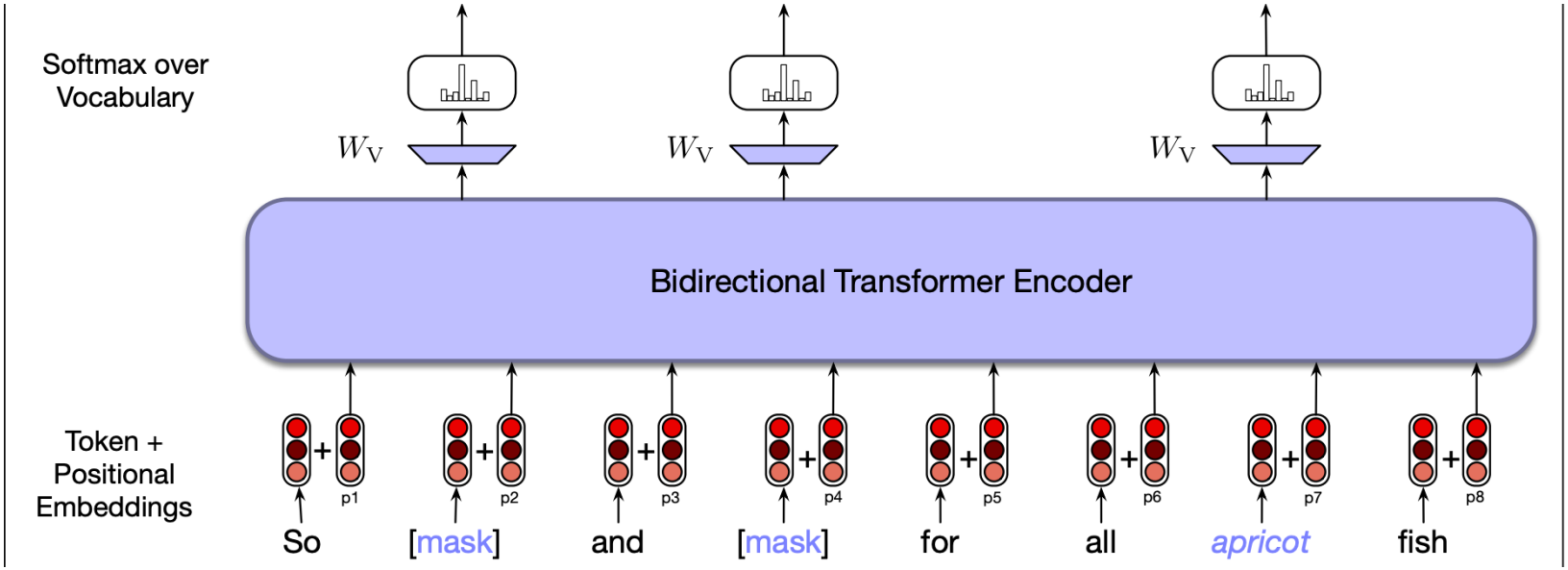
So, its trained on a task called
Masked Language Modeling

Masked Language Modeling

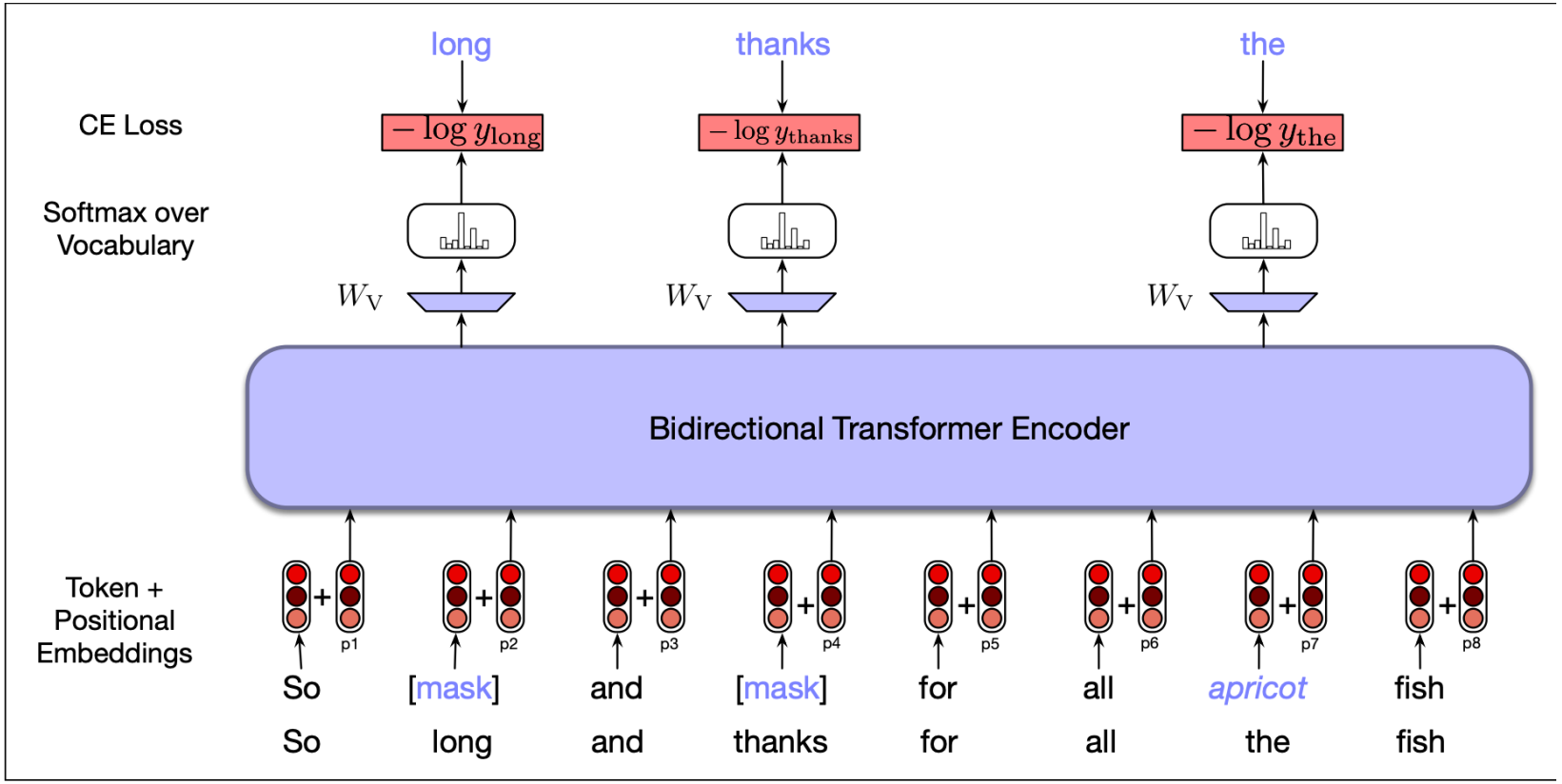
- Predict the hidden words based on the context

- What happens if we mask too many words?
  - Can’t make predictions

Masked Language Modeling



Masked Language Modeling



MLM in Bert

15% of tokens in the training set were sampled

- 80% of which were masked
- 10% were swapped with other words
- 10% were kept the same

Training BERT (2/2)

If Masked Language Modeling tries to predict the missing word, what might another task/object be if we move the unit of analysis beyond words?

Next-Sentence Prediction

Next Sentence Prediction

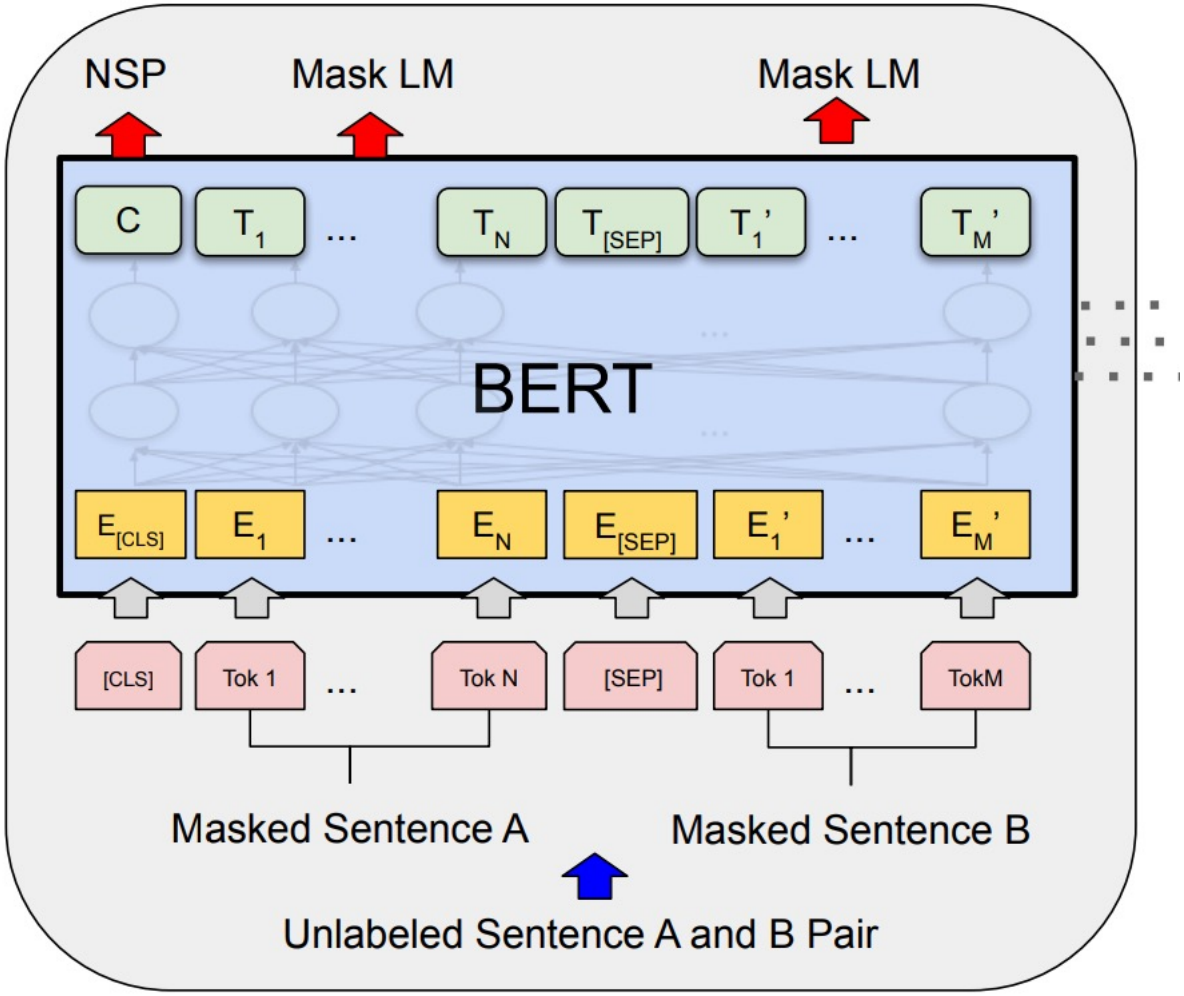
Predict if Sentence B follows Sentence A

Sentence A = The man went to the store.
Sentence B = He bought a gallon of milk.
Label = IsNextSentence

Sentence A = The man went to the store.
Sentence B = Penguins are flightless.
Label = NotNextSentence

Learns ordering of sentences

Training BERT



Training BERT

Training data > 3.3B words

- Wikipedia (2.5B words) + BookCorpus (800 M words)

Optimizer:

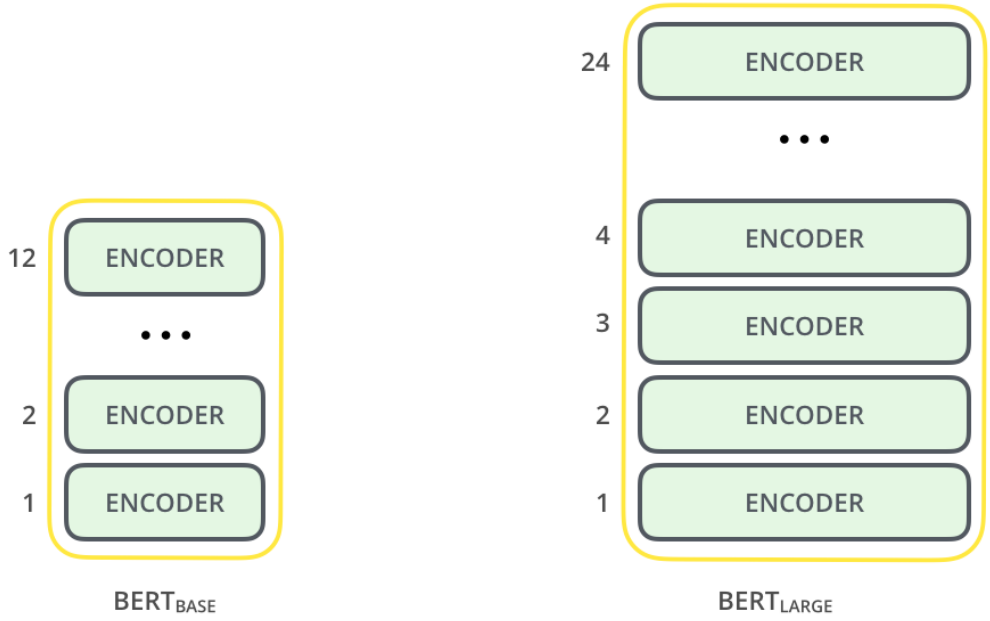
- AdamW, linear decay of the learning rate

Time

- 4 days

2 Versions of BERT

- Base with 12 encoder layers
- Large with 24 encoder layers



BERT (2018)

Contributions:

- Bi-directional self-attention encoder rather than BiLSTM
- MLM & NSP objective
- Large amounts of data

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

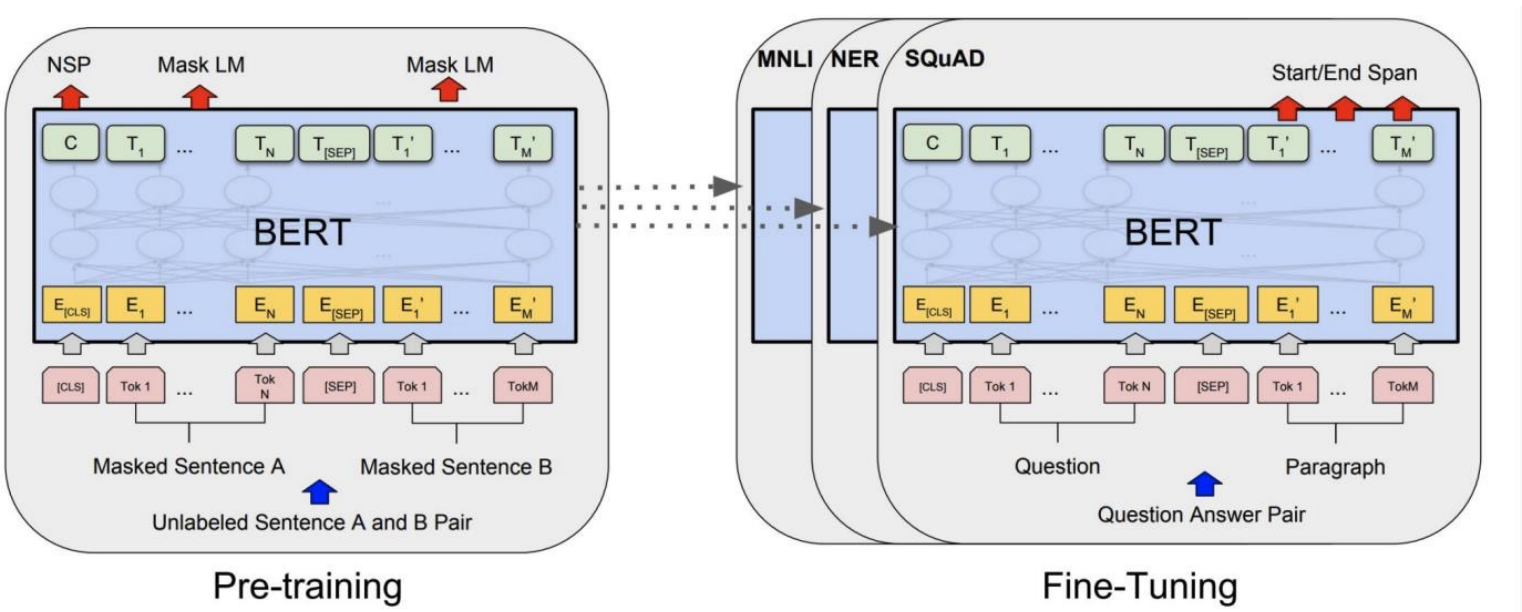
BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers

Fine-tuning

- Used pre-trained models in other tasks
- Initialize model with pre-trained model's parameters
- Update parameters using labeled data from new task



Since BERT

RoBERTa (Liu et al., 2019)

- Only Masked Language Model training
- Trained on 10x data as BERT
- Stronger performance on wide range of tasks

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

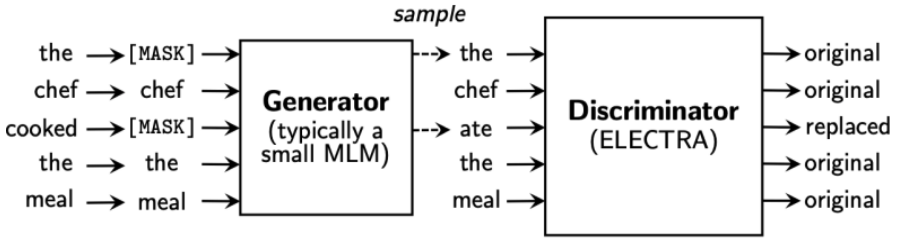
Since BERT

ALBERT (Lan et. al. 2020)

- Increased model size
- Shared parameters across layers

ELECTRA (Clark et. al. 2020)

- Generator and Discriminator pre-training
  - Normal MLM but then choose which of the words were masked



Since BERT

Incorporate longer context

Longformer, Big Bird, …

Multilingual BERT

Domain Specific BERT

SciBERT, BioBERT, TwitterBERT, FinBERT, …

Smaller versions of BERT

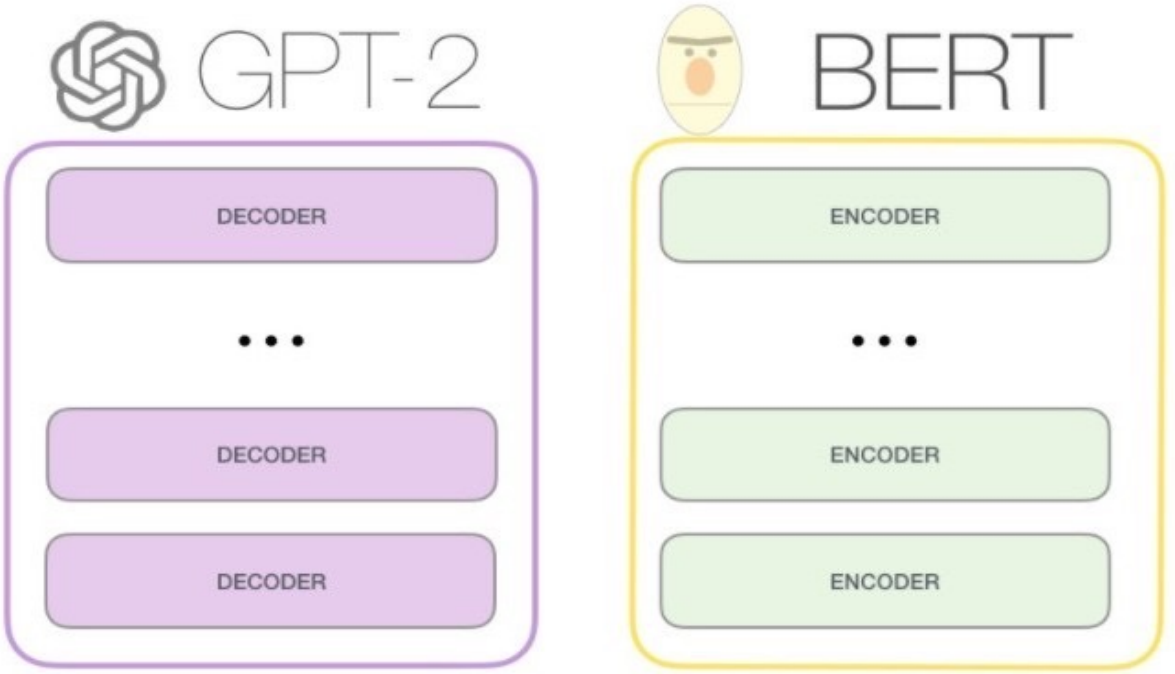
DistillBERT, TinyBert, …

BERT Summary

- Primarily an encoder (w/ classifiers slapped on top)
- Transformer networks trained on massive data
- Learn contextual representations of words
- Pre-train a large LM and then fine-tune for specific tasks (just replace the top layer)
- Not designed for text generation

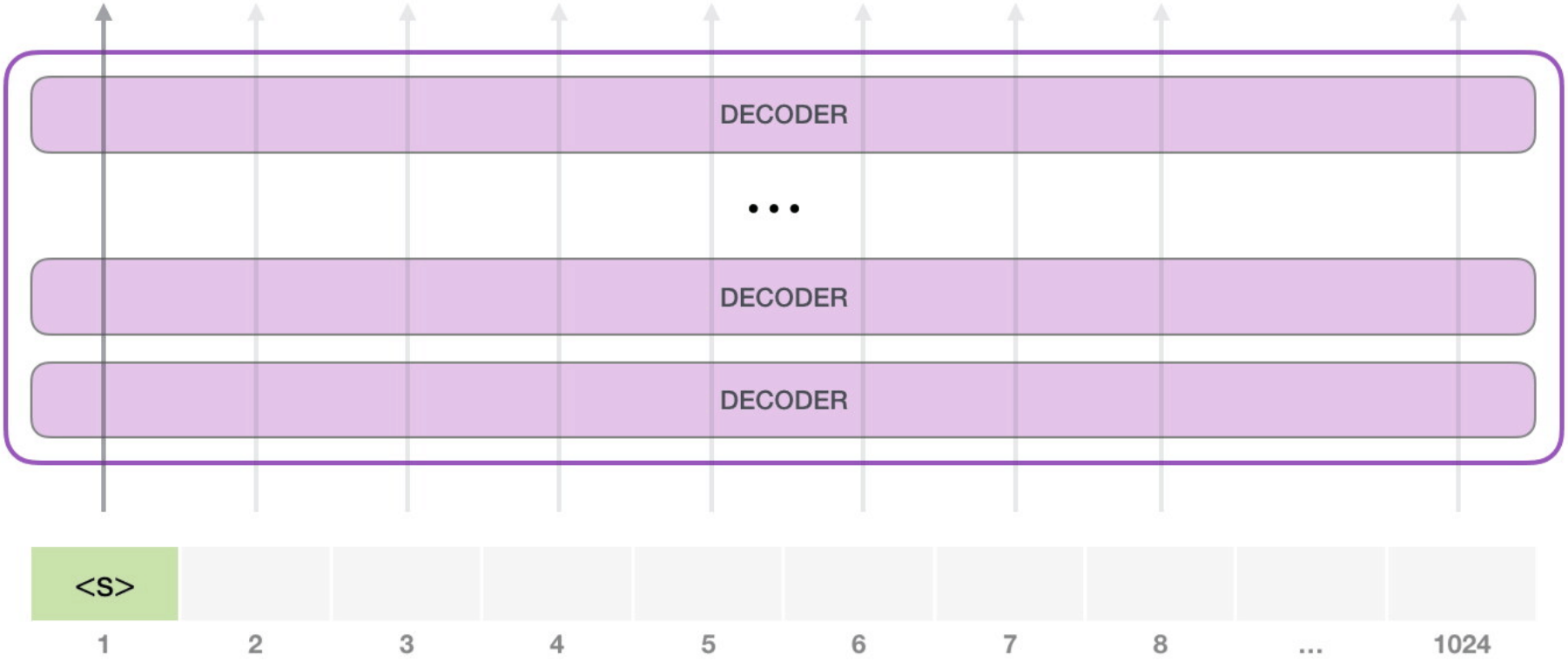
GPT-2

Instead of using encoders, we can stack decoders

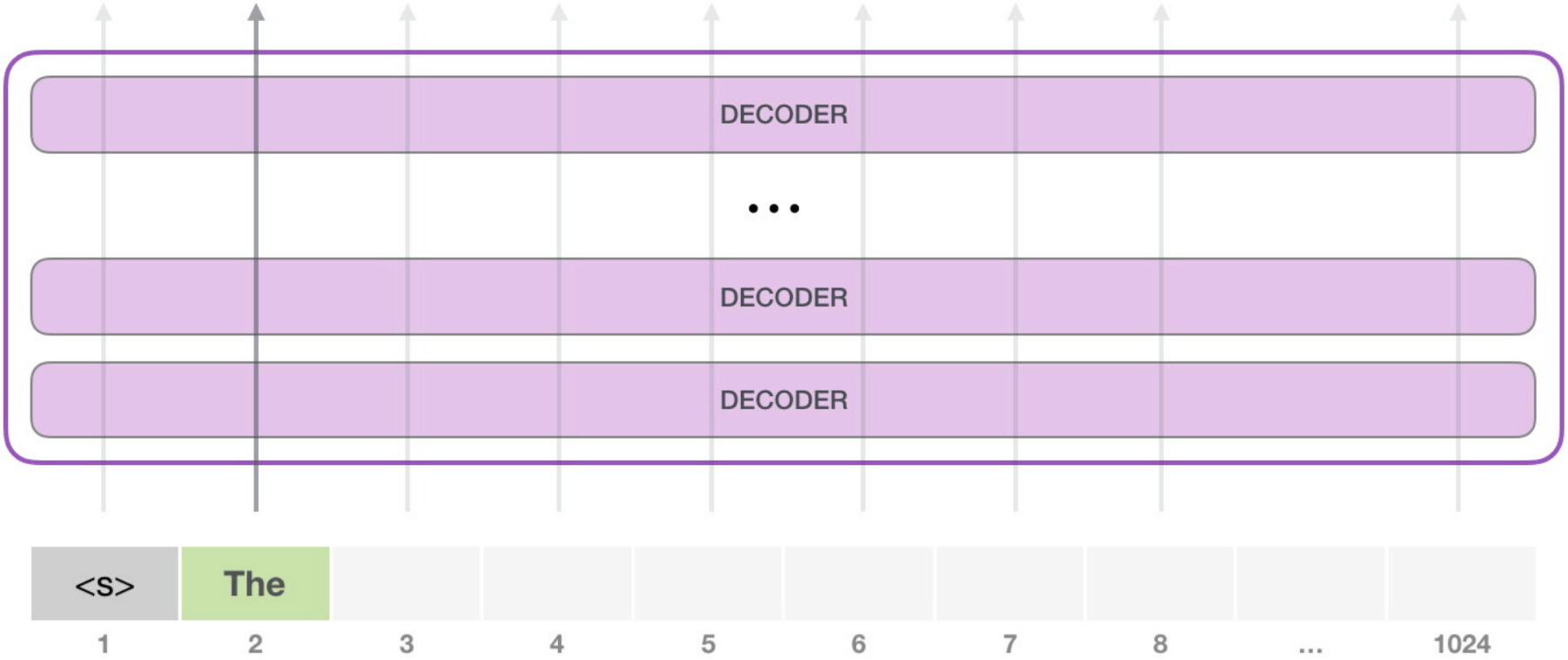


http://jalammar.github.io/illustrated-gpt2/

Decoding



Decoding



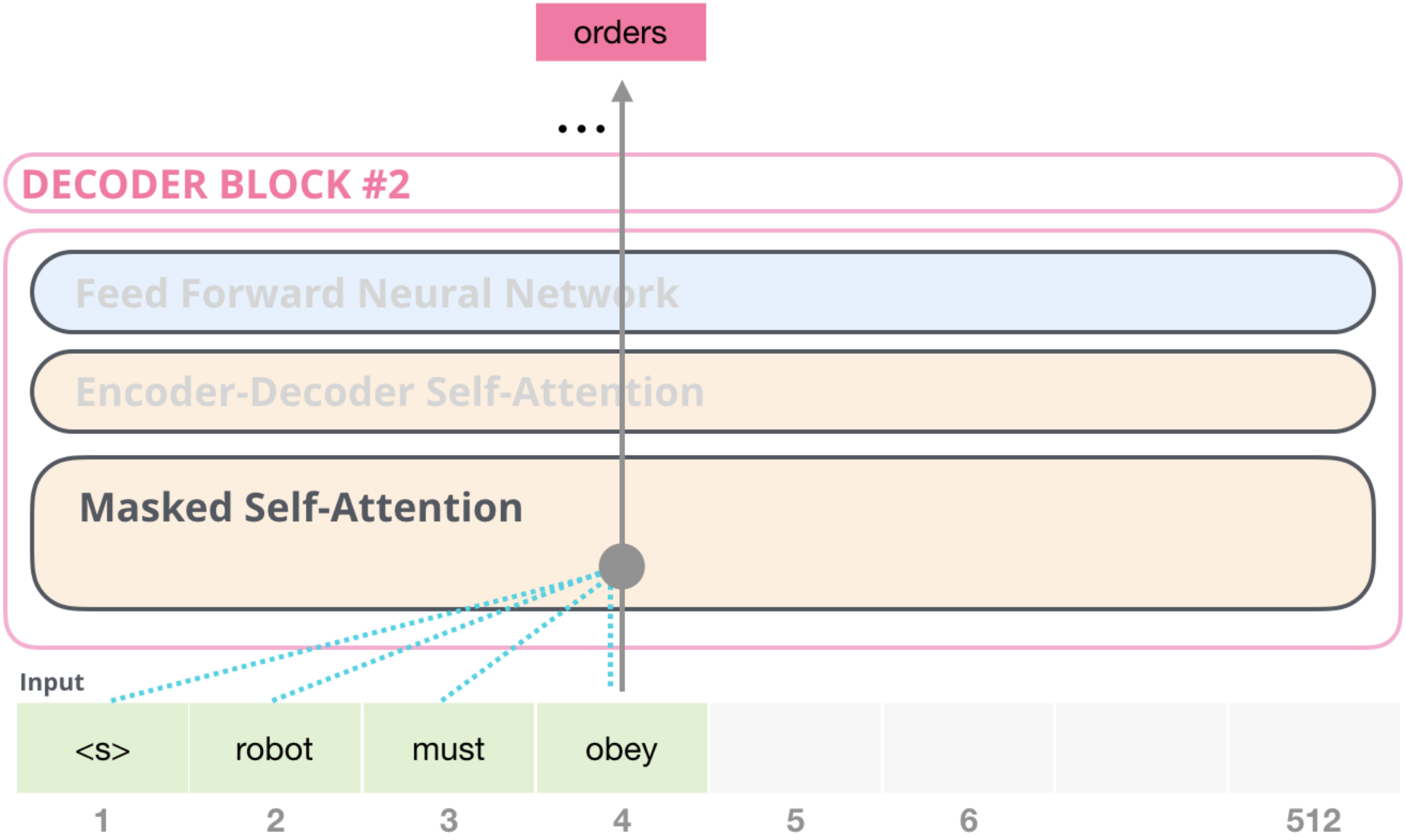
Auto-regressive

When making a new prediction, incorporate the prediction of the previous model

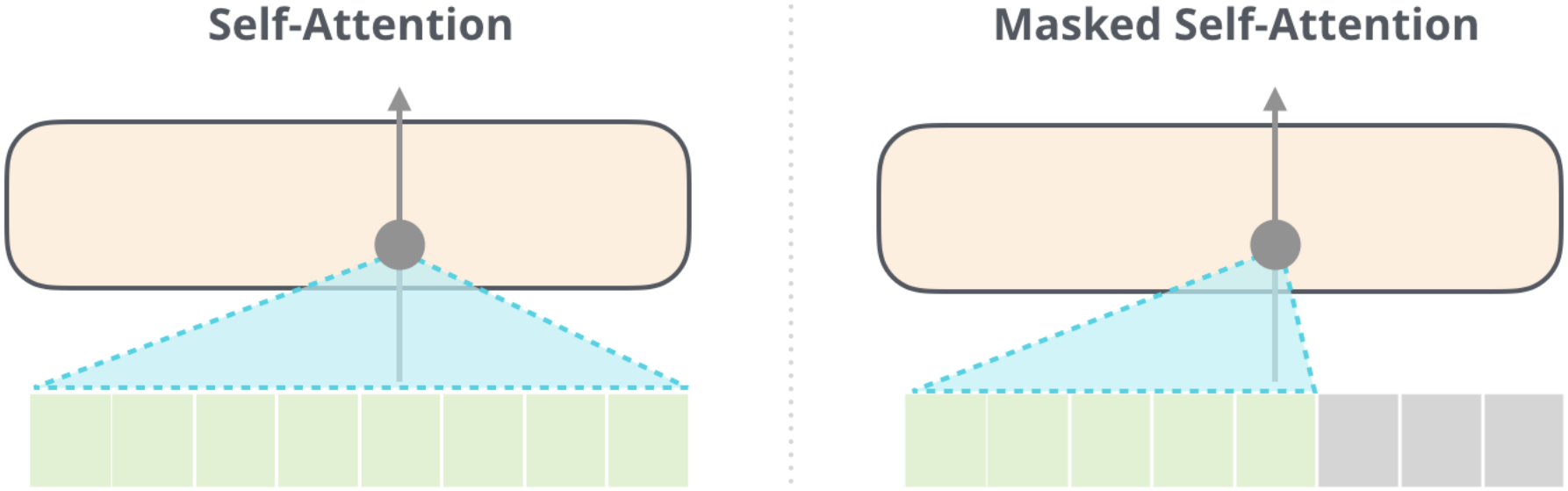
For LM decoding, this means:

- when predicting the next word, consider what the model just generated
- **Question:** What do we lose?
  - Ability to look both ways

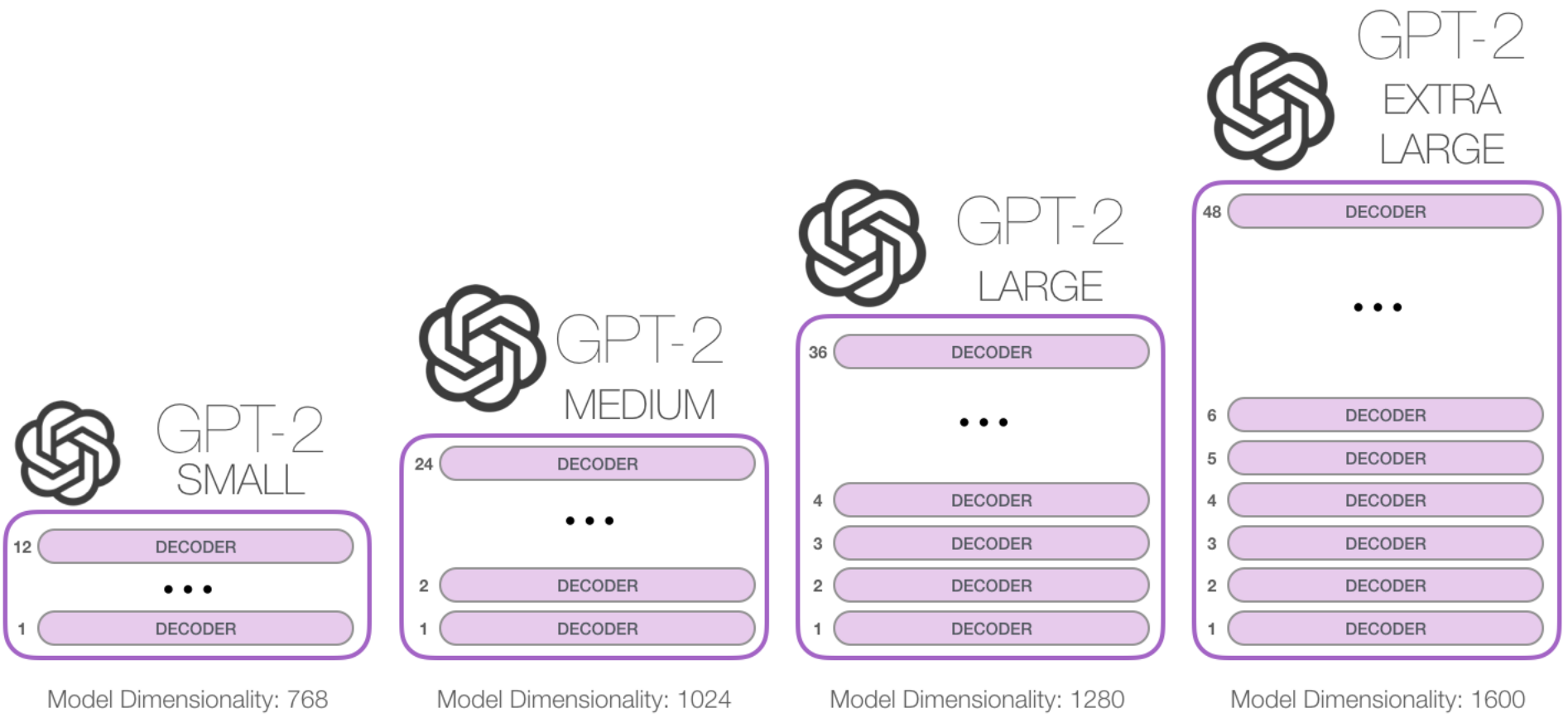
Self-Attention in GPT



BERT vs GPT



GPT-2 sizes


GPT-2
SMALL
12
DECODER
1
DECODER
Model Dimensionality: 768
GPT-2
MEDIUM
24
DECODER
2
DECODER
1
DECODER
Model Dimensionality: 1024
GPT-2
LARGE
36
DECODER
4
DECODER
3
DECODER
2
DECODER
1
DECODER
Model Dimensionality: 1280
GPT-2
EXTRA
LARGE
48
DECODER
6
DECODER
5
DECODER
4
DECODER
3
DECODER
2
DECODER
1
DECODER
Model Dimensionality: 1600

More on GPT

Highly recommend Jay Alammar's blog post
http://jalammar.github.io/illustrated-gpt2/

Next class

Evaluation metrics for classification beyond accuracy:

- Chapter 4.7

Decoding beyond greedy approaches

- Beam Search – Chapter 10.4

Afterwords (Thursday and beyond), hypothesis testing

- How do I know my model is better that your model
- How do I know what I discover isnt just specific to my sampled data or due to chance