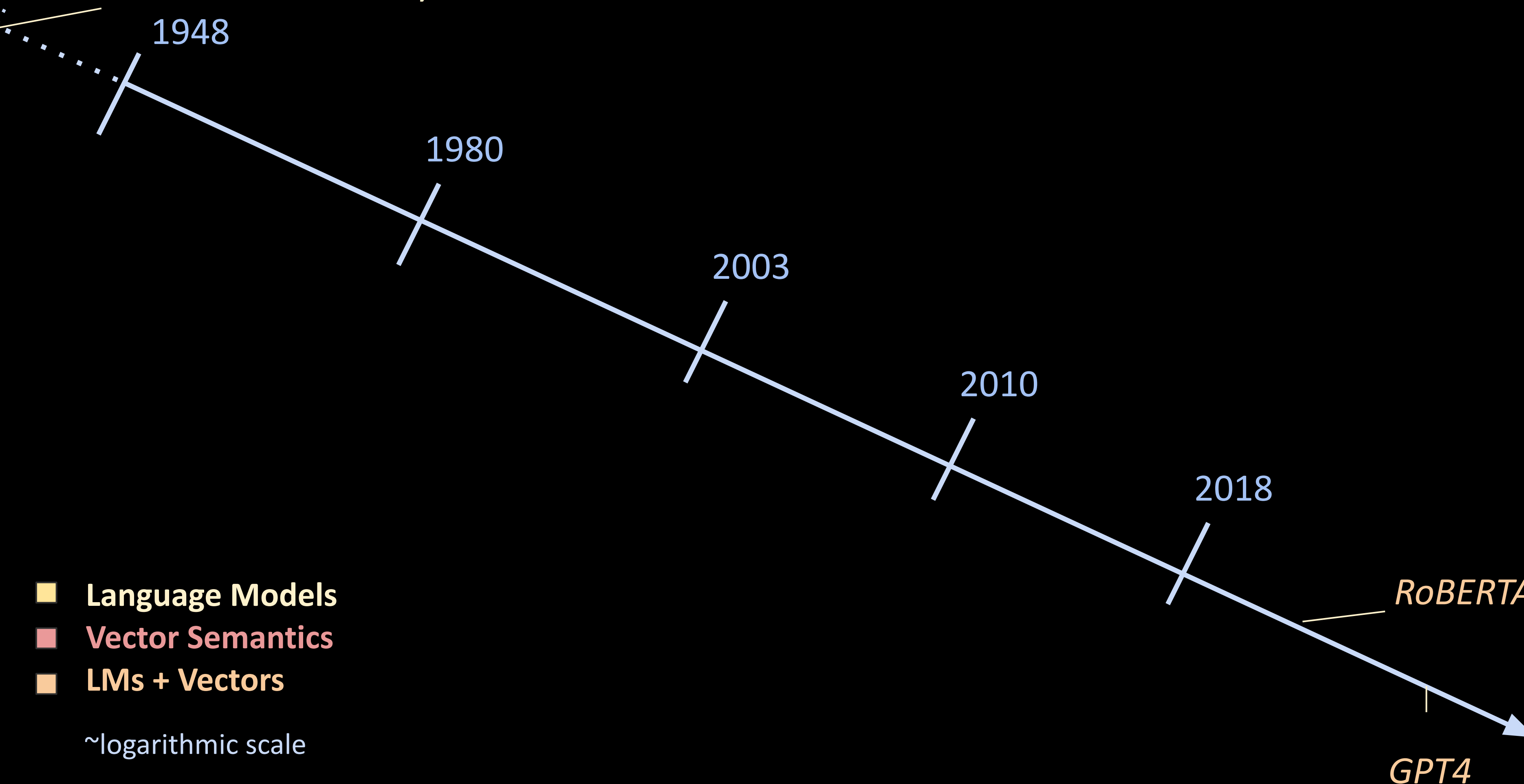
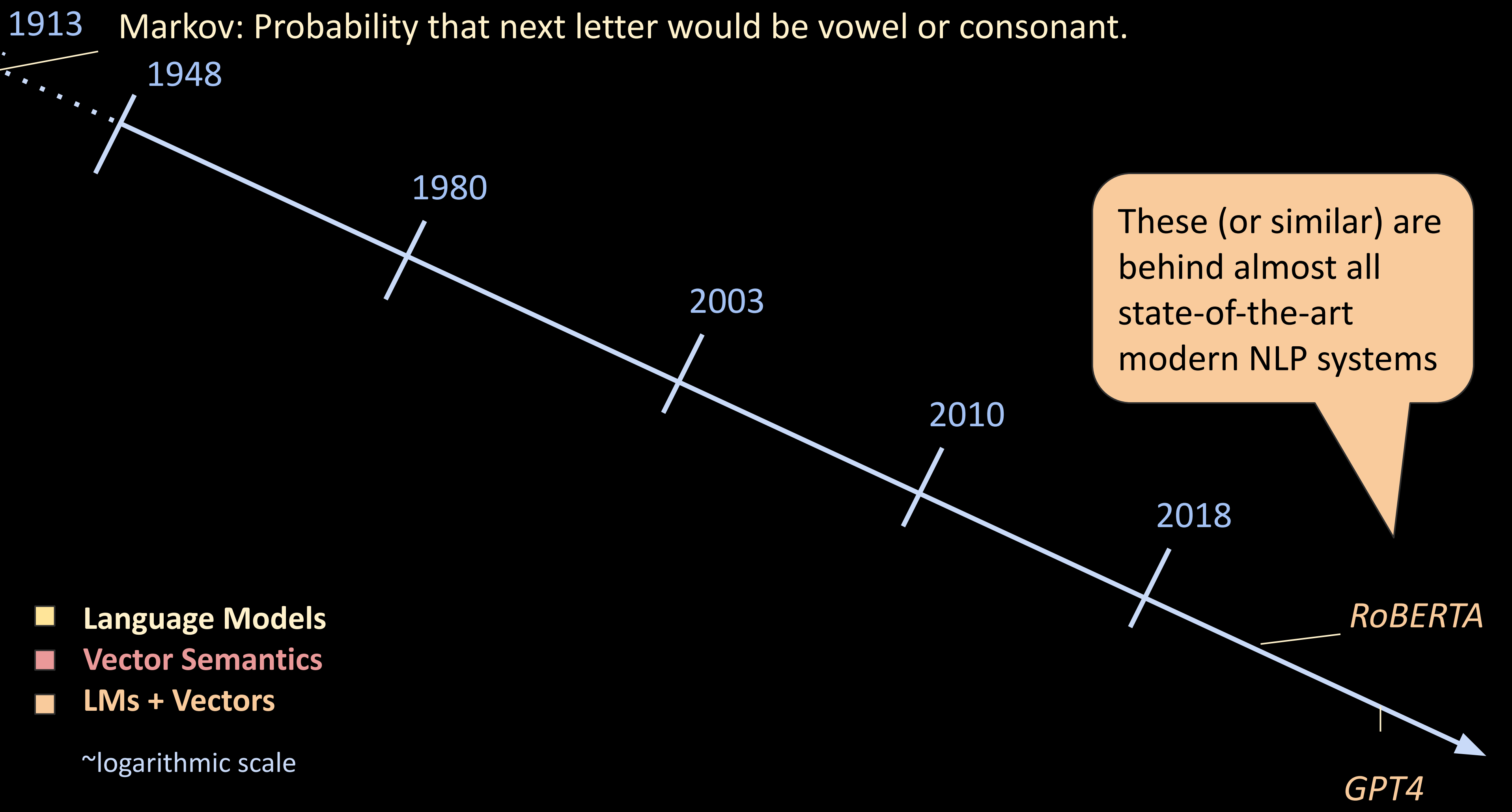
Transformer Language Models

Timeline: *Language Modeling* and *Vector Semantics*

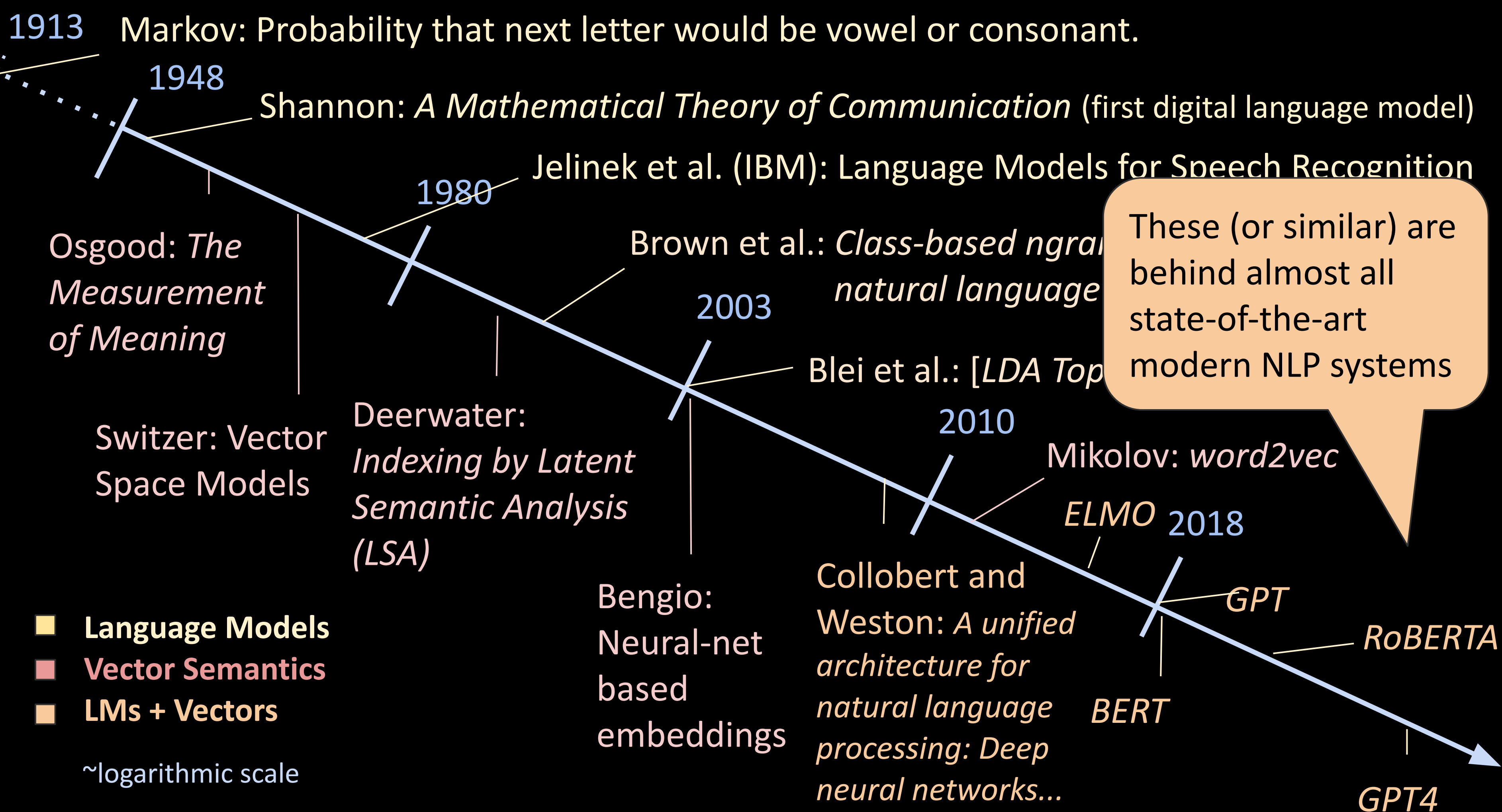
1913 Markov: Probability that next letter would be vowel or consonant.



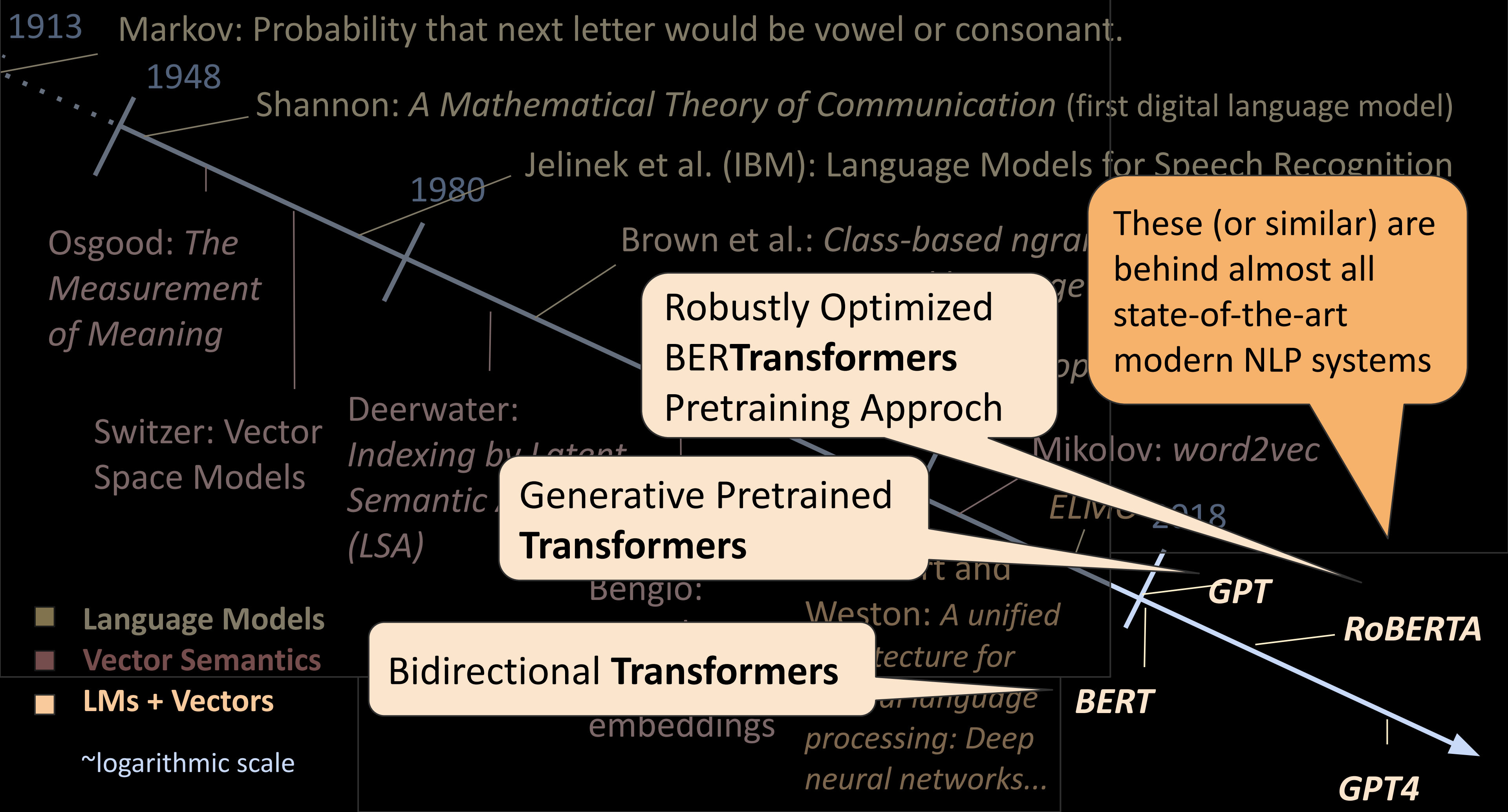
Timeline: *Language Modeling* and *Vector Semantics*



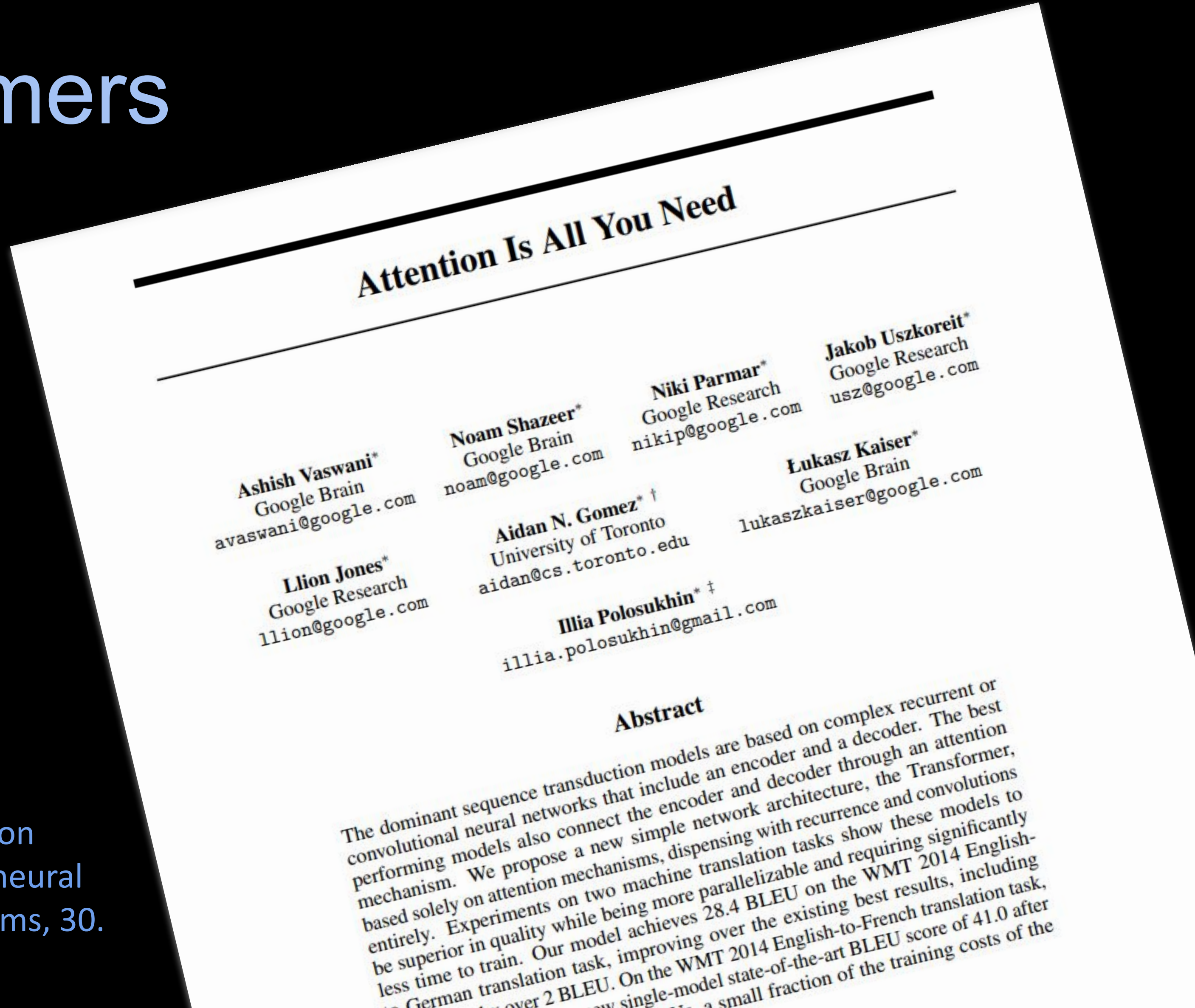
Timeline: *Language Modeling* and *Vector Semantics*



Timeline: *Language Modeling* and *Vector Semantics*



Transformers


Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

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

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-German translation task, improving over the existing best results, including ... over 2 BLEU. On the WMT 2014 English-to-French translation task, ... single-model state-of-the-art BLEU score of 41.0 after ... a small fraction of the training costs of the

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Transformers

Self-Attention

Deep Learning

Neural Networks

Transformers

- multi-headed attention
- positional embeddings
- residual links

(to be introduced later)

Self-Attention

Deep Learning

Neural Networks

Part 1: Deep Learning and Masked Language Models

Adithya V Ganesan

CSE538 - Spring 2024
bit.ly/cse538-sp24-lecture7

Artificial Neural Networks

What is it?

Artificial Neural Networks

What is it?

- Biologically inspired computing model
- Learn patterns from the data
- Can even approximate nonlinear functions in the nature!

Artificial Neural Networks

What is it?

- Biologically inspired **computing model**
- Learn patterns from the data
- Can even approximate nonlinear functions in the nature!

Artificial Neural Networks

How did we do this?

What is it?

- Biologically inspired **computing model**
- Learn patterns from the data
- Can even approximate nonlinear functions in the nature!

But, how do we model complex systems using these linear systems?

Deep Learning

But, how do we model complex systems using these linear systems?

Deep Learning

Non-linear functions + Artificial Neural Networks

Activation Functions

$$z = h_{(t)}W$$

Common Activation Functions

$z = h_{(t)}W$

Logistic: $\sigma(z) = 1 / (1 + e^{-z})$

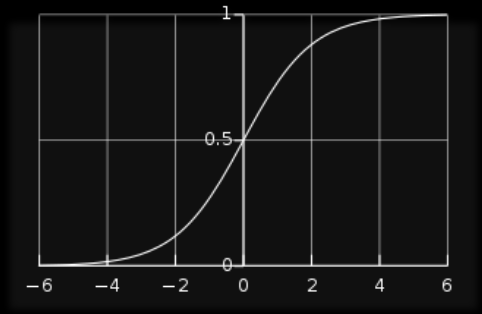
Hyperbolic tangent: $tanh(z) = 2\sigma(2z) - 1 = (e^{2z} - 1) / (e^{2z} + 1)$

Rectified linear unit (ReLU): $ReLU(z) = \max(0, z)$

Common Activation Functions

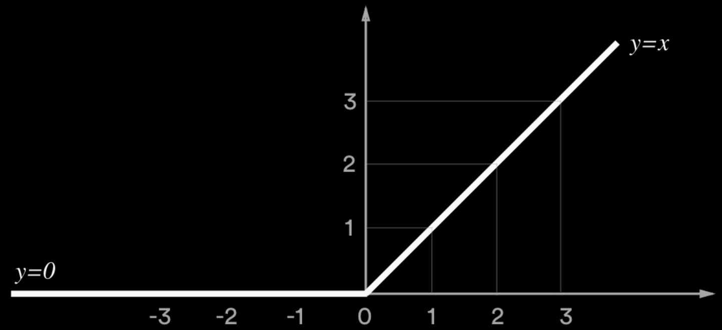
$z = h_{(t)}W$

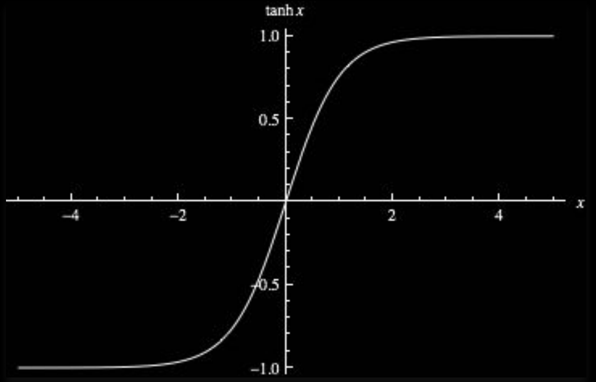
Logistic: $\sigma(z) = 1 / (1 + e^{-z})$



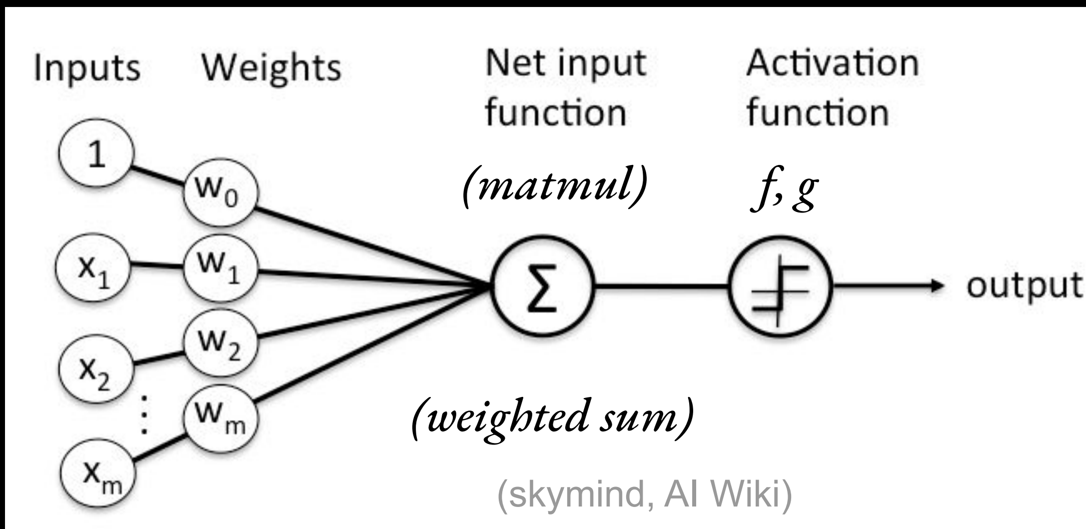
Hyperbolic tangent: $tanh(z) = 2\sigma(2z) - 1 = (e^{2z} - 1) / (e^{2z} + 1)$

Rectified linear unit (ReLU): $ReLU(z) = \max(0, z)$





Neural Networks: Graphs of Operations (excluding the optimization nodes)



Activation Function

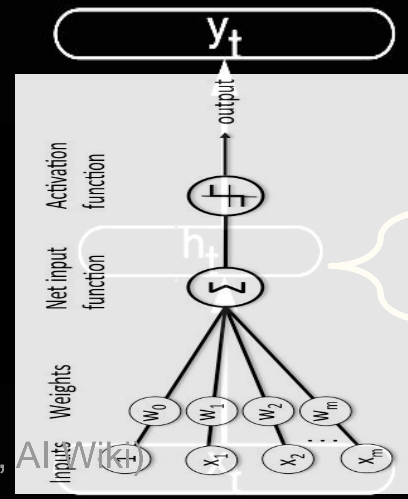
$$h_{(t)} = g(x_{(t)}W)$$

Figure 9.2 Simple recurrent neural network after Elman (Elman, 1990). The hidden layer includes a recurrent connection as part of its input. That is, the activation value of the hidden layer depends on the current input as well as the activation value of the hidden layer from the previous timestep.

(Jurafsky, 2019)

Neural Networks: Graphs of Operations (excluding the optimization nodes)

“hidden layer”



(skymind, AI Wiki)

Activation Function

$$h_{(t)} = g(x_{(t)}W)$$

Figure 9.2 Simple recurrent neural network after Elman (Elman, 1990). The hidden layer includes a recurrent connection as part of its input. That is, the activation value of the hidden layer depends on the current input as well as the activation value of the hidden layer from the previous timestep.

(Jurafsky, 2019)

Neural Networks: Graphs of Operations (excluding the optimization nodes)

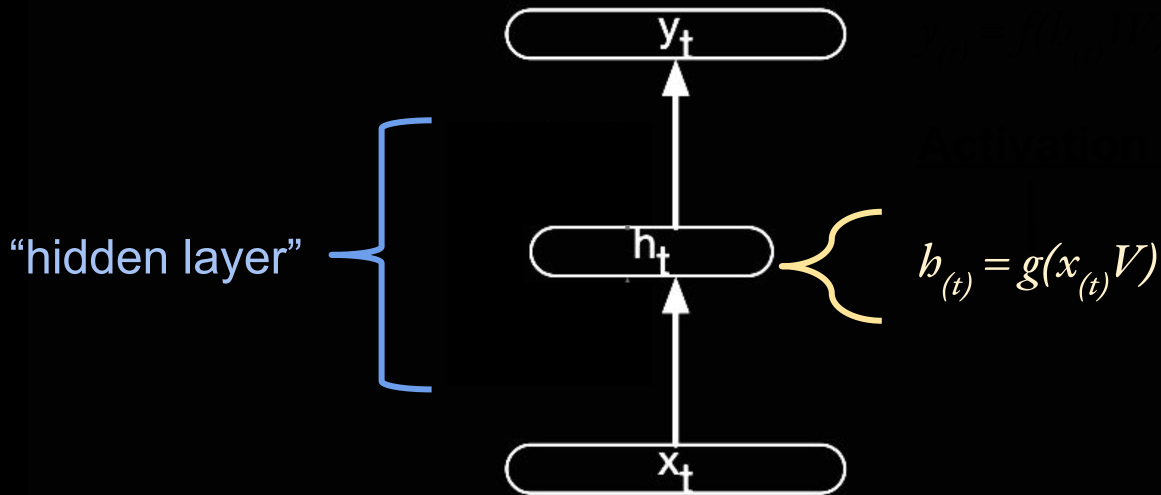
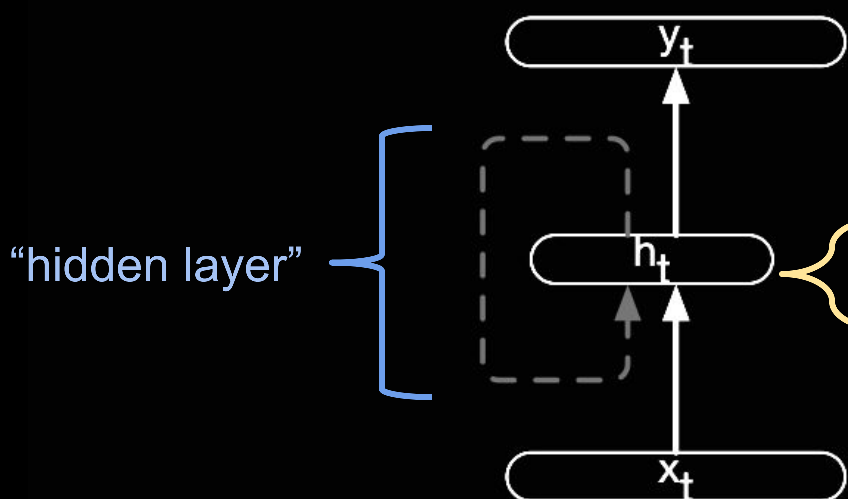


Figure 9.2 Simple recurrent neural network after Elman (Elman, 1990). The hidden layer includes a recurrent connection as part of its input. That is, the activation value of the hidden layer depends on the current input as well as the activation value of the hidden layer from the previous timestep.

(Jurafsky, 2019)

Neural Networks: Graphs of Operations (excluding the optimization nodes)



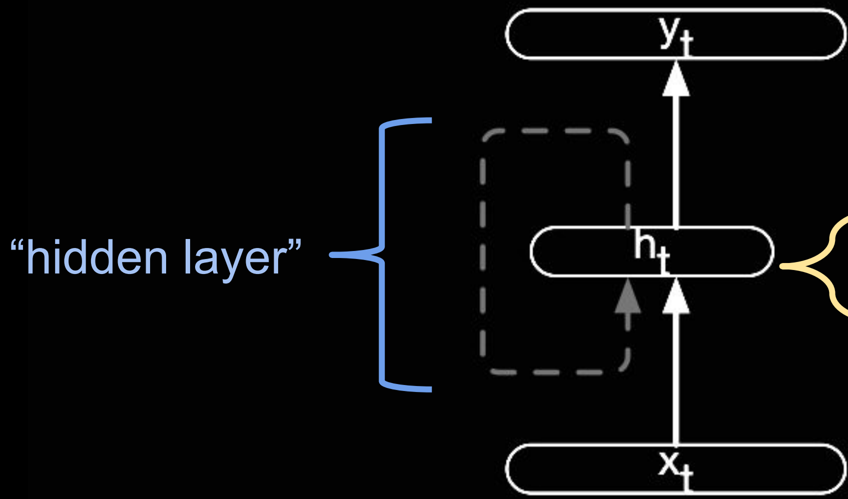
$$h_t = g(h_{t-1}U + x_tV)$$

short hand for vector/ matrix multiply

Figure 9.2 Simple recurrent neural network after Elman (Elman, 1990). The hidden layer includes a recurrent connection as part of its input. That is, the activation value of the hidden layer depends on the current input as well as the activation value of the hidden layer from the previous timestep.

(Jurafsky, 2019)

Neural Networks: Graphs of Operations (excluding the optimization nodes)



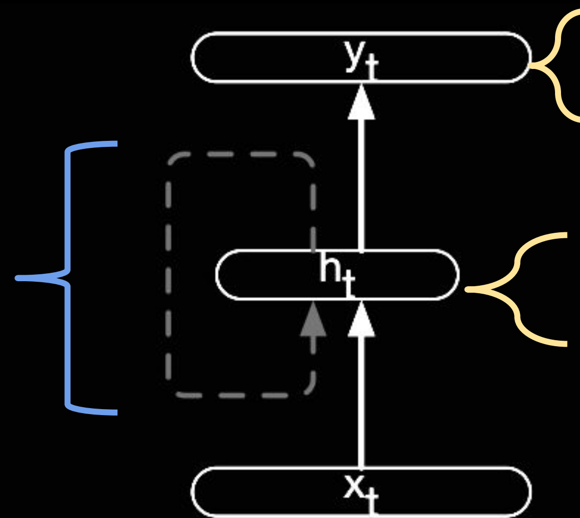
$$h_{(t)} = g(h_{(t-1)}U + x_{(t)}V)$$

Figure 9.2 Simple recurrent neural network after Elman (Elman, 1990). The hidden layer includes a recurrent connection as part of its input. That is, the activation value of the hidden layer depends on the current input as well as the activation value of the hidden layer from the previous timestep.

(Jurafsky, 2019)

Neural Networks: Graphs of Operations (excluding the optimization nodes)

“hidden layer”



$$y_{(t)} = f(h_{(t)}W)$$

Activation Function

$$h_{(t)} = g(h_{(t-1)}U + x_{(t)}V)$$

Figure 9.2 Simple recurrent neural network after Elman (Elman, 1990). The hidden layer includes a recurrent connection as part of its input. That is, the activation value of the hidden layer depends on the current input as well as the activation value of the hidden layer from the previous timestep.

(Jurafsky, 2019)

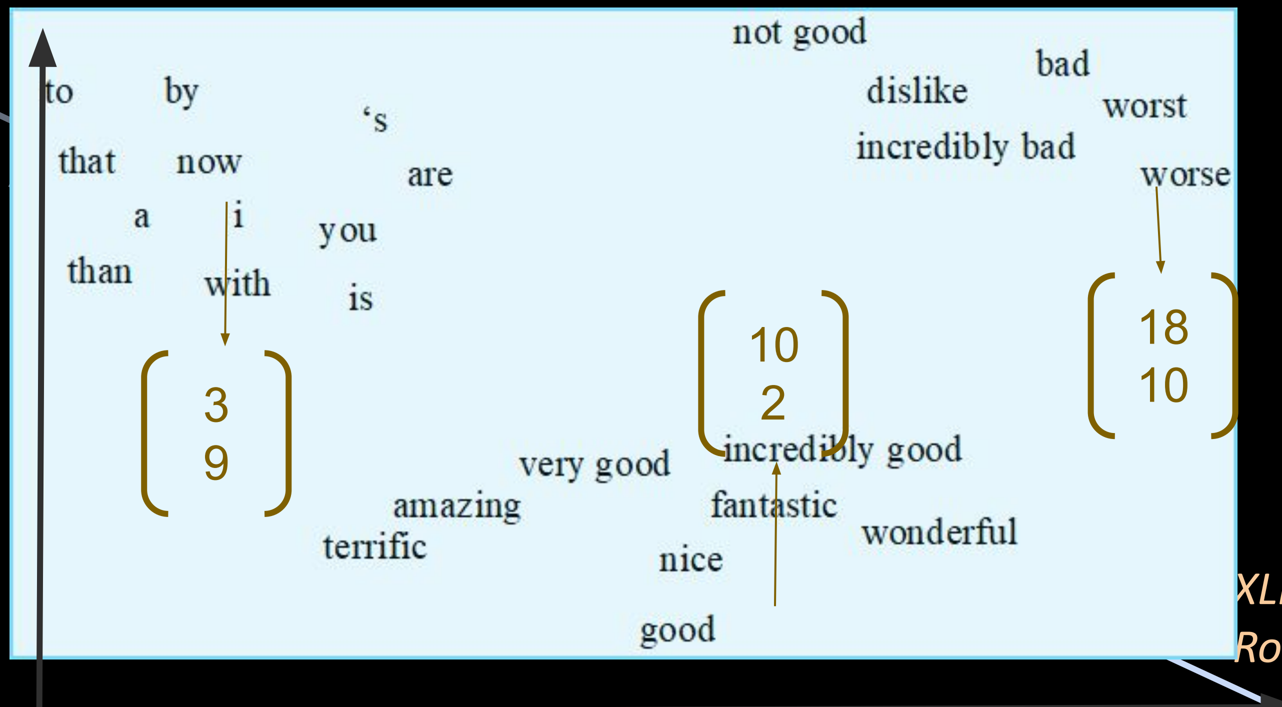
Back Propagation

Timeline: *Language Modeling* and *Vector Semantics*

1913 Markov: Probability that next letter would be vowel or consonant.

1948 Shannon: *A Mathematical Theory of Communication* (first digital language model)

Osgood: *The Measurement of Meaning*



XLNet

RoBERTA

GPT4

- Language Models
- Vector Semantics
- LMs + Vectors

~logarithmic scale

(Li et al. ,2015; Jurafsky et al., 2019)

Word Vectors

To embed: convert a token (or sequence) to a vector that represents **meaning.**

Wittgenstein, 1945: "*The meaning of a word is its use in the language*"

Distributional hypothesis -- A word's meaning is defined by all the different contexts it appears in (i.e. how it is "distributed" in natural language).

Firth, 1957: "*You shall know a word by the company it keeps*"

The nail hit the beam behind the wall.

Word Vectors

	Person A	**Person B**
How are you?	I feel ***fine*** –even ***great***!	My life is a ***great*** mess! I'm having a very hard time being happy.
What is going on?	Earlier, I ***played*** the ***game*** Yahtzee with my ***partner*****.** I could not get that ***die*** to roll a 1! Now I'm ***lying*** on my bed for a ***rest***.	My business ***partner*** was ***lying*** to me. He was trying to ***game*** the system and ***played*** me. I think I am going to ***die*** –he left and now I have to pay the ***rest*** of his ***fine*****.**

(Kjell, Kjell, and Schwartz, 2023)

Objective

$$\textit{great} \xrightarrow{\text{embed}} \begin{pmatrix} 0.53 \\ 1.5 \\ 3.21 \\ -2.3 \\ .76 \end{pmatrix}$$

Objective

great $\xrightarrow{\text{embed}}$ $\begin{bmatrix} 0.53 \\ 1.5 \\ 3.21 \\ -2.3 \\ .76 \end{bmatrix}$?

great.a.1 (relatively large in size or number or extent; larger than others of its kind)

great.a.2, outstanding (of major significance or importance)

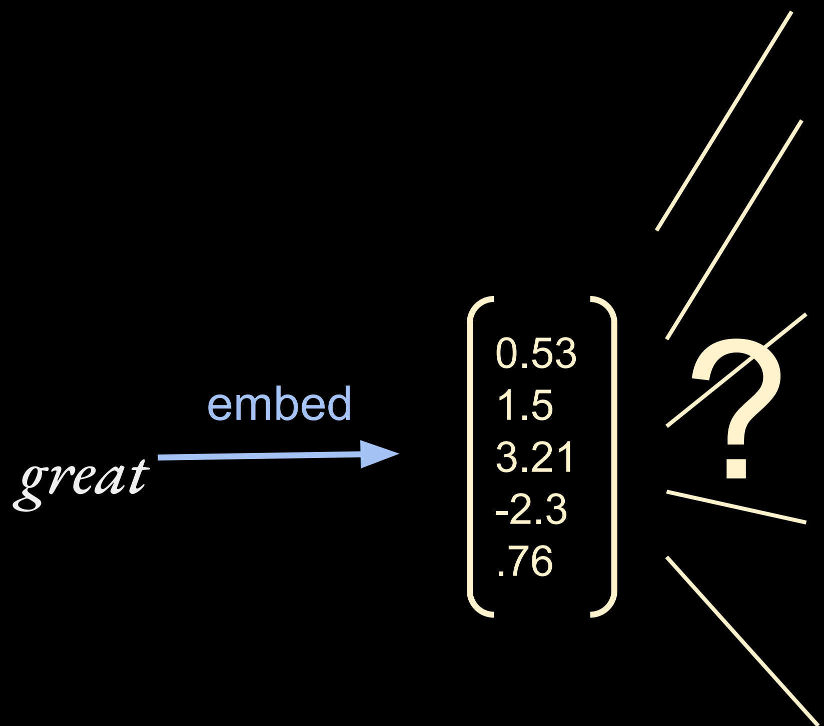
great.a.3 (remarkable or out of the ordinary in degree or magnitude or effect)

bang-up, bully, corking, cracking, dandy, **great.a.4**, groovy, keen, neat, nifty, not bad, peachy, slap-up, swell, smashing, old (very good)

capital, **great.a.5**, majuscule (uppercase)

big, enceinte, expectant, gravid, **great.a.6**, large, heavy, with child (in an advanced stage of pregnancy)

Objective



great.a.1 (relatively large in size or number or extent; larger than others of its kind)

great.a.2, outstanding (of major significance or importance)

great.a.3 (remarkable or out of the ordinary in degree or magnitude or effect)

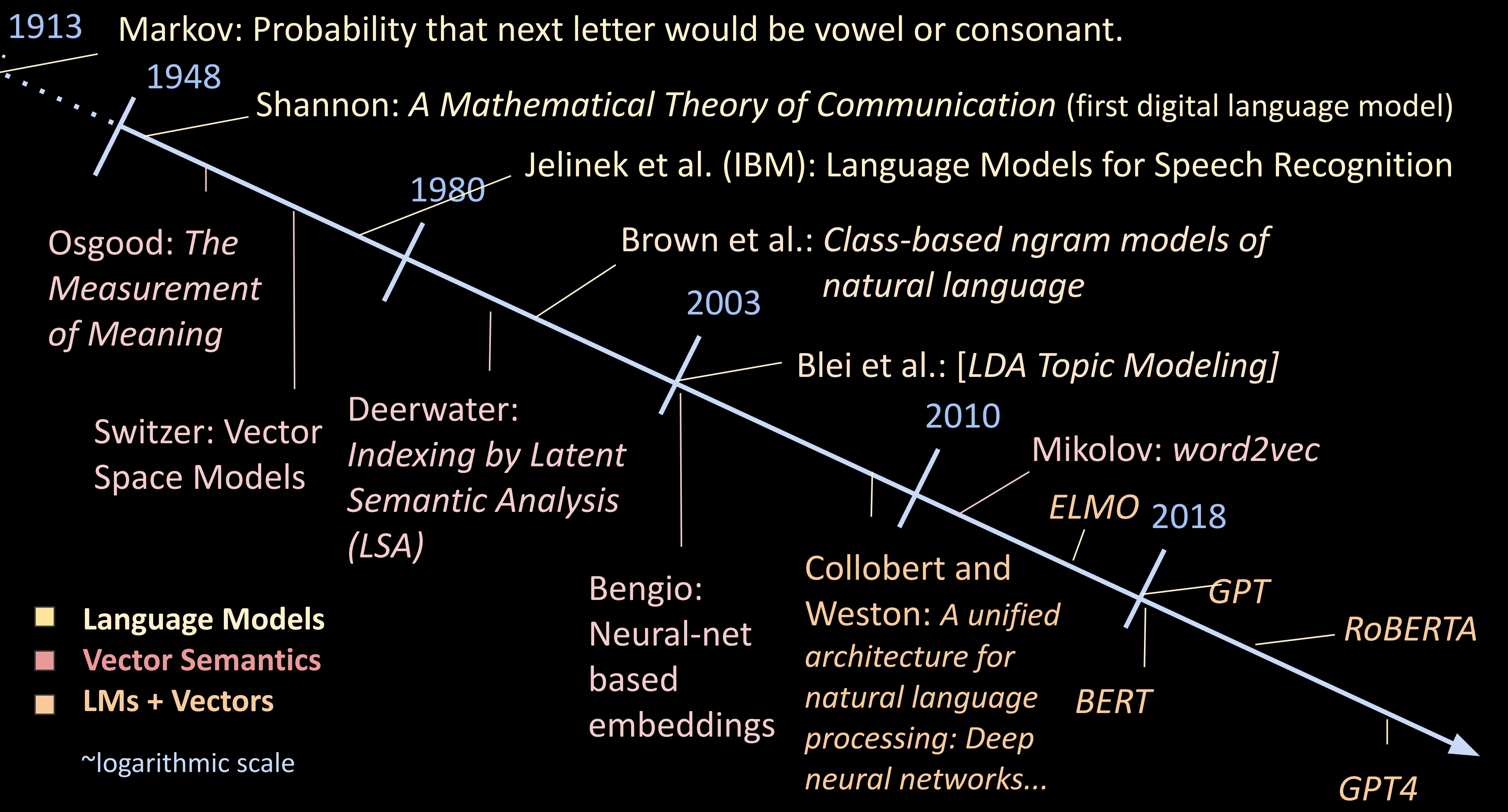
bang-up, bully, corking, cracking, dandy, **great.a.4**, groovy, keen, neat, nifty, not bad, peachy, slap-up, swell, smashing, old (very good)

capital, **great.a.5**, majuscule (uppercase)

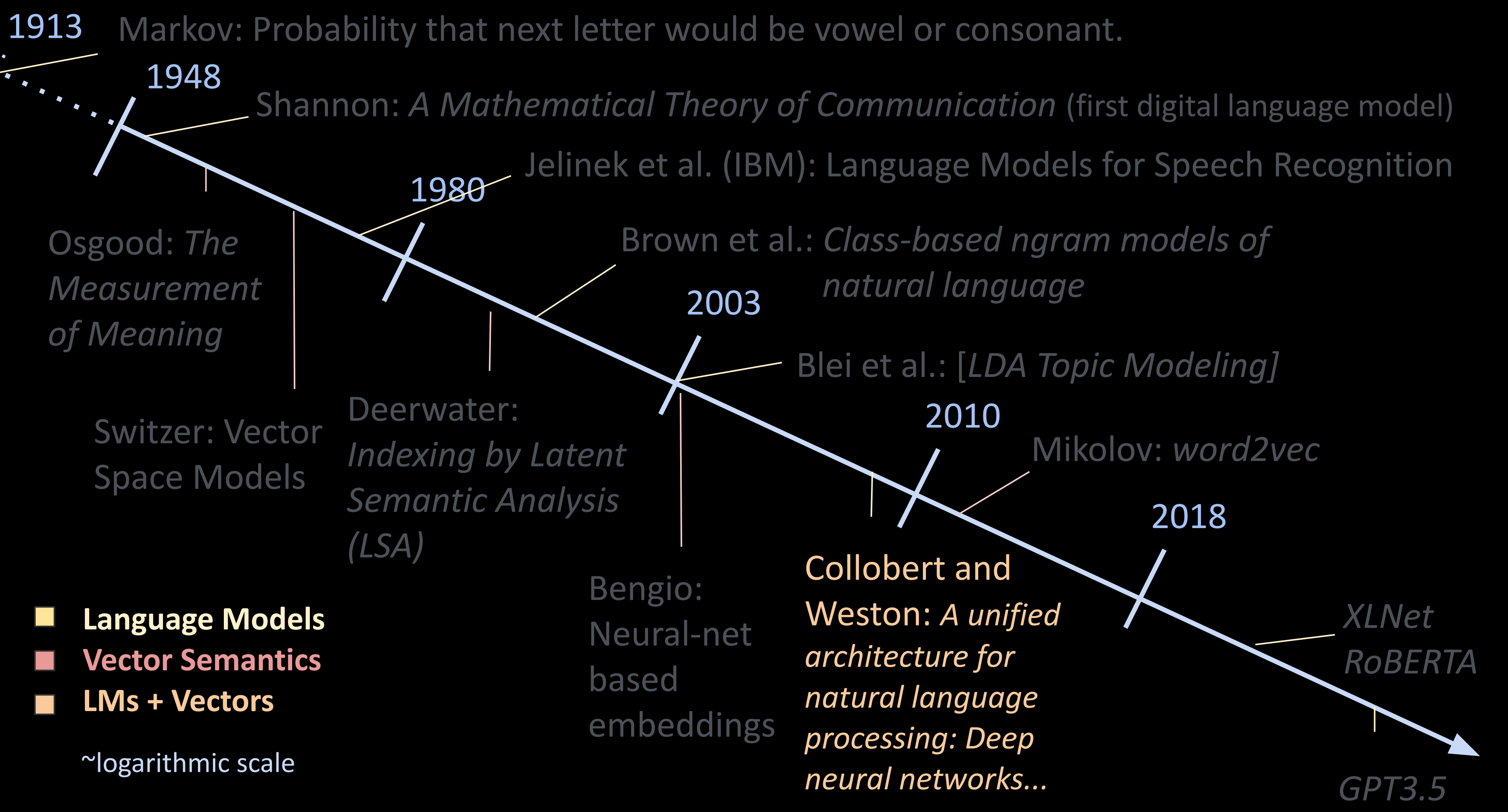
big, enceinte, expectant, gravid, **great.a.6**, large, heavy, with child (in an advanced stage of pregnancy)

great.n.1 (a person who has achieved distinction and honor in some field)

Timeline: *Language Modeling* and *Vector Semantics*



Timeline: *Language Modeling* and *Vector Semantics*



T… odeling and *Vector Semantics*

1913 … r would be vowel or consonant.

… *l Theory of Communication* (first digital language model)

… et al. (IBM): Language Models for Speech Recognition

… rown et al.: *Class-based ngram models of natural language*

Os… *M… of…*

2003

Blei et al.: [*LDA Topic Modeling*]

… gio: … ural-net … ed … beddings

2010

Mikolov: *word2vec*

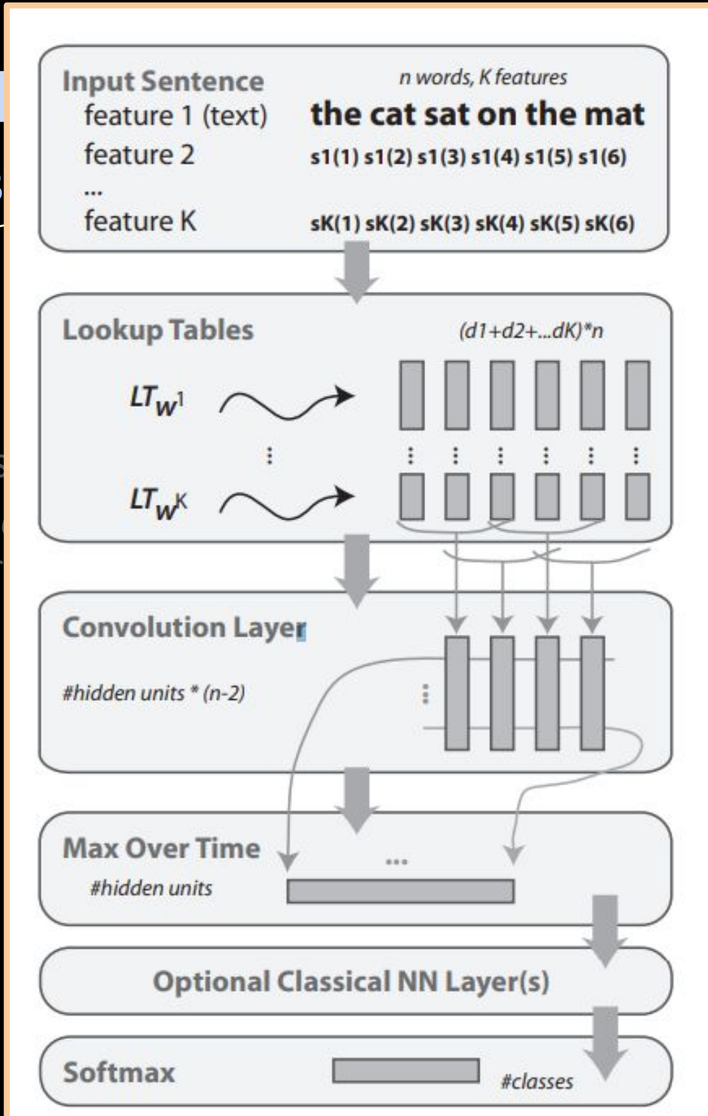
Collobert and Weston: *A unified architecture for natural language processing: Deep neural networks...*

2018

XLNet
RoBERTA

GPT3.5

… logarithmic scale



T… odeling and *Vector Semantics*

1913

…r would be vowel or consonant.

…*l Theory of Communication* (first digital language model)

…et al. (IBM): Language Models for Speech Recognition

…rown et al.: *Class-based ngram models of natural language*

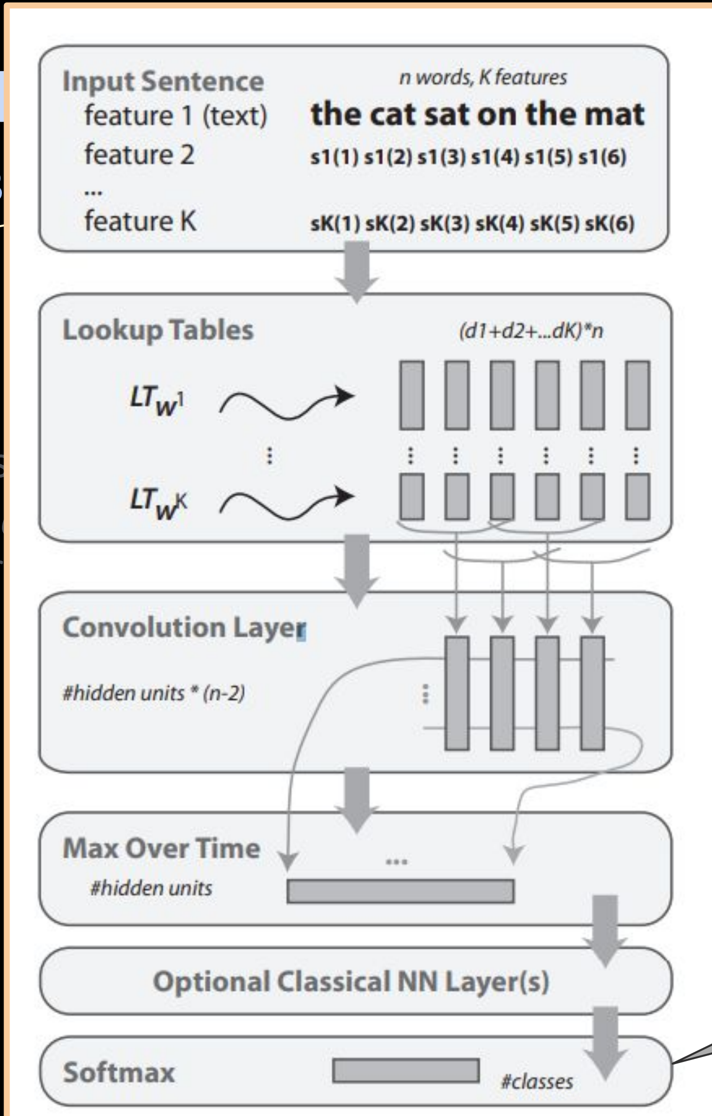
Os… *M… of…*

2003

Blei et al.: [*LDA Topic Modeling*]

…lov: *word2vec*

2018



POS, Chunking (Shallow Parsing), NER, SRL, Modified Language Modelling

…architecture for natural language processing: Deep neural networks…

…g… …ur… …ed… …beddings

XLNet
RoBERTA

GPT3.5

logarithmic scale

odeling and *Vector Semantics*

1913

r would be vowel or consonant.

l Theory of Communication (first digital language model)

et al. (IBM): Language Models for Speech Recognition

rown et al.: *Class-based ngram models of natural language*

2003

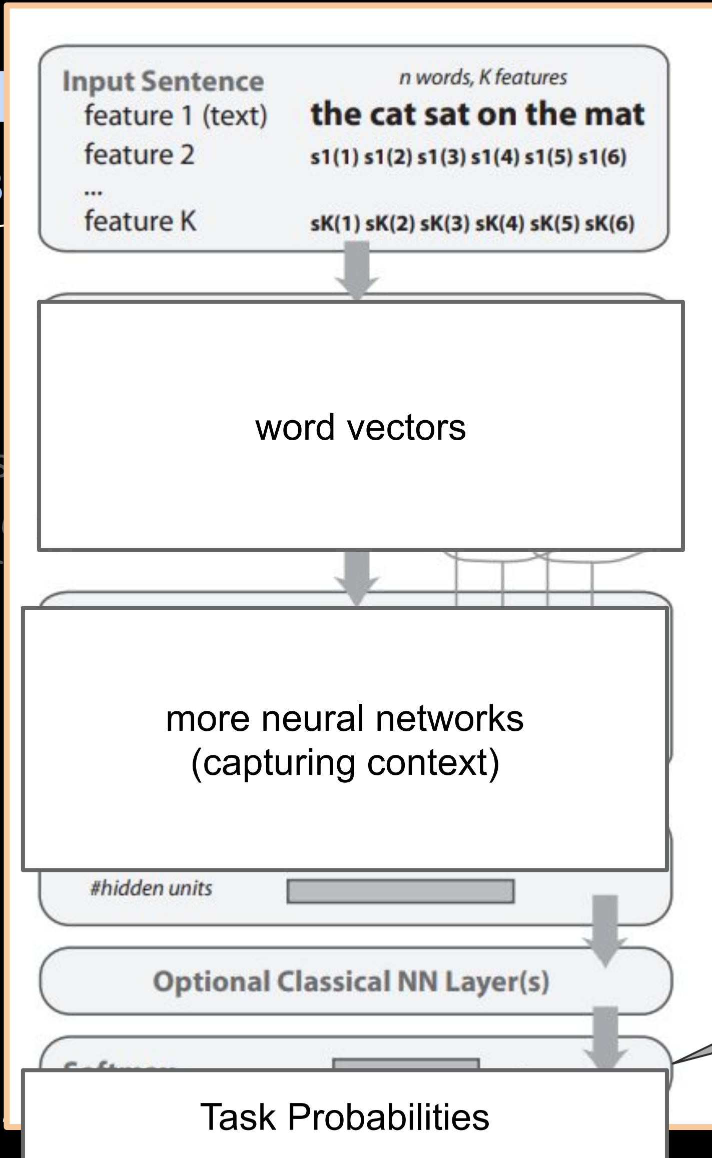
Blei et al.: [*LDA Topic Modeling*]

lov: *word2vec*

2018

XLNet
RoBERTA

GPT3.5



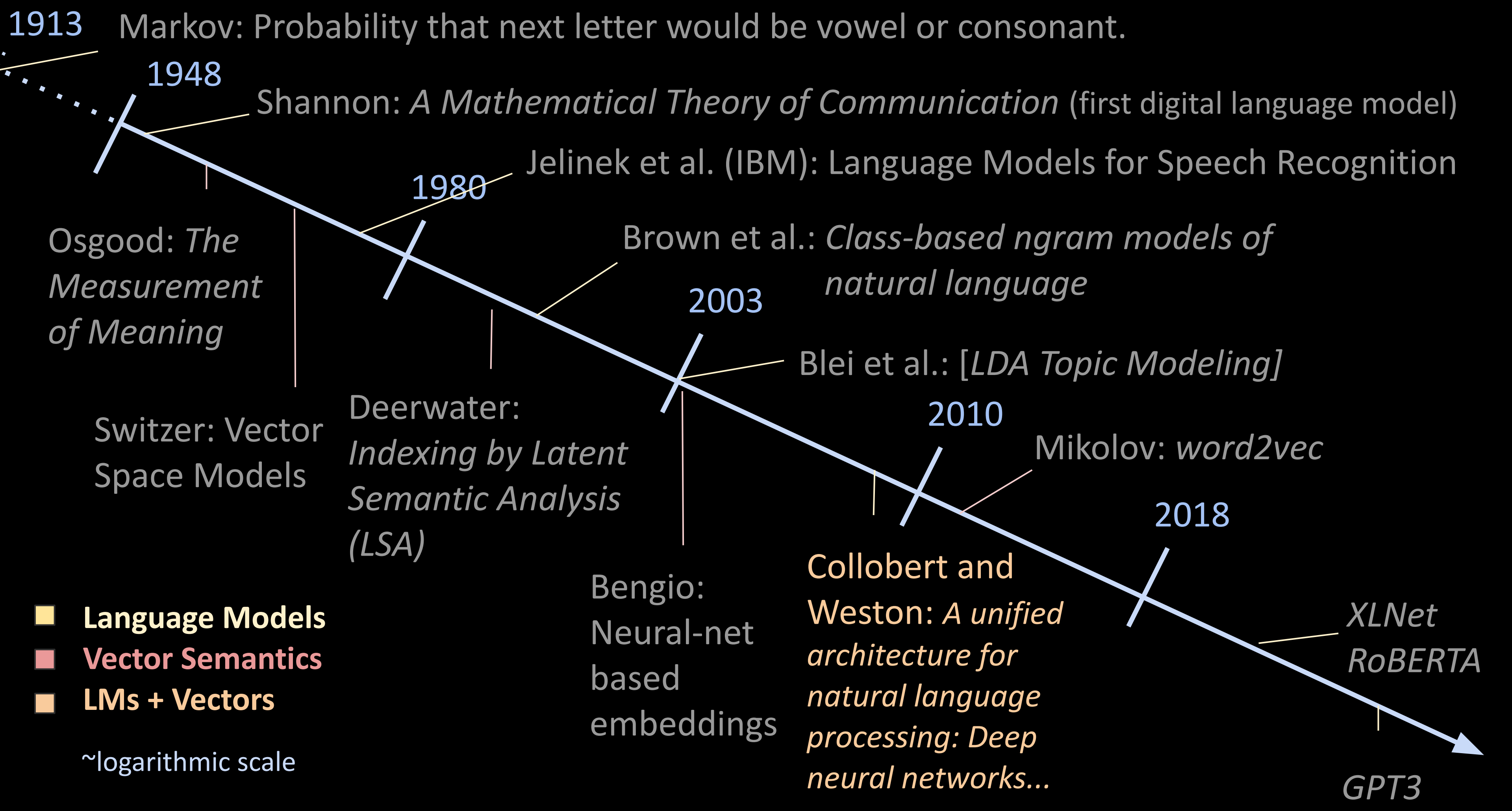
POS, Chunking (Shallow Parsing), NER, SRL, Modified Language Modelling

architecture for natural language processing: Deep neural networks...

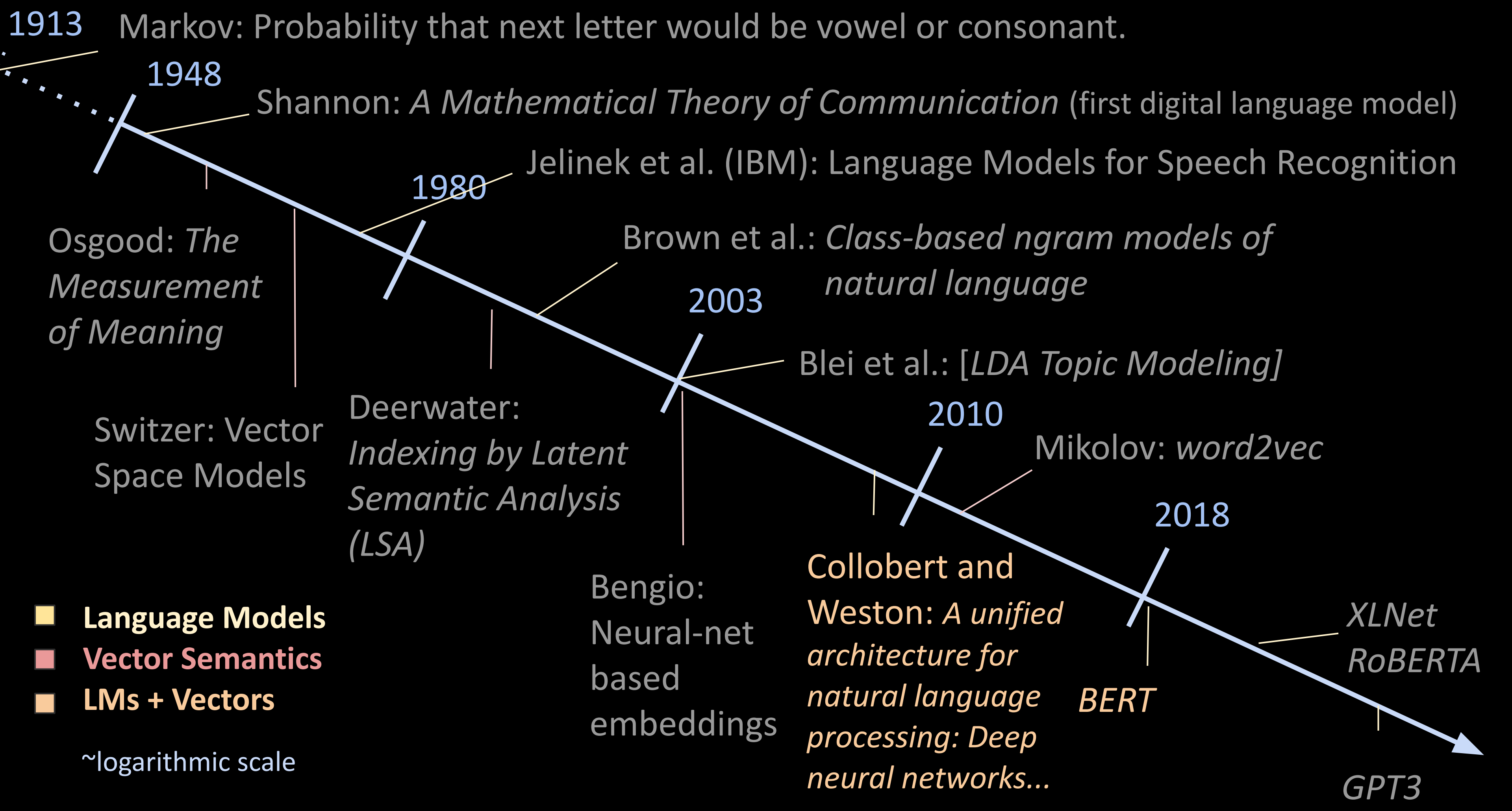
ed

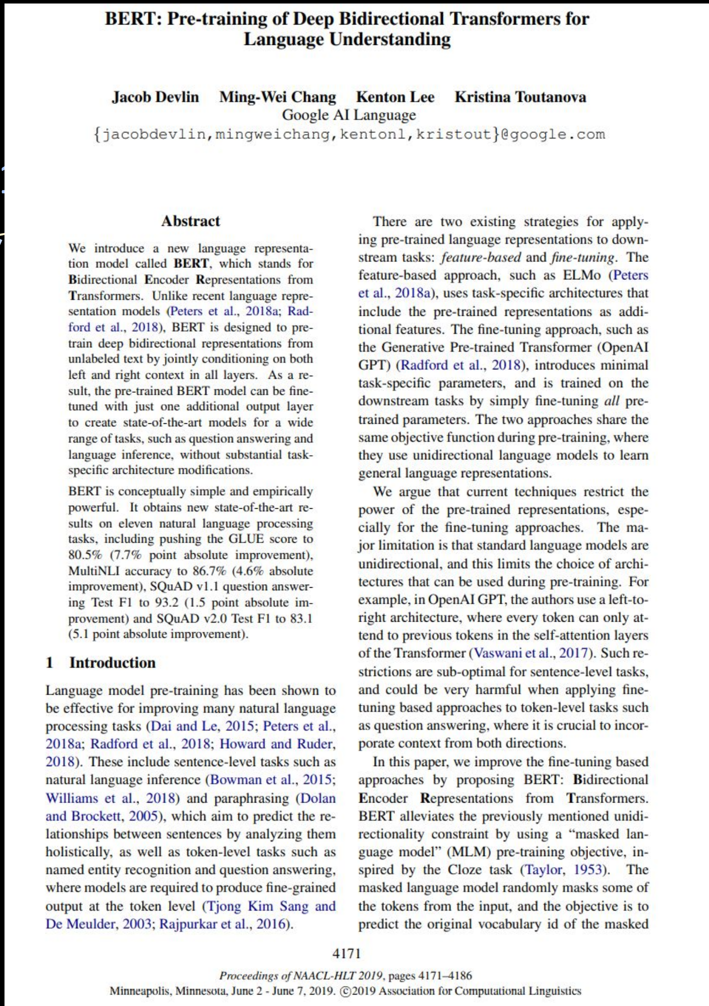
beddings

Timeline: *Language Modeling* and *Vector Semantics*



Timeline: *Language Modeling* and *Vector Semantics*





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 pretrain 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).

1 Introduction

Language model pre-training has been shown to be effective for improving many natural language processing tasks (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan and Brockett, 2005), which aim to predict the relationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering, where models are required to produce fine-grained output at the token level (Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016).

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 of the Transformer (Vaswani et al., 2017). Such restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks such as question answering, where it is crucial to incorporate context from both directions.

In this paper, we improve the fine-tuning based approaches by proposing BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. BERT alleviates the previously mentioned unidirectionality constraint by using a "masked language model" (MLM) pre-training objective, inspired by the Cloze task (Taylor, 1953). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked

Proceedings of NAACL-HLT 2019, pages 4171–4186
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

e Modeling and *Vector Semantics*

letter would be vowel or consonant.

atical Theory of Communication (first digital language model)

inek et al. (IBM): Language Models for Speech Recognition

Brown et al.: *Class-based ngram models of natural language*

2003

Blei et al.: [*LDA Topic Modeling*]

2010

Mikolov: *word2vec*

tent

ysis

2018

Bengio: Neural-net based embeddings

Collobert and Weston: *A unified architecture for natural language processing: Deep neural networks...*

BERT

XLNet
RoBERTA

GPT3

LMs + Vectors

~logarithmic scale

BERT: Pre-training of Deep Bidirectional Transformers for

BERT Rediscovers the Classical NLP Pipeline

Ian Tenney[1] **Dipanjan Das**[1] **Ellie Pavlick**[1,2]
[1]Google Research [2]Brown University
{iftenney,dipanjand,epavlick}@google.com

Abstract

Pre-trained text encoders have rapidly advanced the state of the art on many NLP tasks. We focus on one such model, BERT, and aim to quantify where linguistic information is captured within the network. We find that the model represents the steps of the traditional NLP pipeline in an interpretable and localizable way, and that the regions responsible for each step appear in the expected sequence: POS tagging, parsing, NER, semantic roles, then coreference. Qualitative analysis reveals that the model can and often does adjust this pipeline dynamically, revising lower-level decisions on the basis of disambiguating information from higher-level representations.

of the network directly, to assess whether there exist localizable regions associated with distinct types of linguistic decisions. Such work has produced evidence that deep language models can encode a range of syntactic and semantic information (e.g. Shi et al., 2016; Belinkov, 2018; Tenney et al., 2019), and that more complex structures are represented hierarchically in the higher layers of the model (Peters et al., 2018b; Blevins et al., 2018).

We build on this latter line of work, focusing on the BERT model (Devlin et al., 2019), and use a suite of probing tasks (Tenney et al., 2019) derived from the traditional NLP pipeline to quantify where specific types of linguistic information are

lationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering, where models are required to produce fine-grained output at the token level (Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016).

rectionality constraint by using a "masked language model" (MLM) pre-training objective, inspired by the Cloze task (Taylor, 1953). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked

4171

Proceedings of NAACL-HLT 2019, pages 4171–4186
Minneapolis, Minnesota, June 2 - June 7, 2019. ©2019 Association for Computational Linguistics

and *Vector Semantics*

wel or consonant.

ommunication (first digital language model)

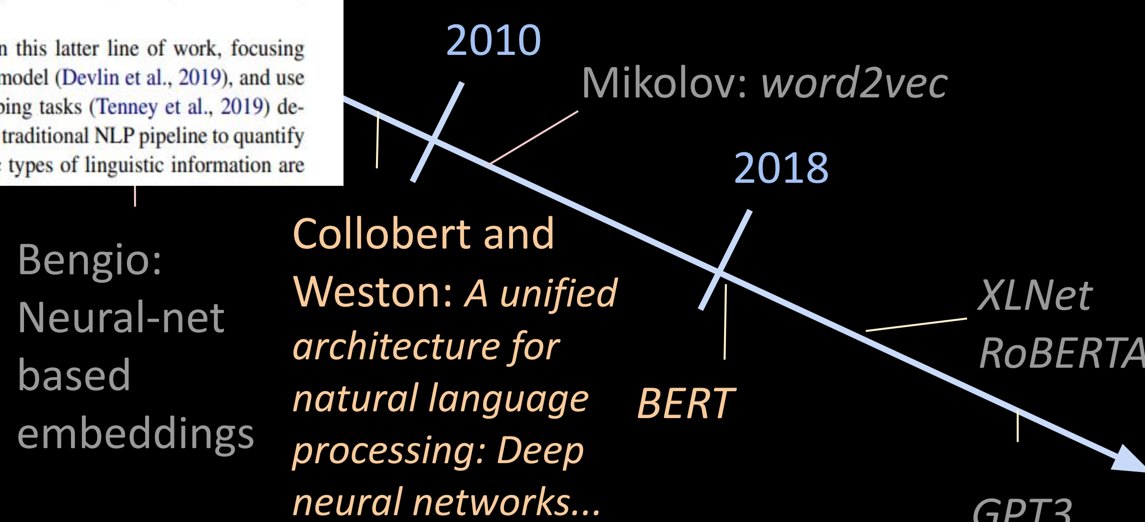
anguage Models for Speech Recognition

lass-based ngram models of atural language

i et al.: [*LDA Topic Modeling*]

- **LMs + Vectors**

~logarithmic scale



BERT: Pre-trair

Jacob Devlin Mi

{jacobdevlin,mi

Abstract

We introduce a new languag tion model called **BERT**, whi **B**idirectional **E**ncoder **R**eprese **T**ransformers. Unlike recent la sentation models (Peters et al. ford et al., 2018), BERT is de train deep bidirectional represe unlabeled text by jointly conditi left and right context in all lay sult, the pre-trained BERT mod tuned with just one additional to create state-of-the-art mode range of tasks, such as question language inference, without su specific architecture modificatio

BERT is conceptually simple a powerful. It obtains new state sults on eleven natural langua tasks, including pushing the G 80.5% (7.7% point absolute i MultiNLI accuracy to 86.7% (improvement), SQuAD v1.1 qu ing Test F1 to 93.2 (1.5 point provement) and SQuAD v2.0 T (5.1 point absolute improvemen

1 Introduction

Language model pre-training ha be effective for improving many processing tasks (Dai and Le, 20 2018a; Radford et al., 2018; Ho 2018). These include sentence-le natural language inference (Bow Williams et al., 2018) and para and Brockett, 2005), which aim lationships between sentences by holistically, as well as token-lev named entity recognition and qu where models are required to pro output at the token level (Tjong De Meulder, 2003; Rajpurkar et a

Minneapolis, Minnesota, J

Journalism Quarterly

DEVOTED TO RESEARCH STUDIES IN THE FIELD OF MASS COMMUNICATIONS

FALL 1953

"Cloze Procedure": A New Tool For Measuring Readability

BY WILSON L. TAYLOR*

Here is the first comprehensive statement of a research method and its theory which were introduced briefly during a workshop at the 1953 AEJ convention. Included are findings from three pilot studies and two experiments in which "cloze procedure" results are compared with those of two readability formulas.

"CLOZE PROCEDURE" IS A NEW PSYchological tool for measuring the effectiveness of communication. The method is straightforward; the data are easily quantifiable; the findings seem to stand up.

At the outset, this tool was looked on as a new approach to "readability." It was so used in three pilot studies and two experiments, the main findings of which are reported here.

First, the results of the new method were repeatedly shown to conform with the results of the Flesch and Dale-Chall devices for estimating readability. Then the scope broadened, and cloze procedure was pitted against those standard formulas.

If future research substantiates the results so far, this tool seems likely to both have a variety of applications, both theoretical and practical, in other fields involving communication functions.

THE "CLOZE UNIT"

At the heart of the procedure is a functional unit of measurement tentatively dubbed a "cloze." It is pronounced like the verb "close" and is derived from "closure." The last term is one gestalt psychology applies to the human tendency to complete a familiar but not-quite-finished pattern—to "see" a broken circle as a whole one, for example, by mentally closing up the gaps.

*The writer is particularly obligated to Prof. Charles E. Osgood, University of Illinois, and Melvin R. Marks, Personnel Research Section, A.G.O., Department of the Army, for instigating and assisting in the series of efforts that yielded the notion of "cloze procedure." Both are experimental psychologists. Among others who have advised, encouraged or otherwise aided are these of the University of Illinois: Prof. Lee J. Cronbach, educational psychologist and statistician; Dean Wilbur Schramm, Division of Communications; Prof. Charles E. Swanson, Institute of Communications Research, and George R. Klare, psychologist, both of whom have authored articles on readability; and several journalism teachers who lent their classes. Kalmer E. Stordahl and Clifford M. Christensen, until recently research associates of the Institute, also contributed.

ng and *Vector Semantics*

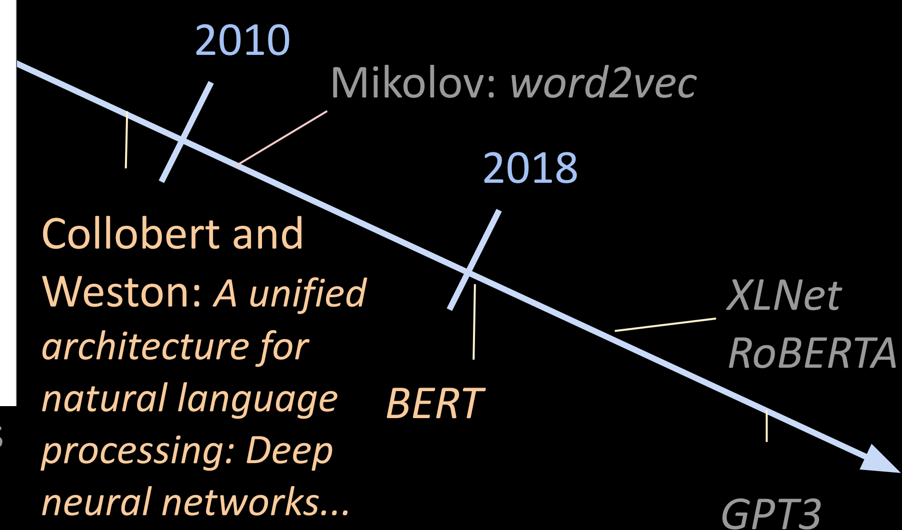
e vowel or consonant.

of Communication (first digital language model)

M): Language Models for Speech Recognition

l.: *Class-based ngram models of natural language*

Blei et al.: [*LDA Topic Modeling*]

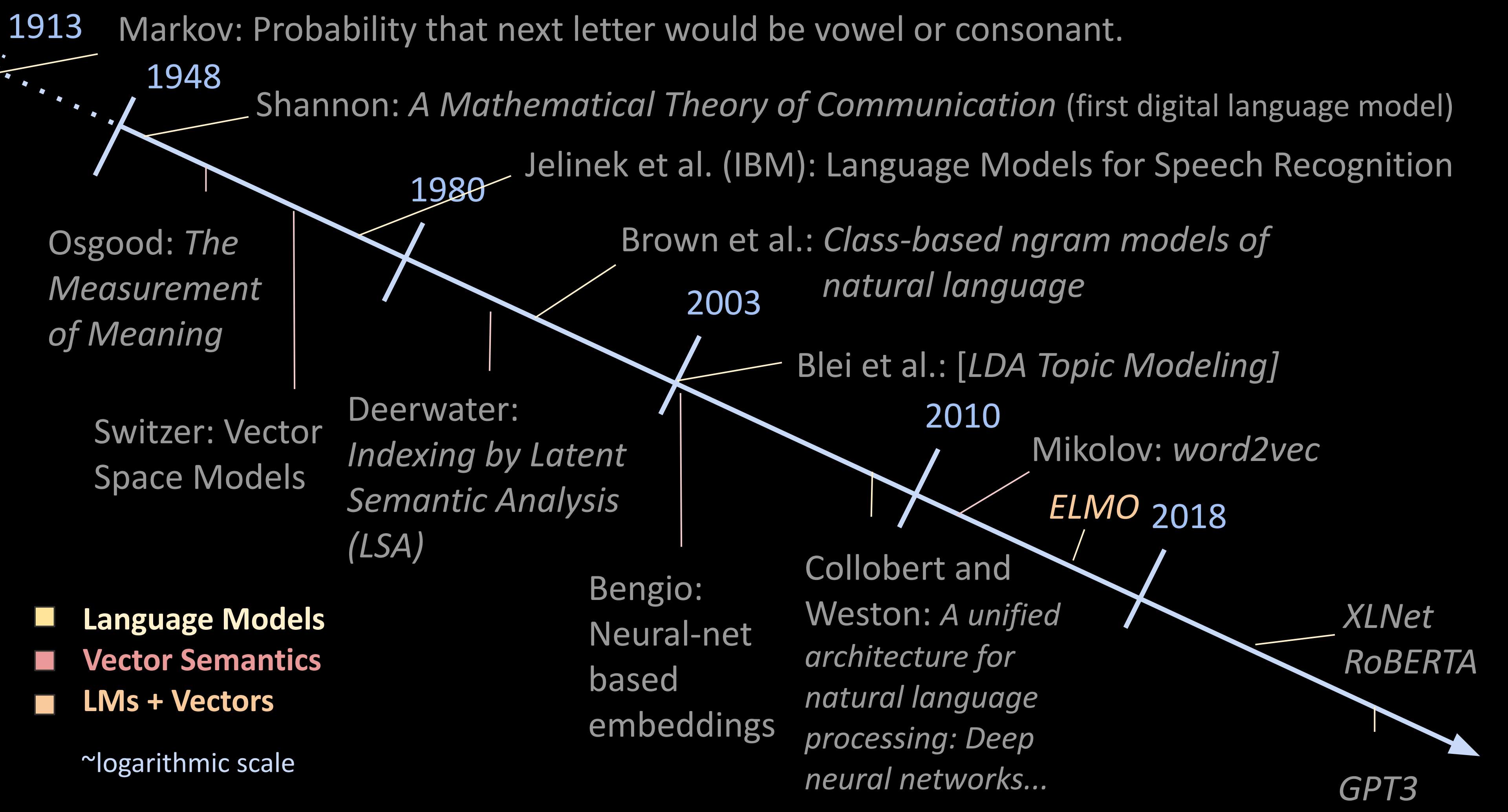


embeddings

LMs + Vectors

~logarithmic scale

Timeline: *Language Modeling* and *Vector Semantics*



Masked Language Modeling

Task: Estimate $P(w_i | w_1,\ldots w_{i-1}, w_{i+1}\ldots w_n)$

:P(masked word given history)

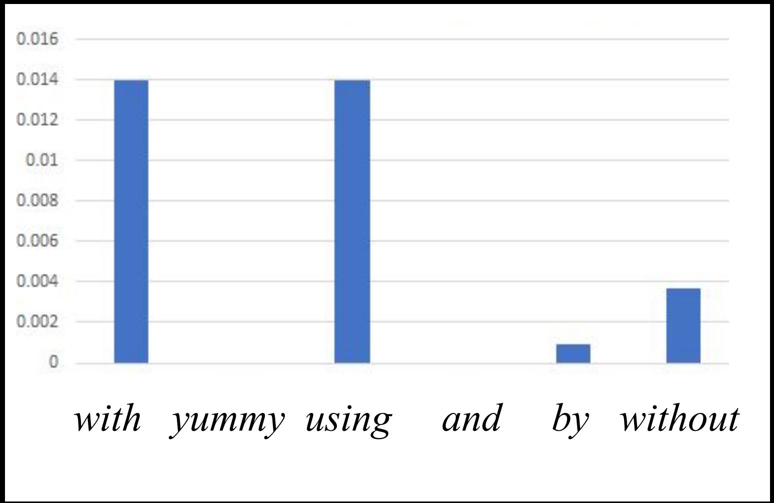
$P(with \mid He\ ate\ the\ cake <M> the\ fork) = ?$

Masked Language Modeling

Task: Estimate $P(w_i \mid w_1,\ldots w_{i-1}, w_{i+1}\ldots w_n)$

:P(masked word given history)

P(with | He ate the cake <M> the fork) = ?

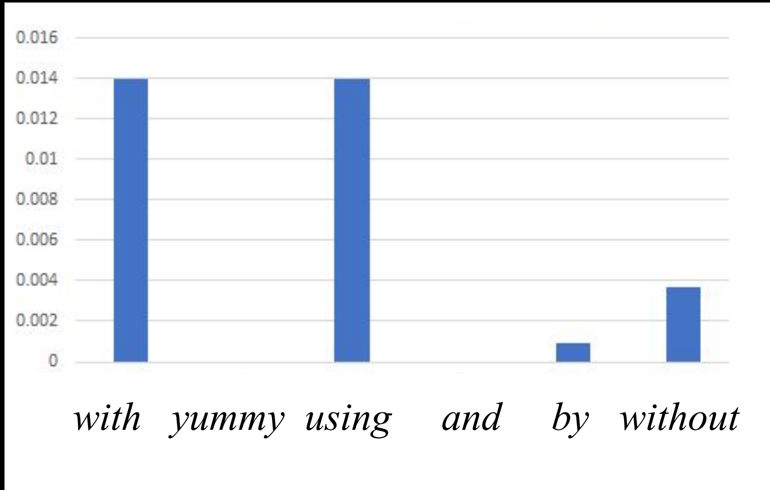


Masked Language Modeling

Task: Estimate $P(w_i \mid w_1, \ldots w_{i-1}, w_{i+1} \ldots w_n)$

:P(masked word given history)

P(with | He ate the cake <M> the fork) = ?



Masked Language Modeling

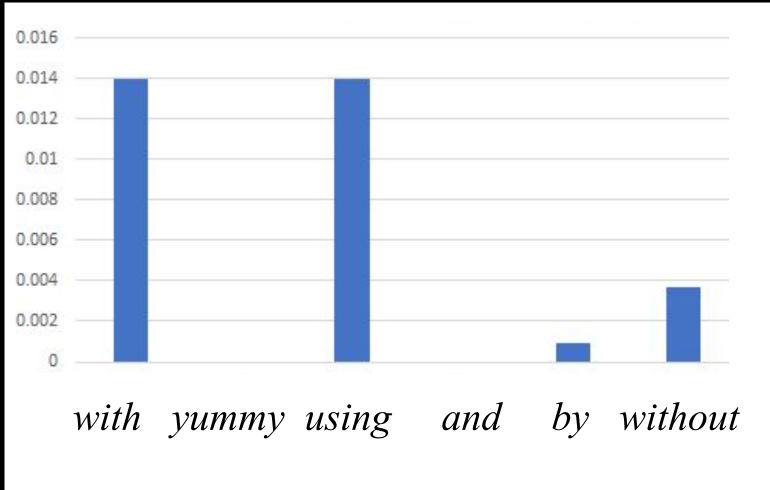
Task: Estimate $P(w_i | w_1,...w_{i-1}, w_{i+1}... w_n)$

:P(masked word given history)

P(with | He ate the cake <M> the fork) = ?

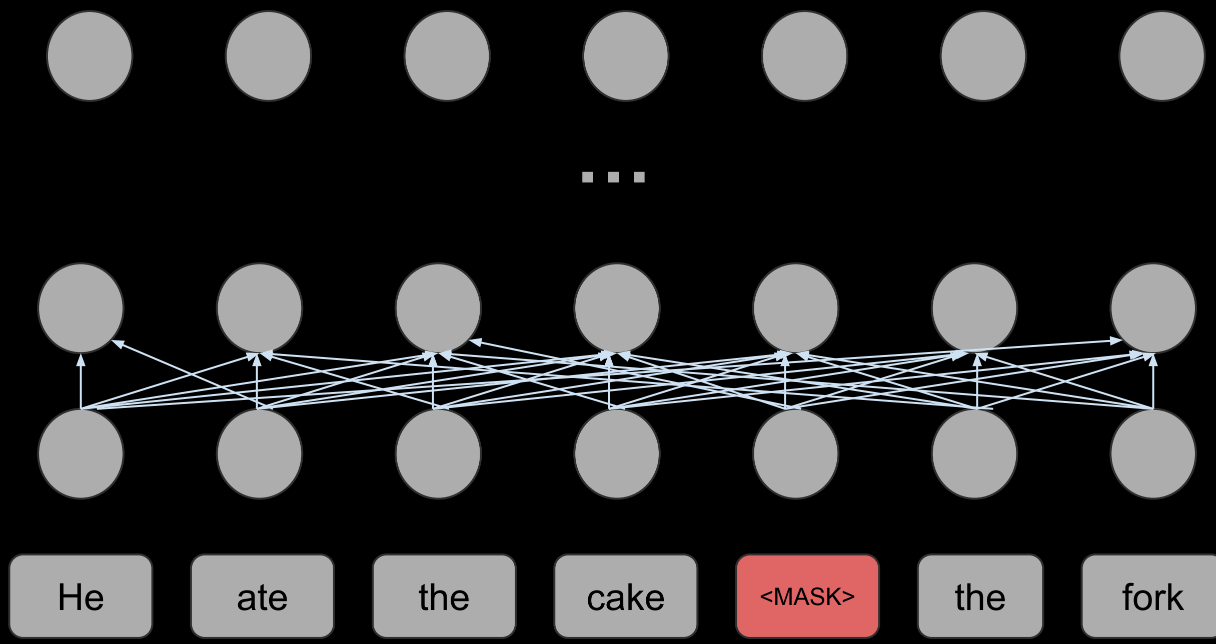
Sequence
(He, at, the, cake,<MASK>, the, fork)

→ Neural Network → What is the masked word in the sequence?

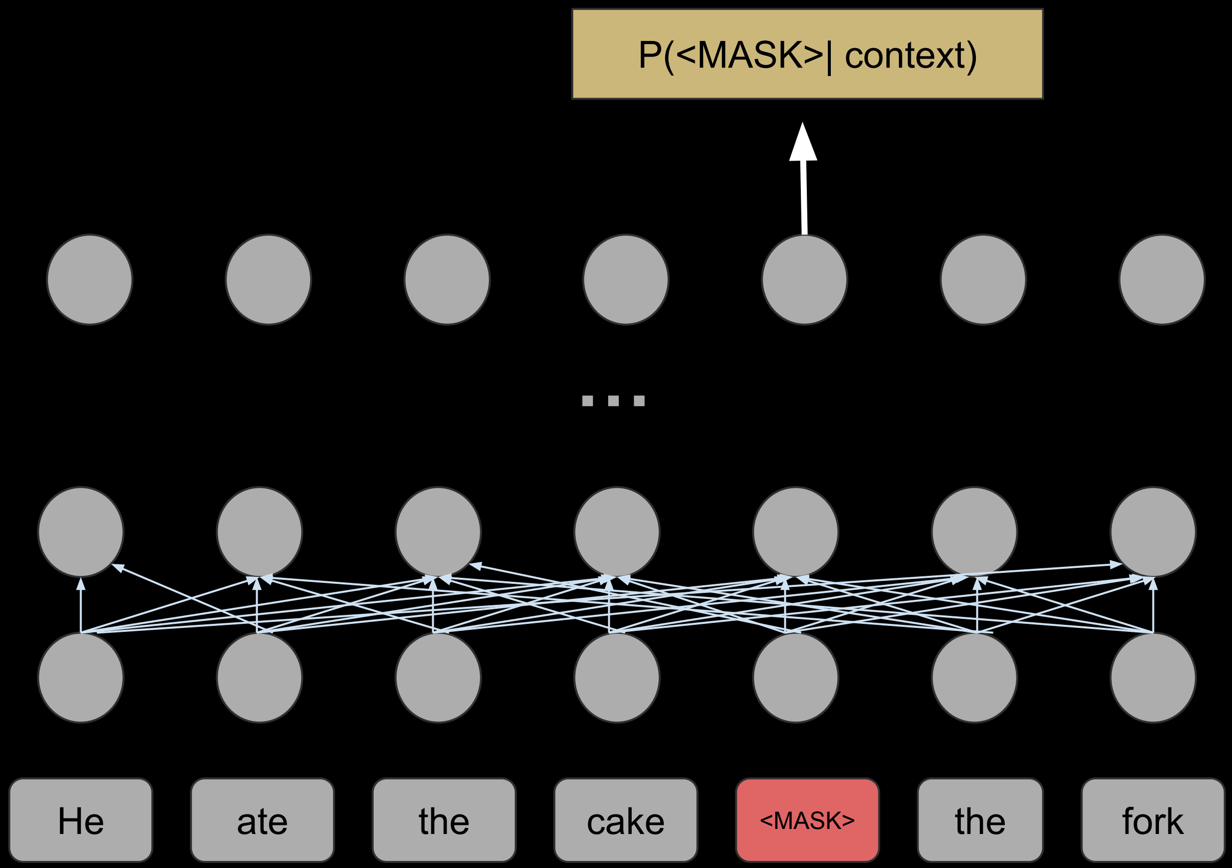


Masked Language Modelling with DNN

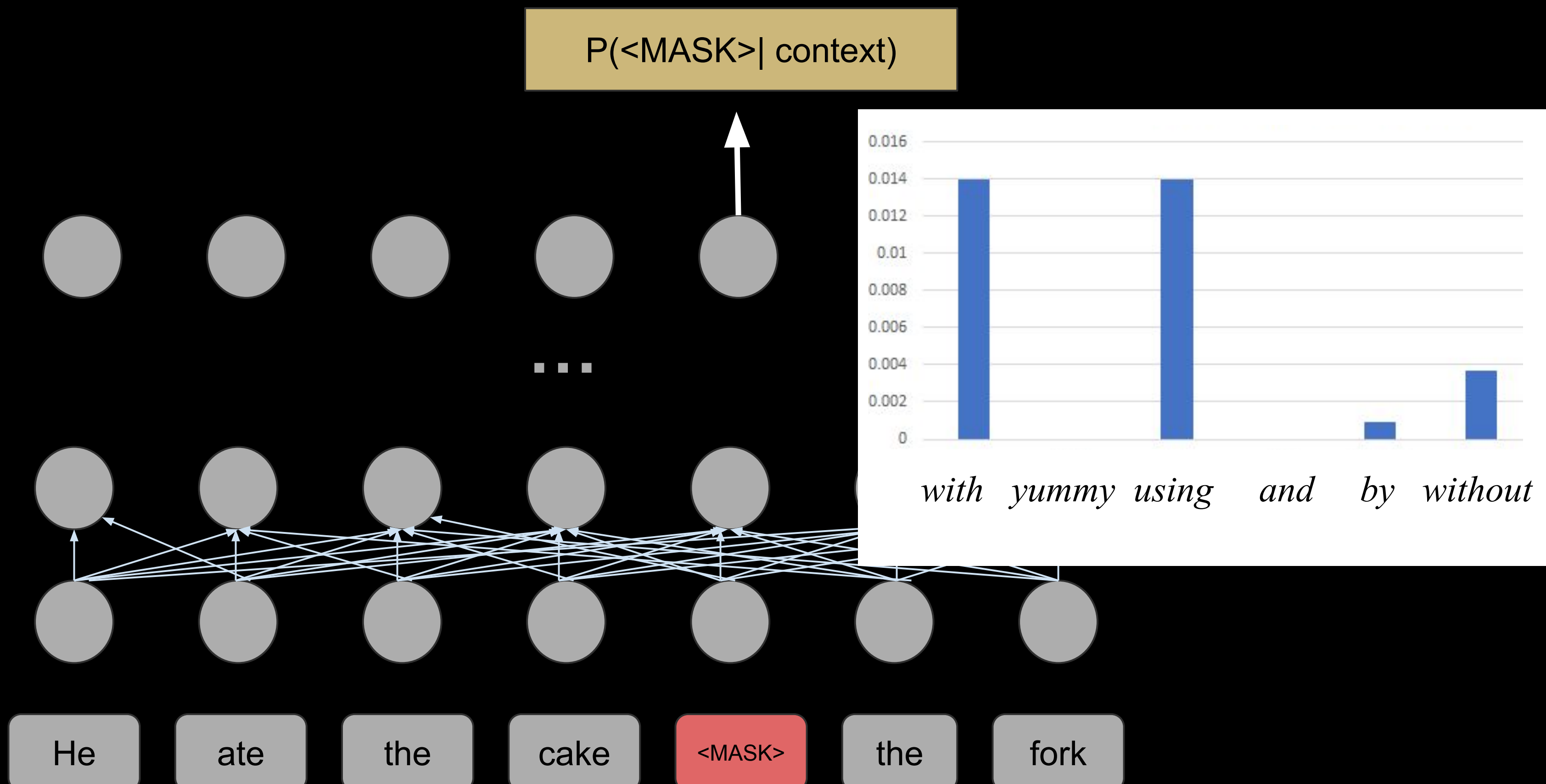
Masked Language Modelling with DNN



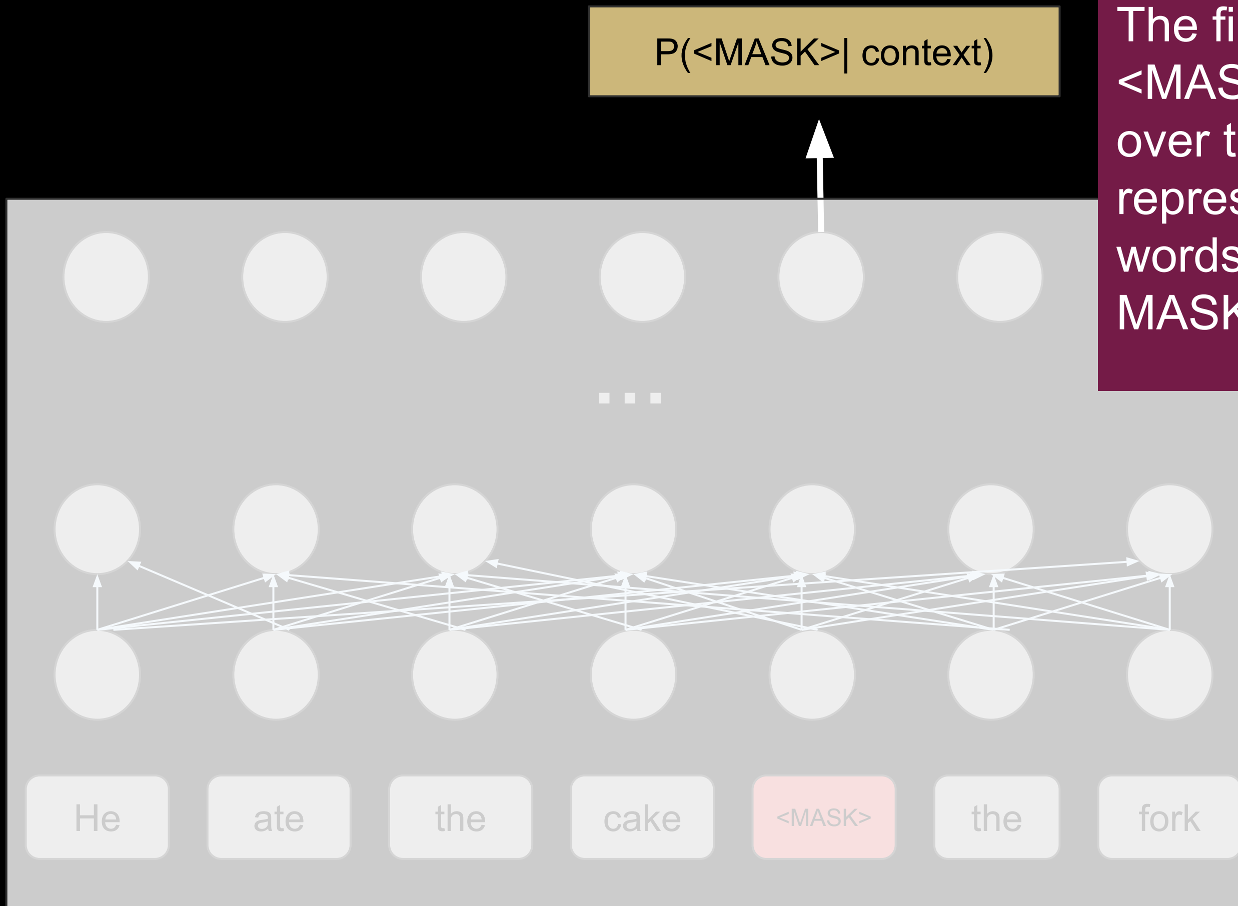
Masked Language Modelling with DNN



Masked Language Modelling with DNN

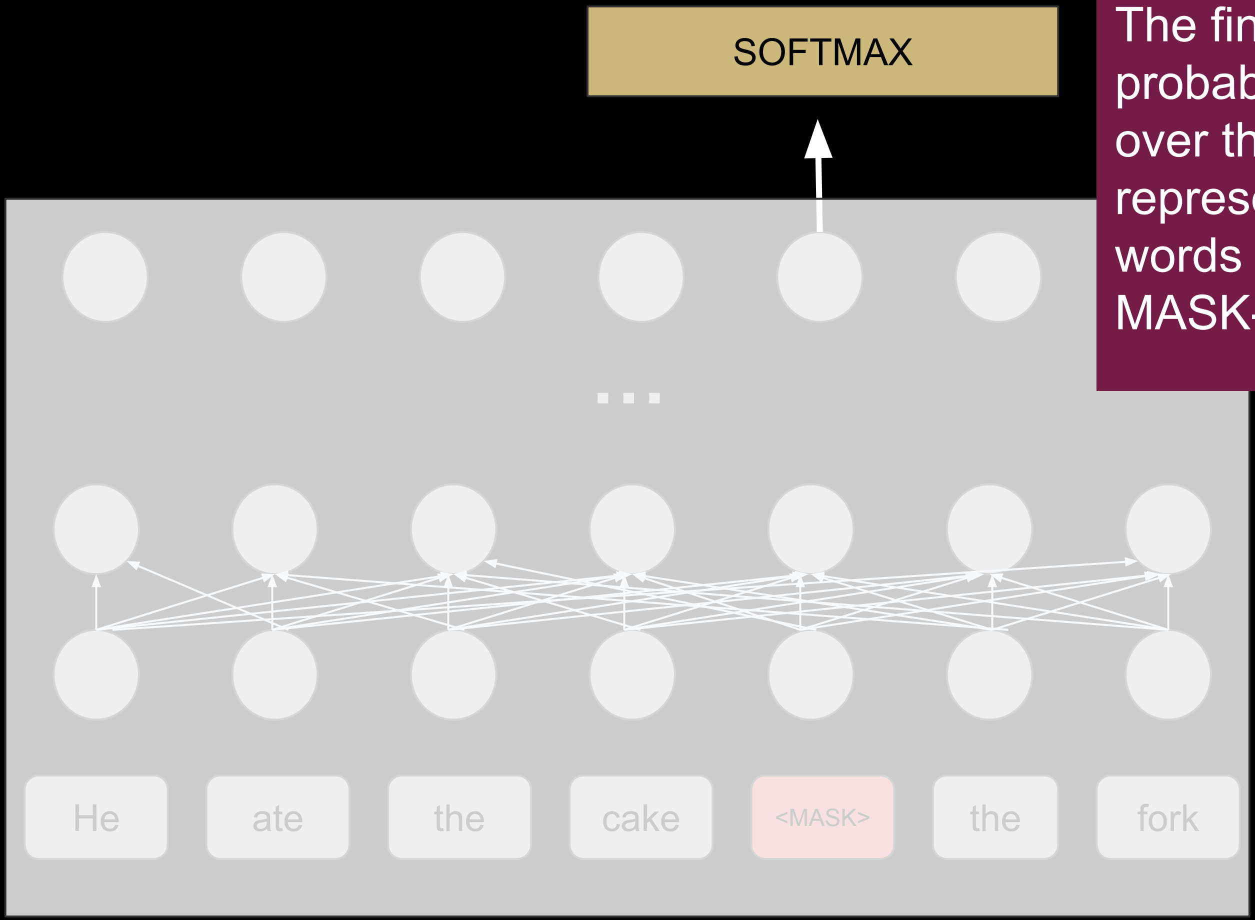


Masked Language Modelling with ANN



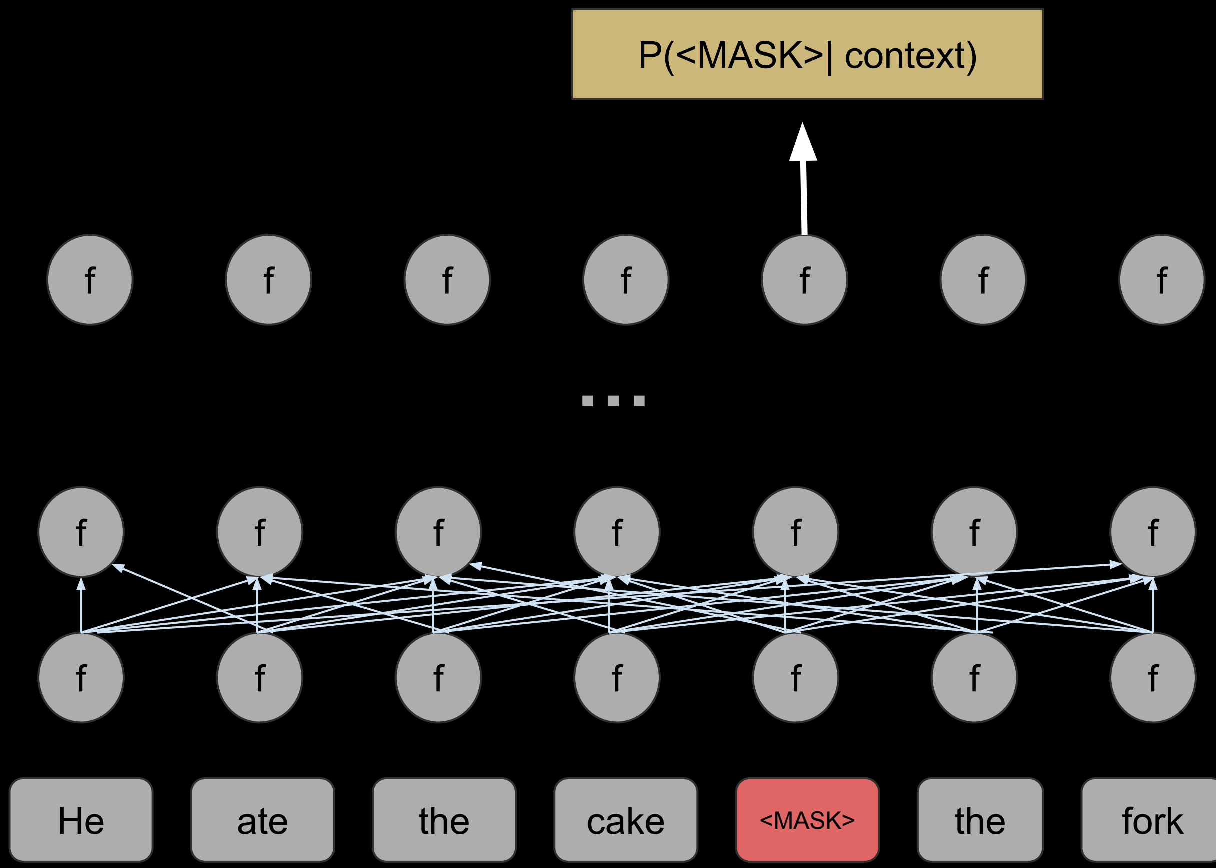
The final layer produces a <MASK> distribution over the vocabulary, representing the likely words to fill in the MASK-ed token

Masked Language Modelling with DNN

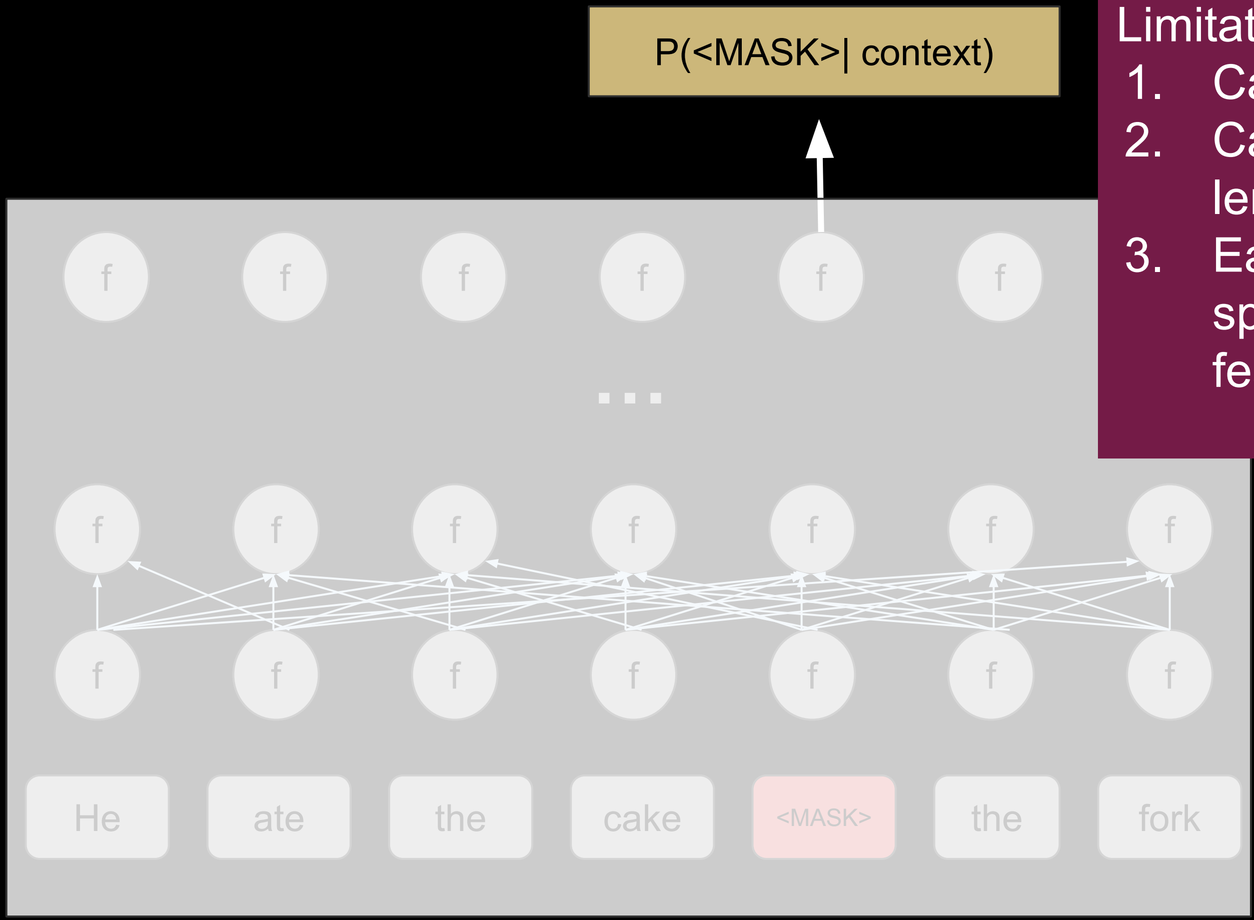


The final layer produces a probability distribution over the vocabulary, representing the likely words to fill in the MASK-ed token

Masked Language Modelling with DNN



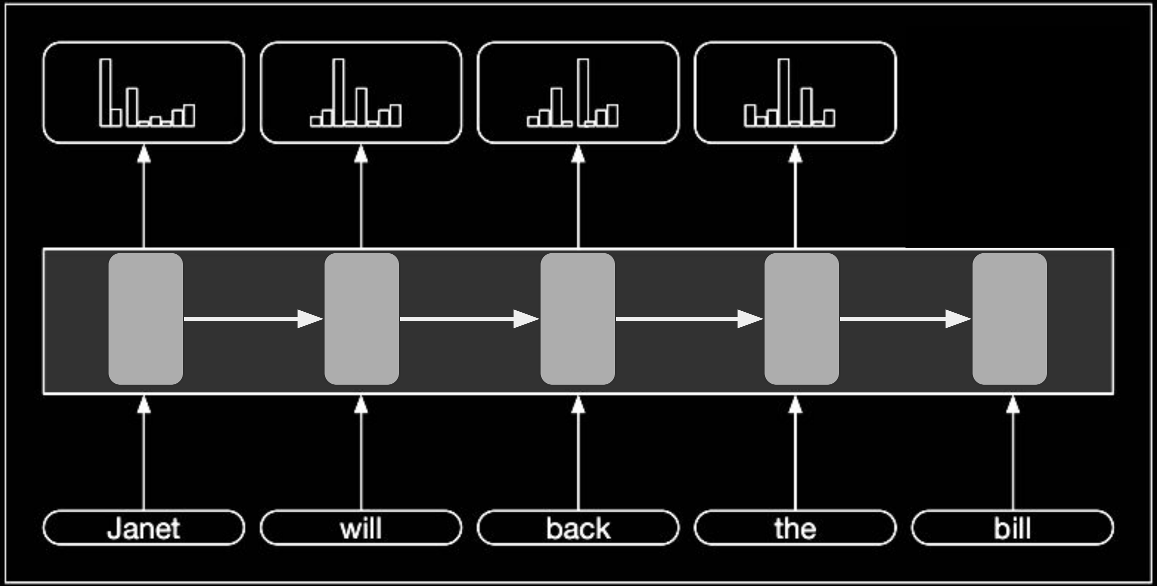
Masked Language Modelling with DNN



Limitations:

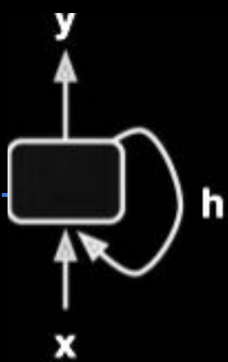
1. Can’t handle order
2. Can’t handle variable length sequences
3. Each parameter to specific to the input feature (token)

Recurrent Neural Network



Masked Language modeling with an RNN

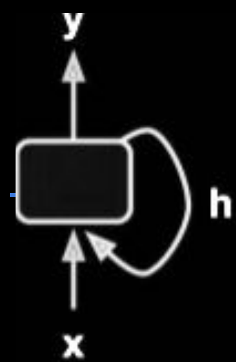
Example: Forward Pass



(Geron, 2017)

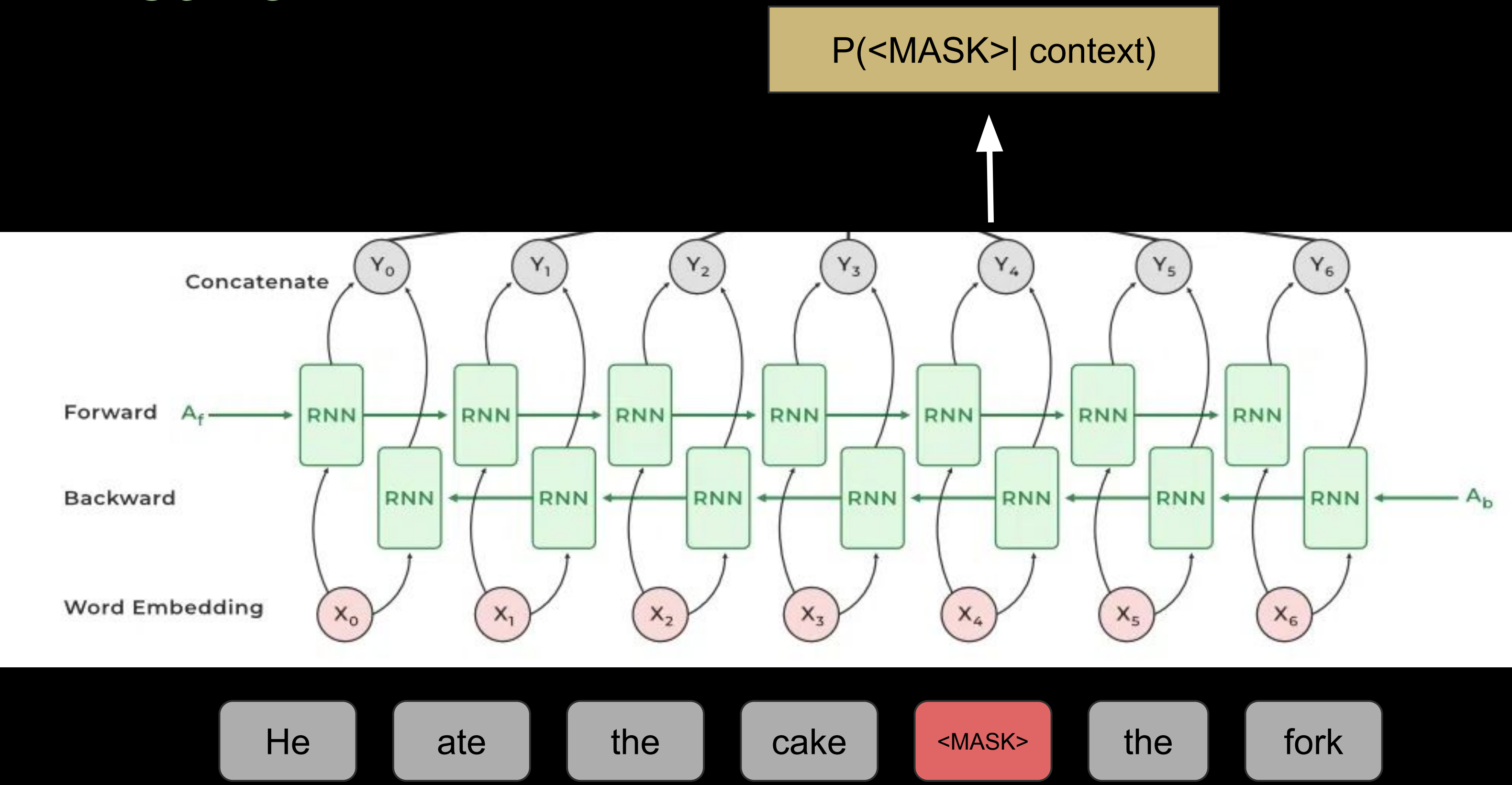
```
#define forward pass graph:
h(0) = 0
for i in range(1, len(x)):
    h(i) = g(U h(i-1) + W x(i)) #update hidden state
    y(i) = f(V h(i)) #update output
```

Example: Forward Pass




```
#define forward pass graph:
h(0) = 0
for i in range(1, len(x)):
     h(i) = tanh(matmul(U,h(i-1))+ matmul(W,x(i))) #update hidden state
     y(i) = softmax(matmul(V, h(i))) #update output
```

Masked Language Modelling with Recurrent Network

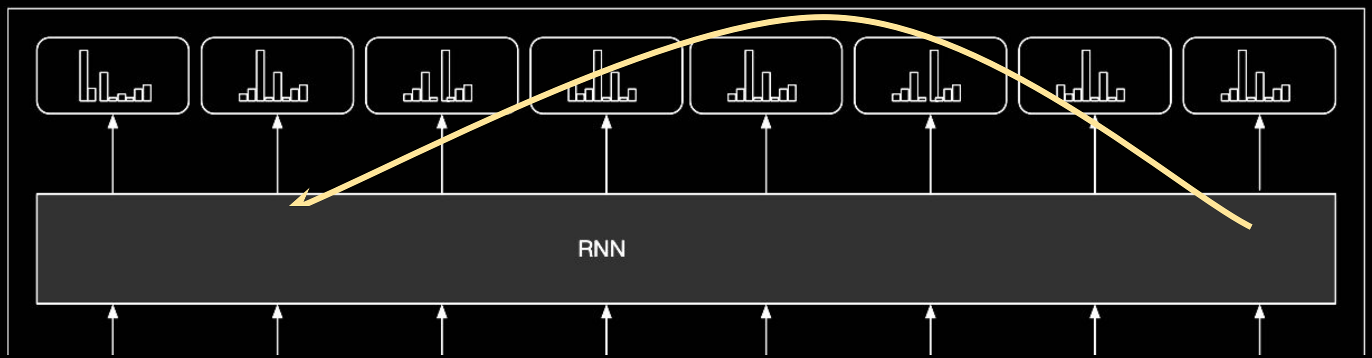


Vanishing/exploding gradients (Computational graph)

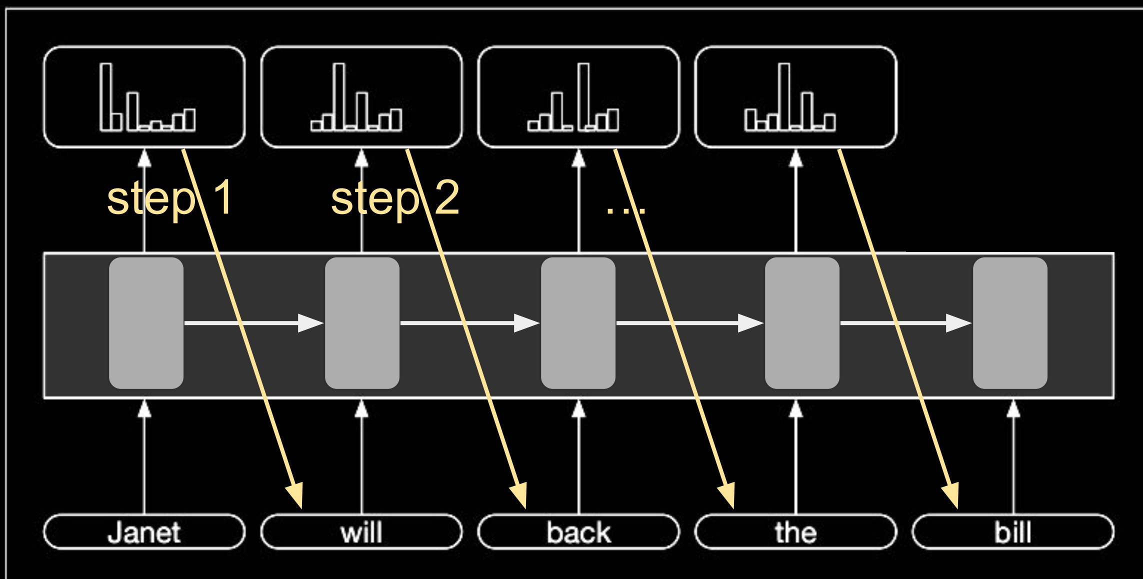
GRU and LSTM cells solve.

RNN Limitation:
Losing Track of Long Distance Dependencies

The horse which was raced past the barn tripped .



RNN: Limitation: Not parallelizable



Language modeling with an RNN

Next Lecture

- Deep dive into Self Attention (Vaswani et al., 2017)
- Masked Language Modelling using Transformers (Devlin et al., 2019)

Part 2: Transformer and Self-attention

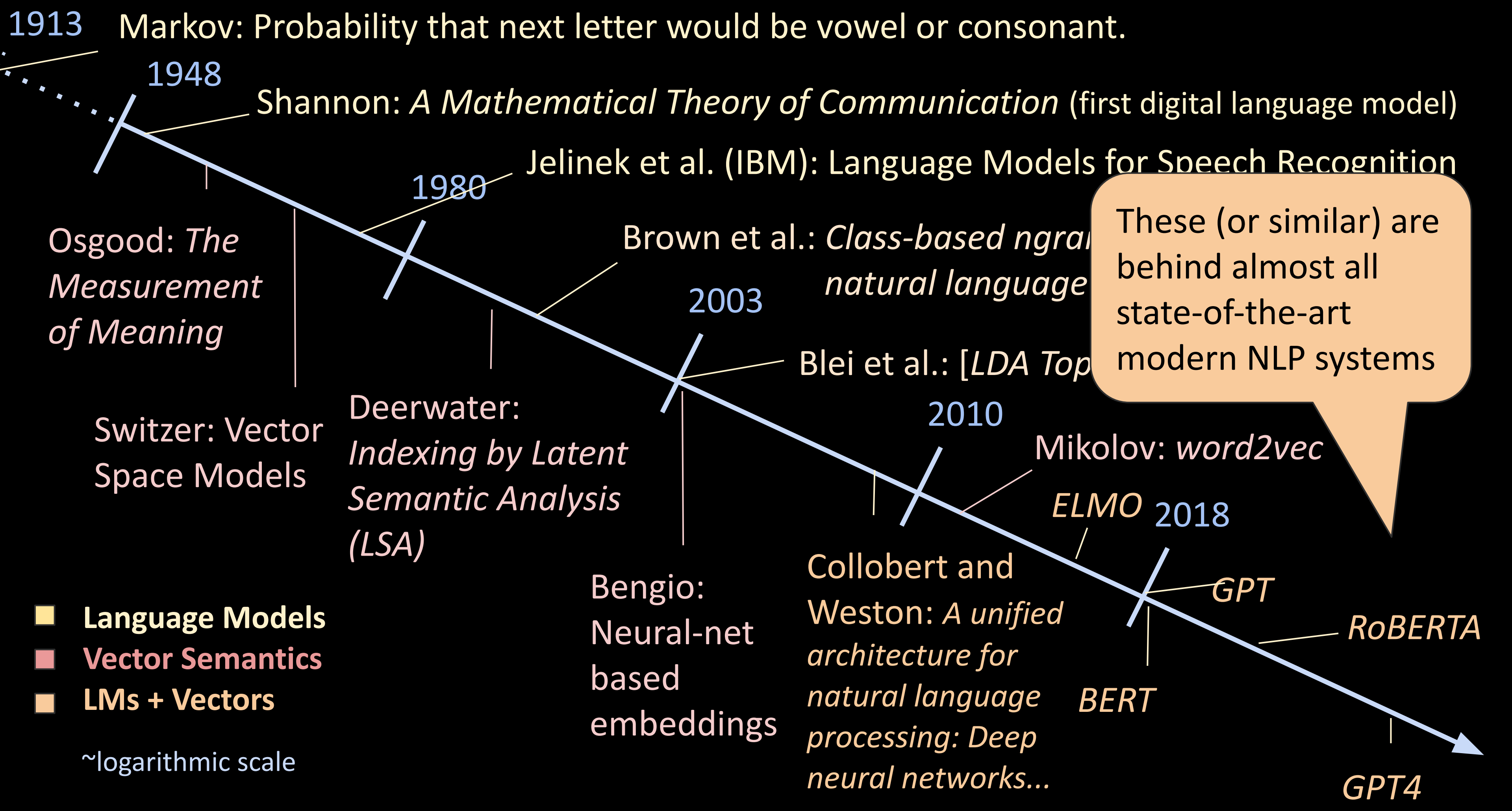
Nikita Soni
nisoni@cs.stonybrook.edu

CSE538 - Spring 2024

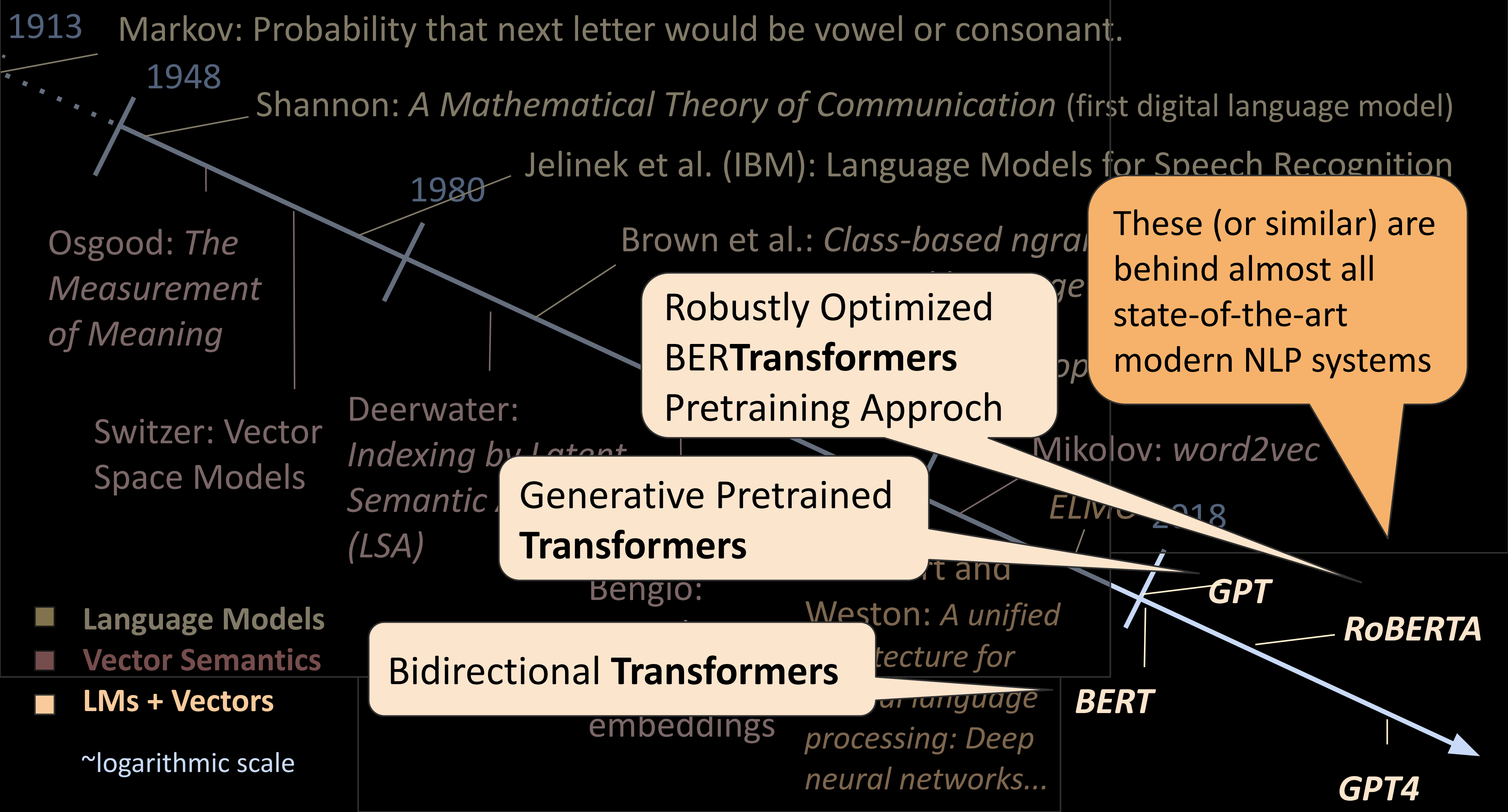
Recap: RNN Limitations

- Difficult to capture long-distance dependencies
- Not parallelizable -- need sequential processing.
    - Slow computation for long sequences
- Vanishing or exploding gradients

Timeline: *Language Modeling* and *Vector Semantics*



Timeline: *Language Modeling* and *Vector Semantics*



The Transformer: Motivation

- Capture long-distance dependencies
- Preserving sequential distances / periodicity
- Capture multiple relationships
- Easy to parallelize -- don’t need sequential processing.

Introducing the Transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

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

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

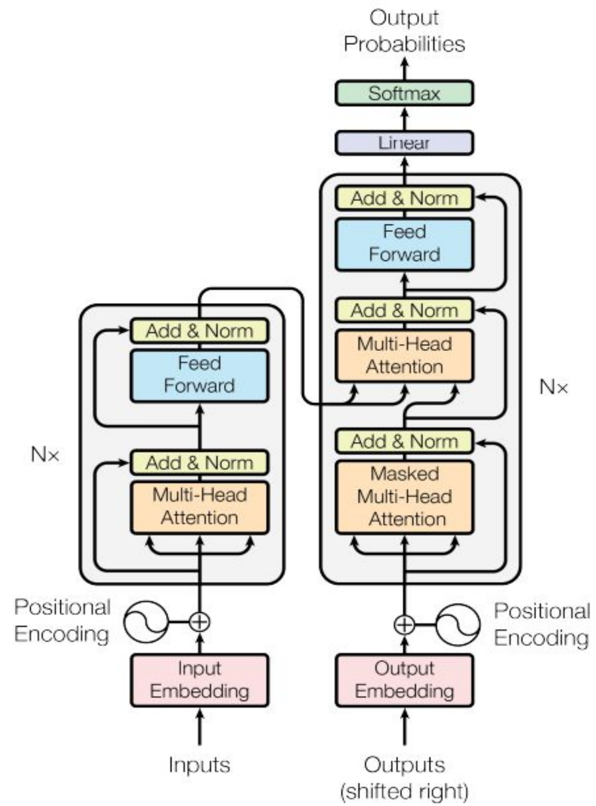


Figure 1: The Transformer - model architecture.

Introducing the Transformer

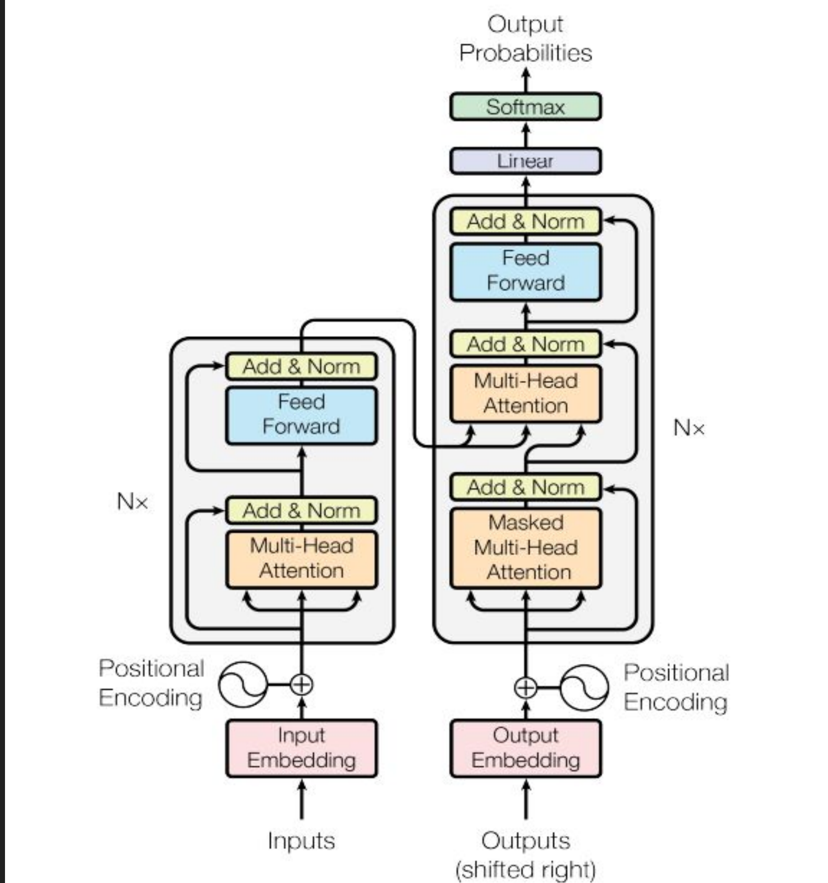
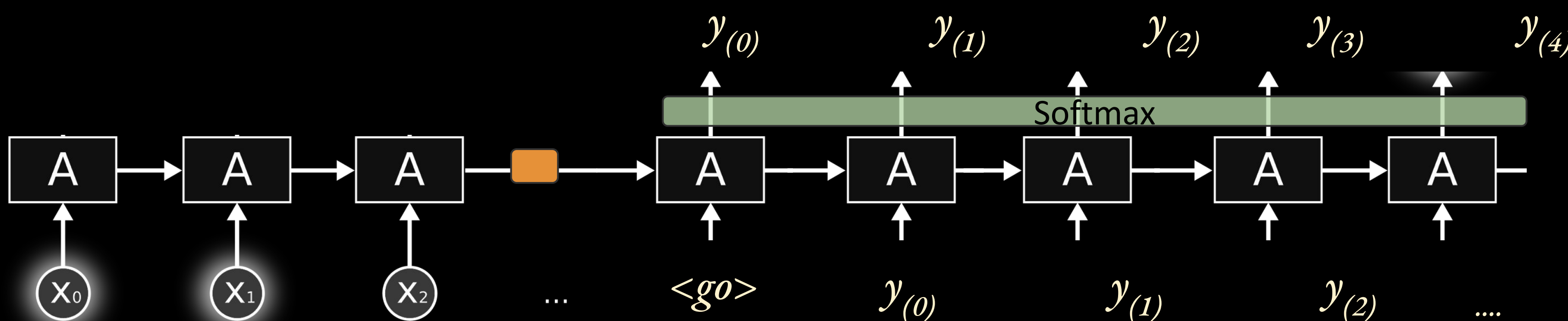
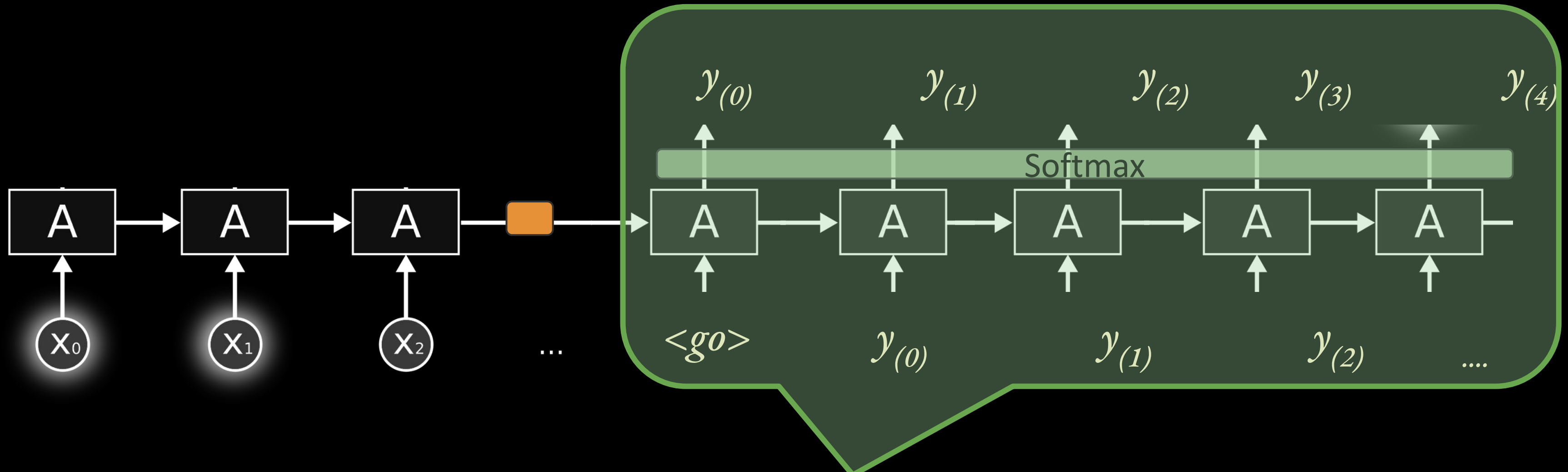


Figure 1: The Transformer - model architecture.

Encoder-Decoder (Simpler Representation)

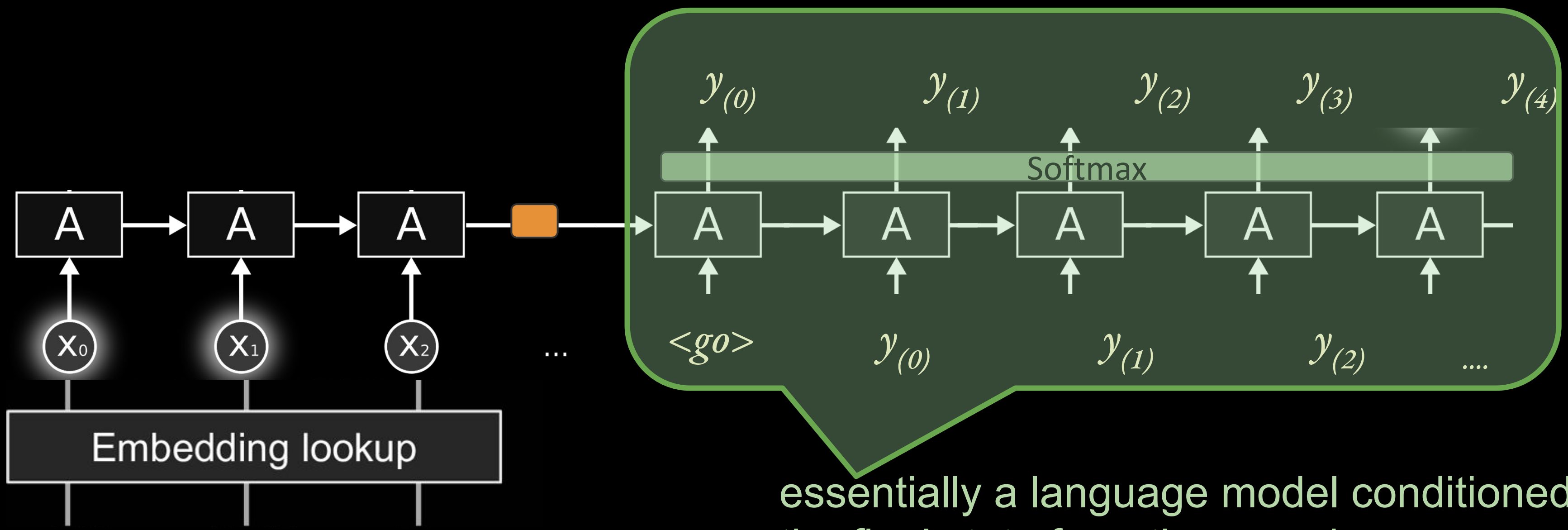


Encoder-Decoder (Simpler Representation)



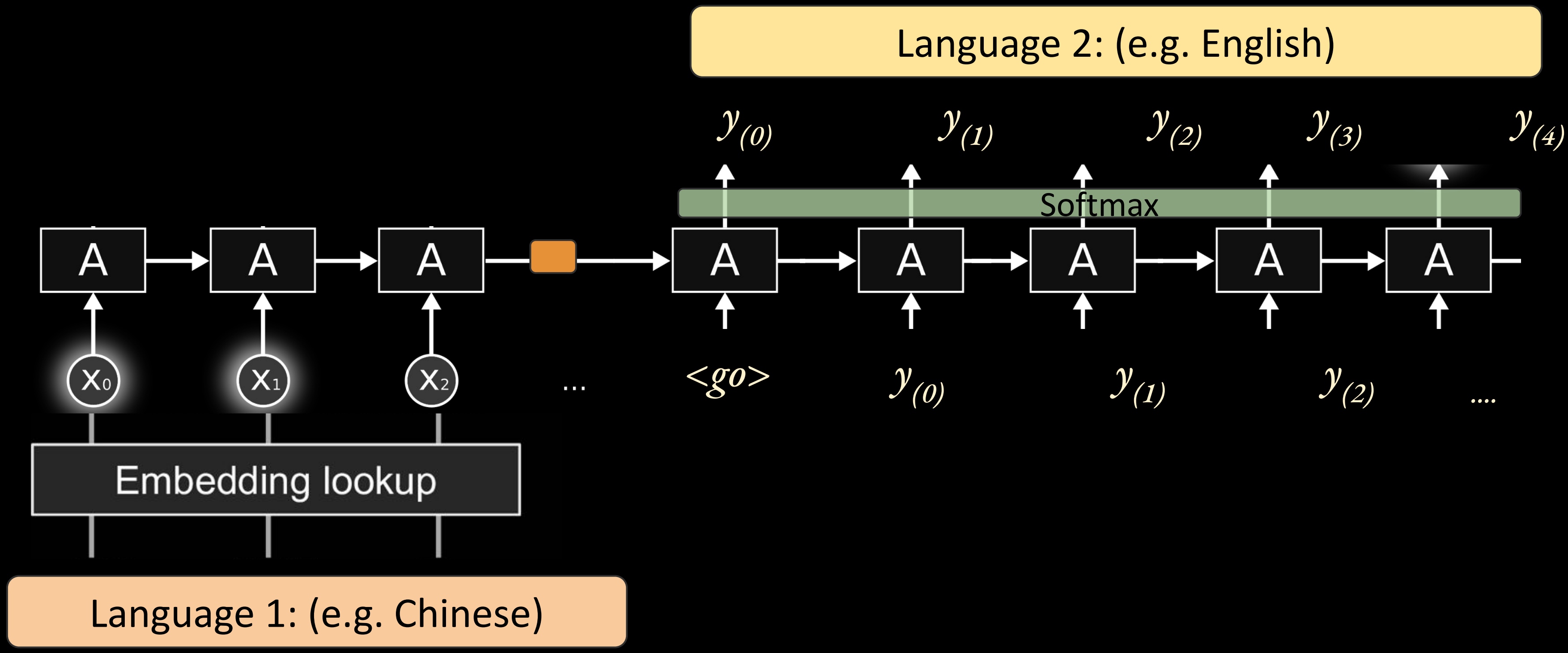
essentially a language model conditioned on the final state from the encoder.

Encoder-Decoder (Simpler Representation)



essentially a language model conditioned on the final state from the encoder.

Encoder-Decoder (Simpler Representation)

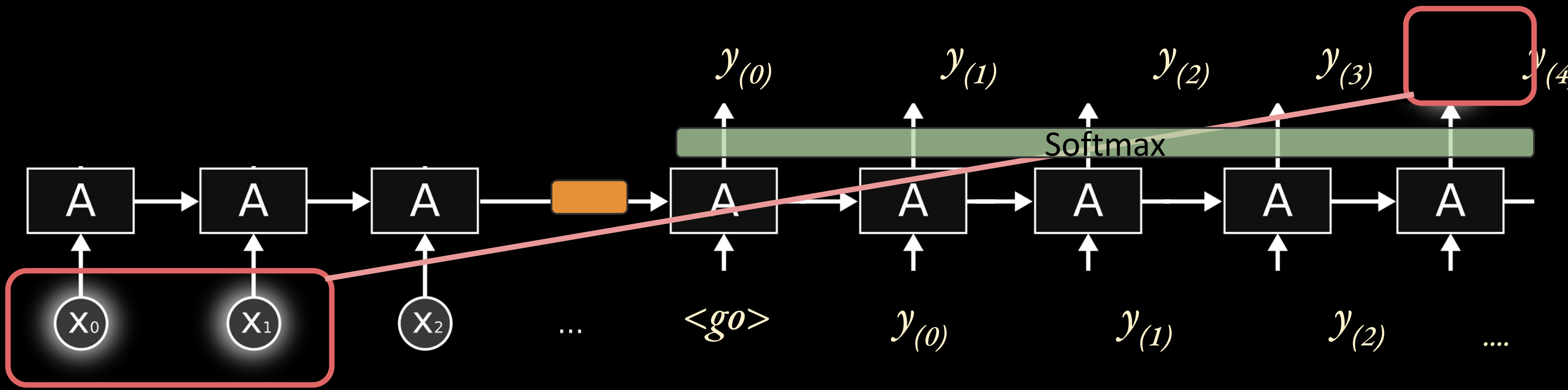


Encoder-Decoder

Challenge:

- Long distance dependency when translating:

The ball was kicked by kayla.



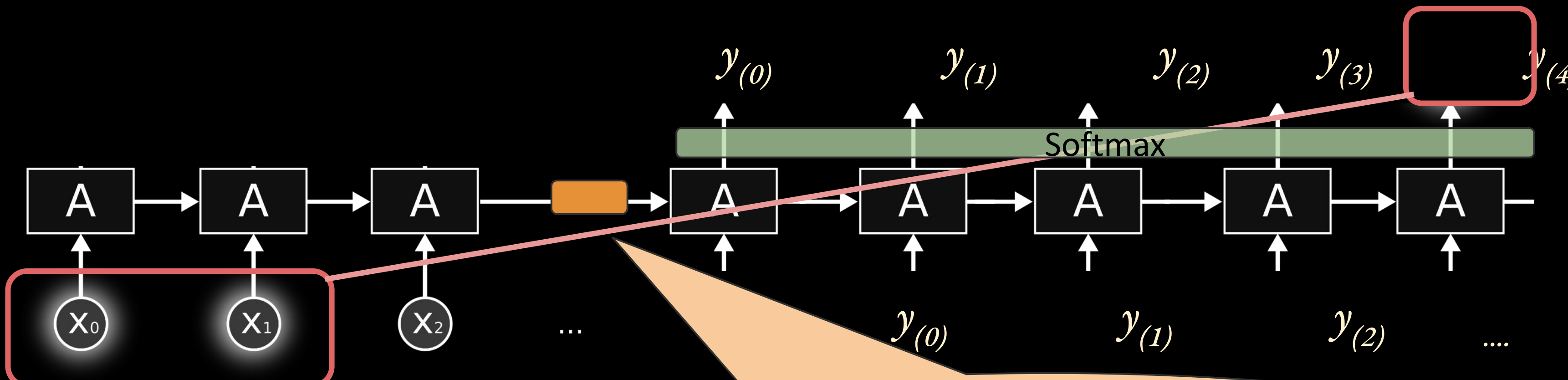
Kayla kicked the ball.

Encoder-Decoder

Challenge:

The ball was kicked by kayla.

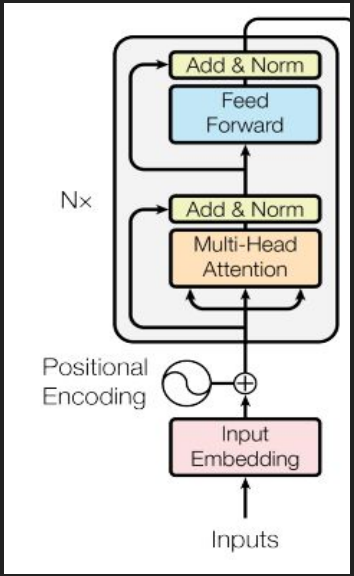
- Long distance dependency when translating:



Kayla kicked the ball.

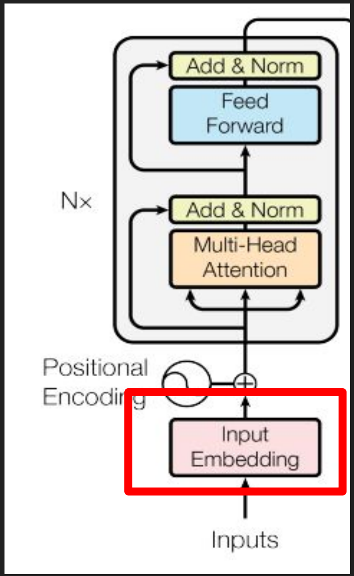
A lot of responsibility put fixed-size hidden state passed from encoder to decoder

Encoder



Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

Encoder: Input Embedding



Input Embedding

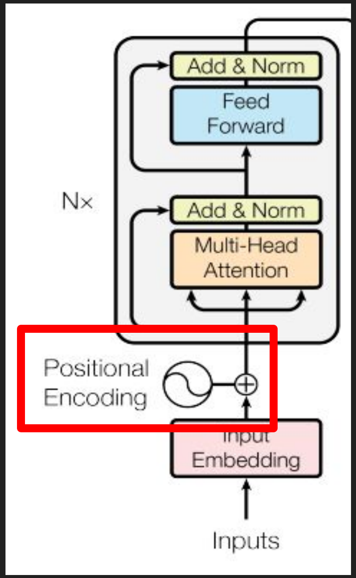
Original Sentence

Tokenization

Input IDs
(embedding lookup: position in the vocab - FIXED)

Embeddings
(vector of size d_{model}= 512 or 1024 or … LEARNED)

Encoder: Positional Encoding



Positional Encoding

Original Sentence
(tokens)

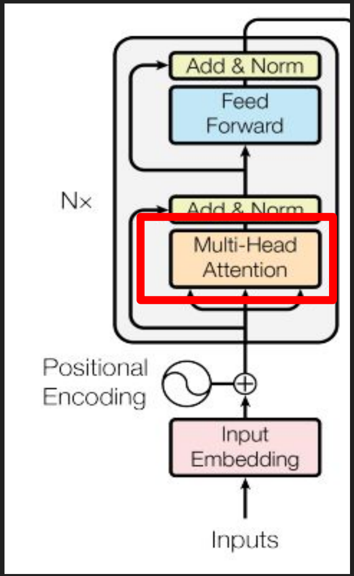
Embeddings
(vector of size d_{model}= 512 or 1024 or …
Learned)

Positional Embedding
(vector of size d_{model}= 512 or 1024 or …
Can be Learned or FIxed)

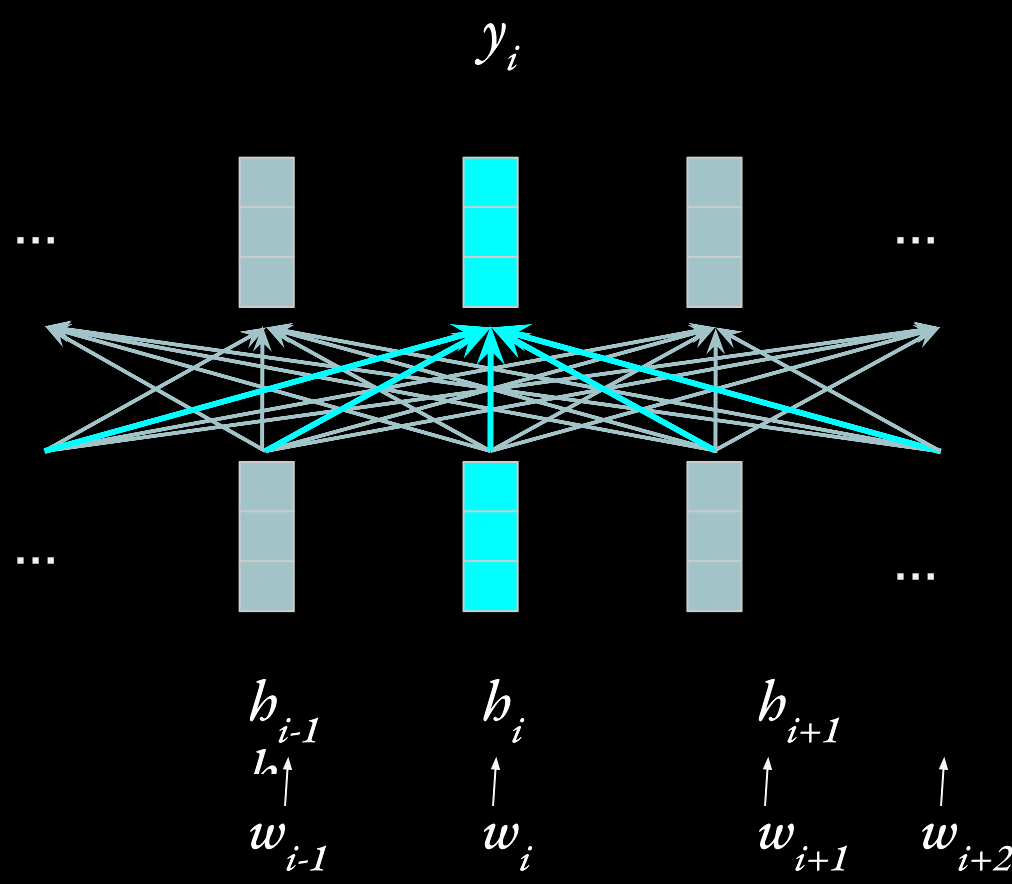
Positional Encoding

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

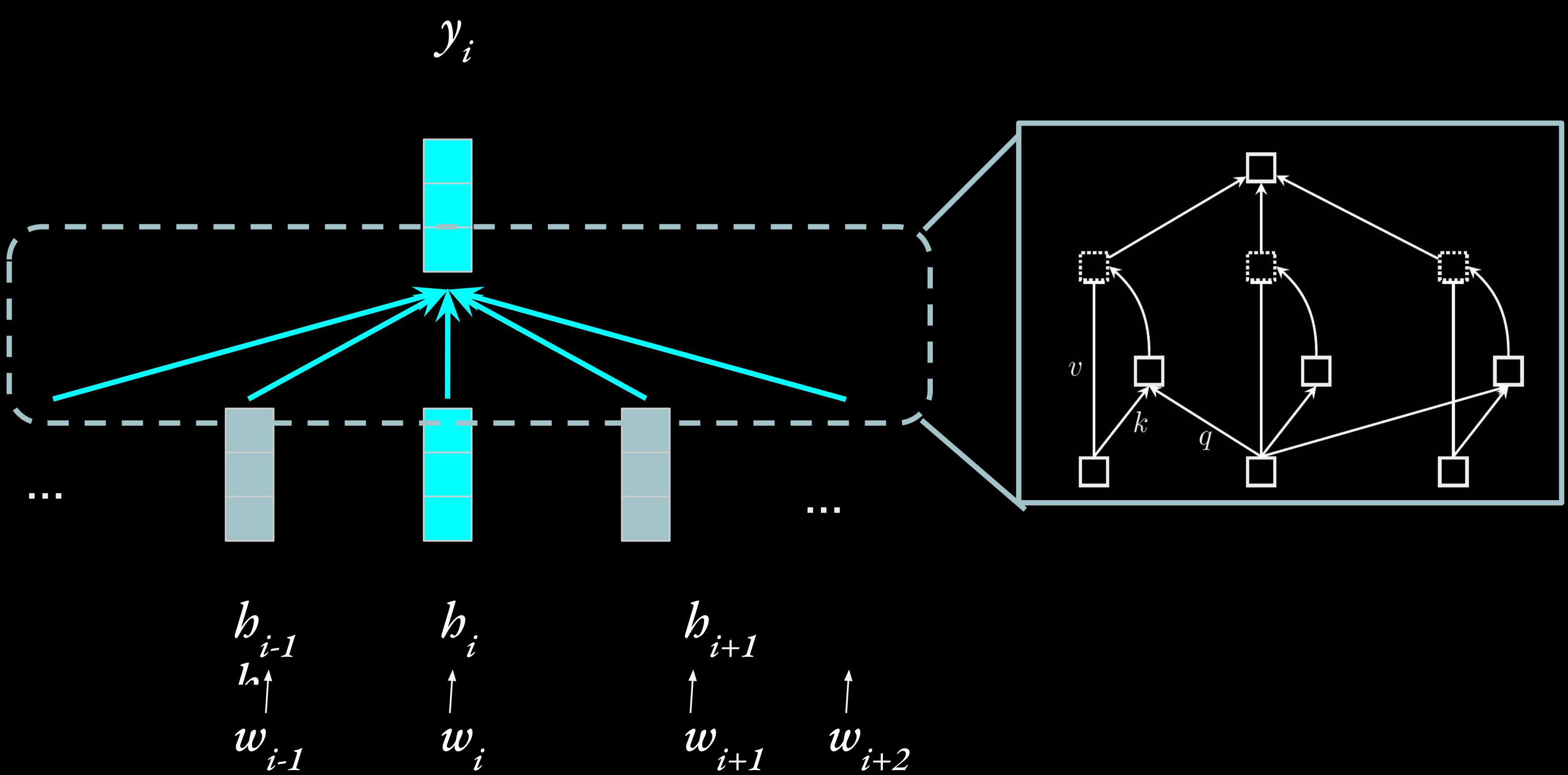
Encoder: Multi-Head Attention



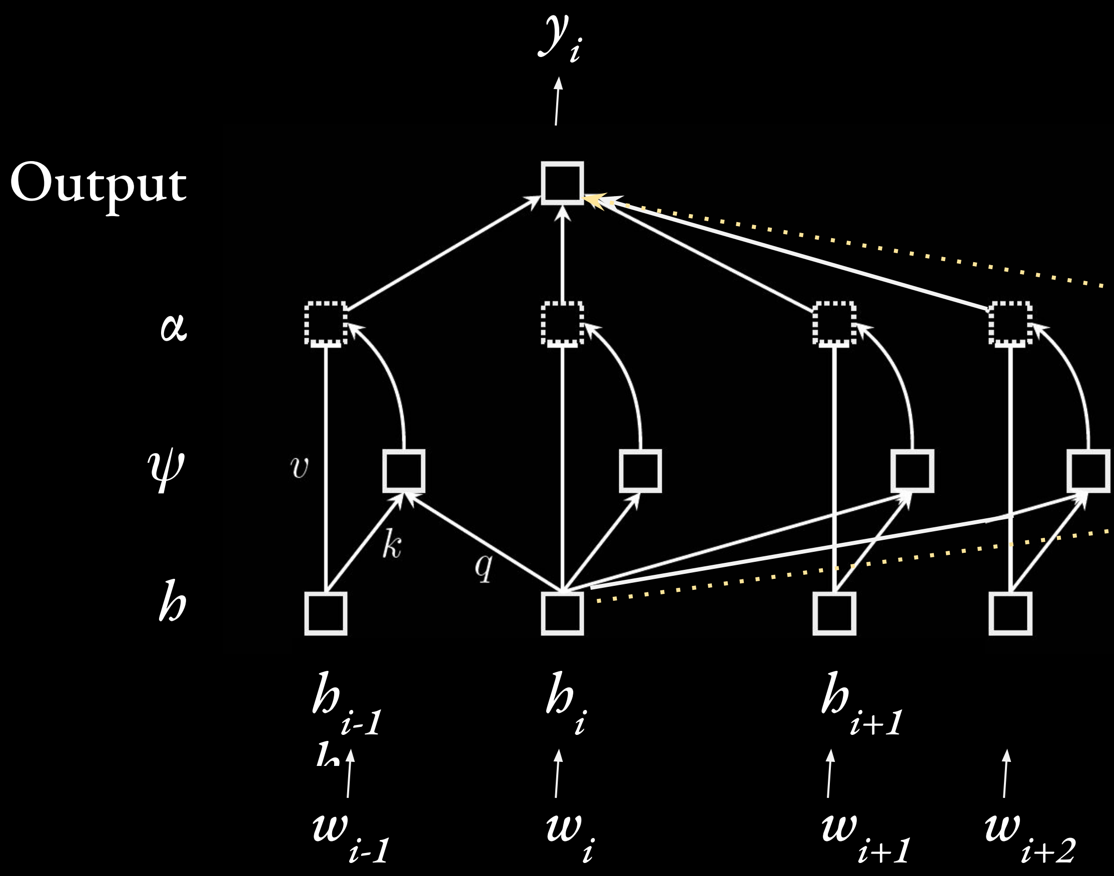
The Transformer's Heart: Self-Attention



The Transformer's Heart: Self-Attention

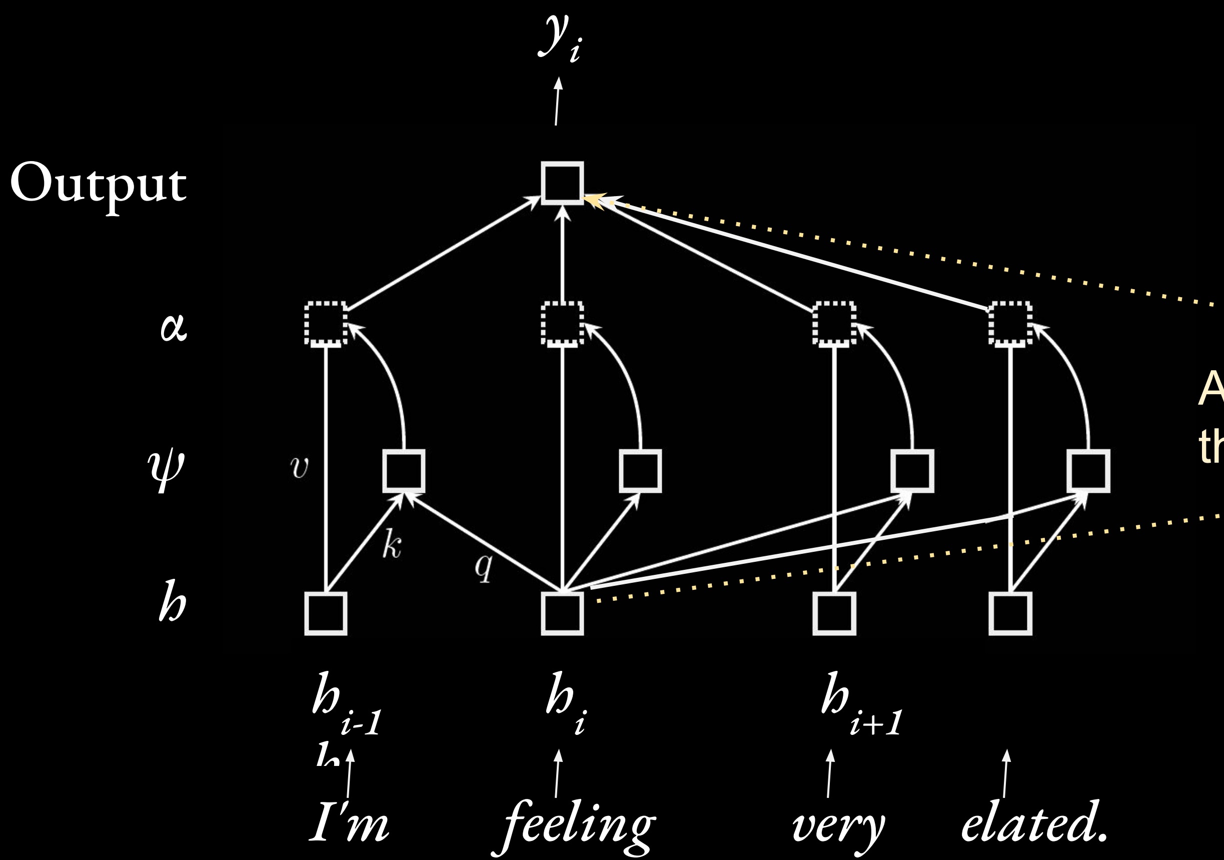


The Transformer's Heart: Self-Attention



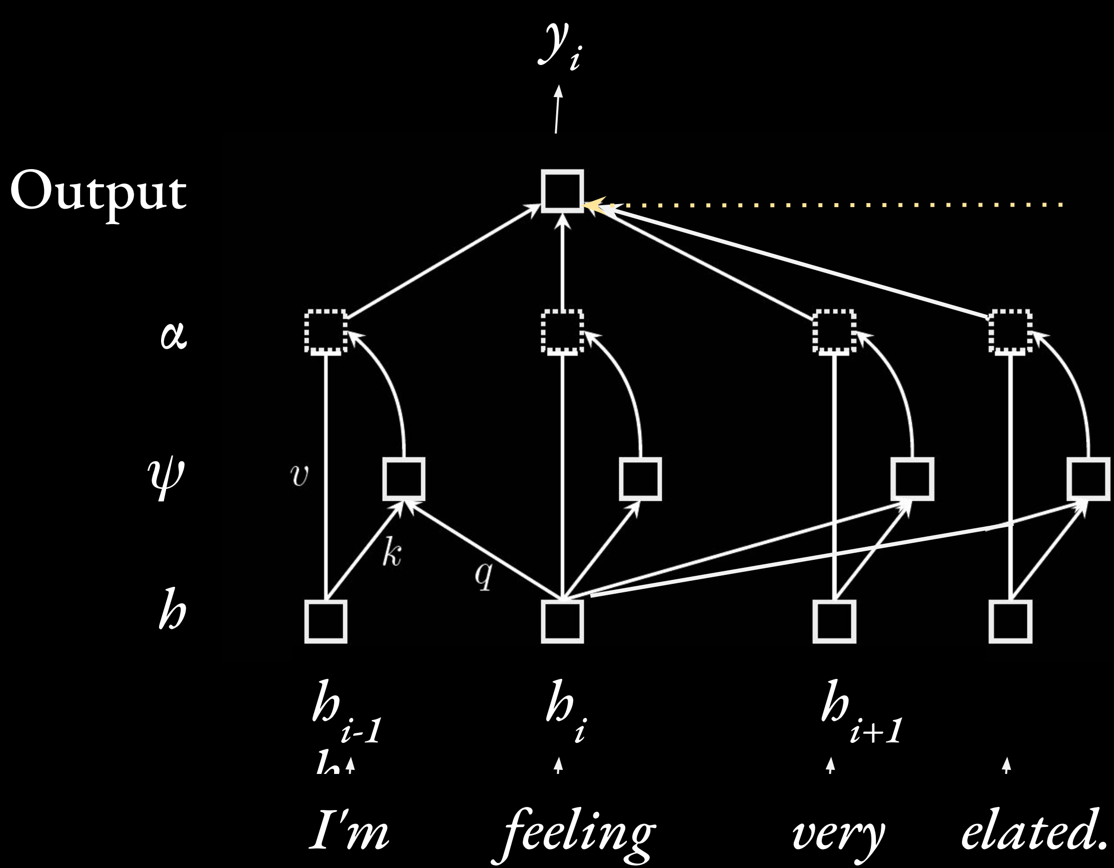
Attend to all other words in the sequence

The Transformer's Heart: Self-Attention



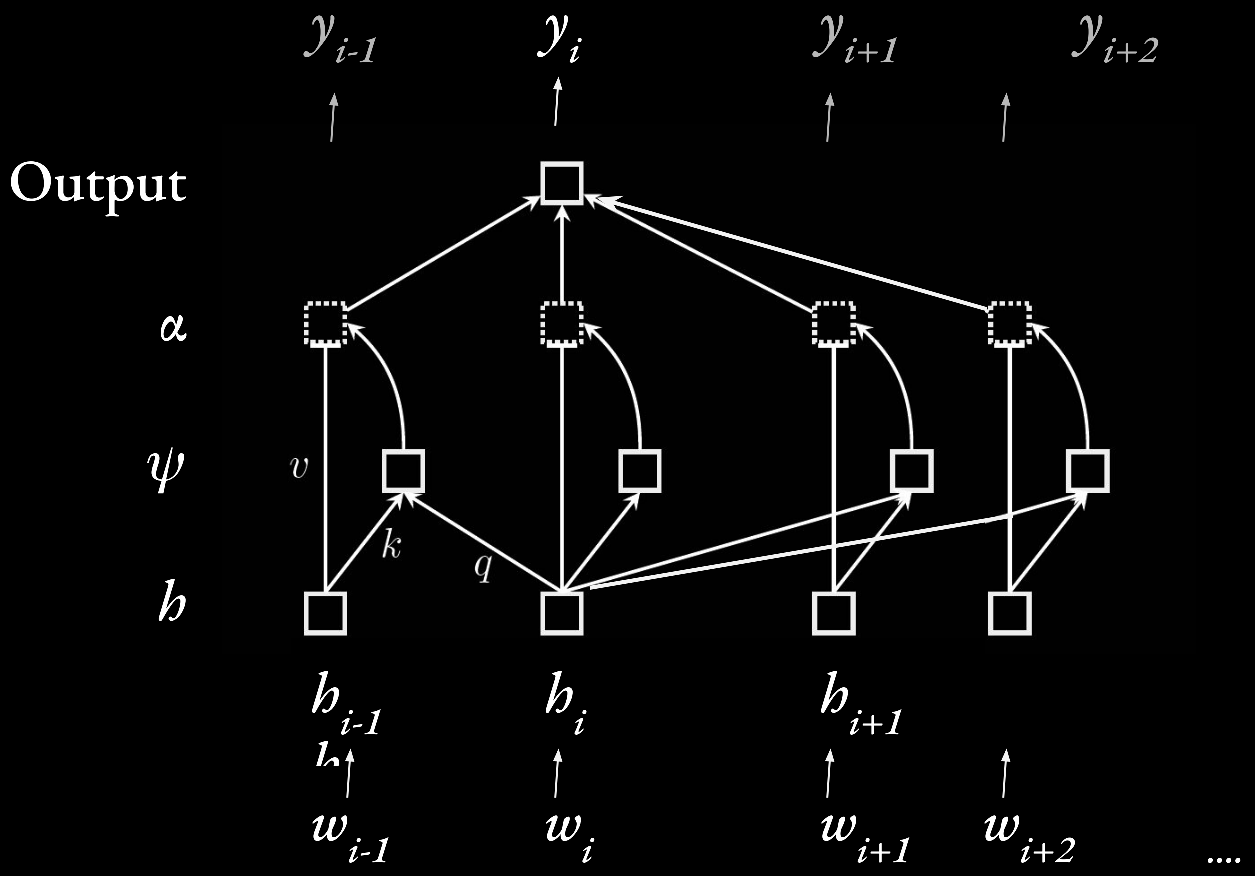
Attend to all other words in the sequence

The Transformer's Heart: Self-Attention

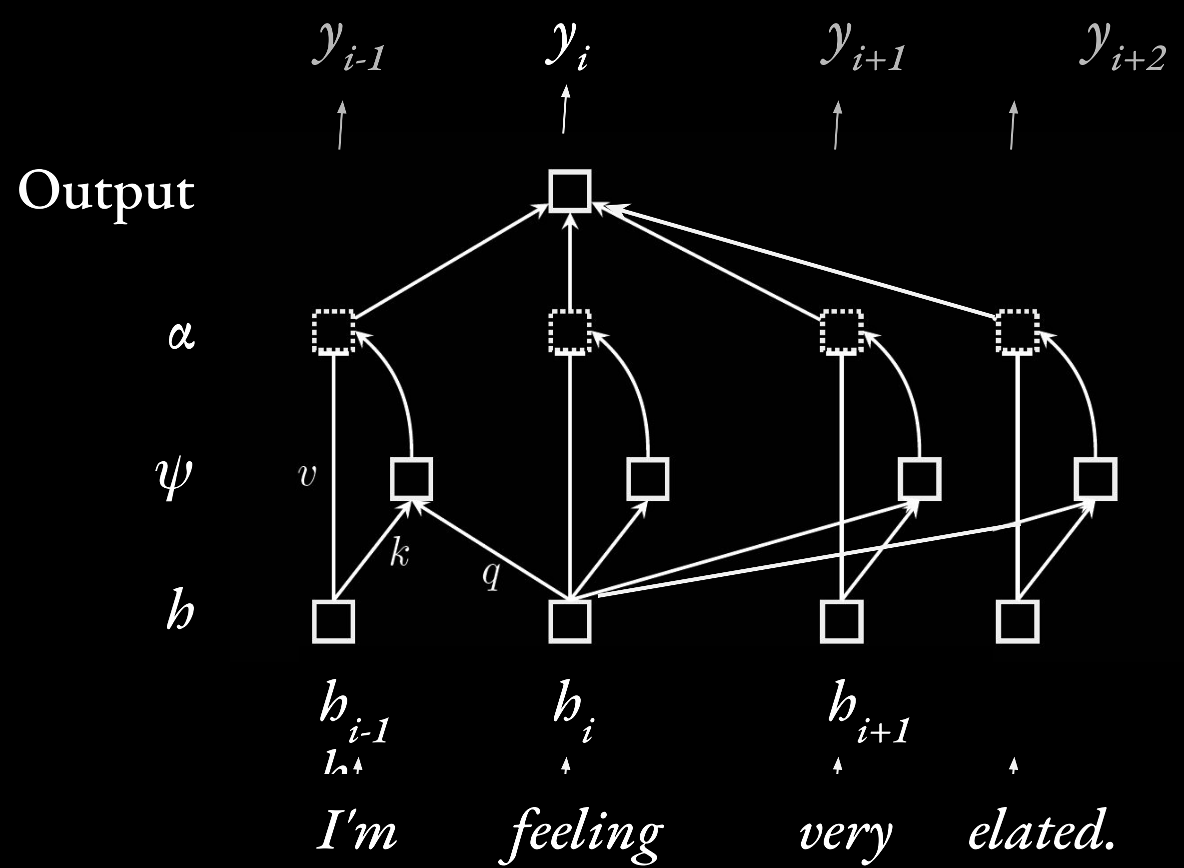


A weighted combination of other words' vectors.

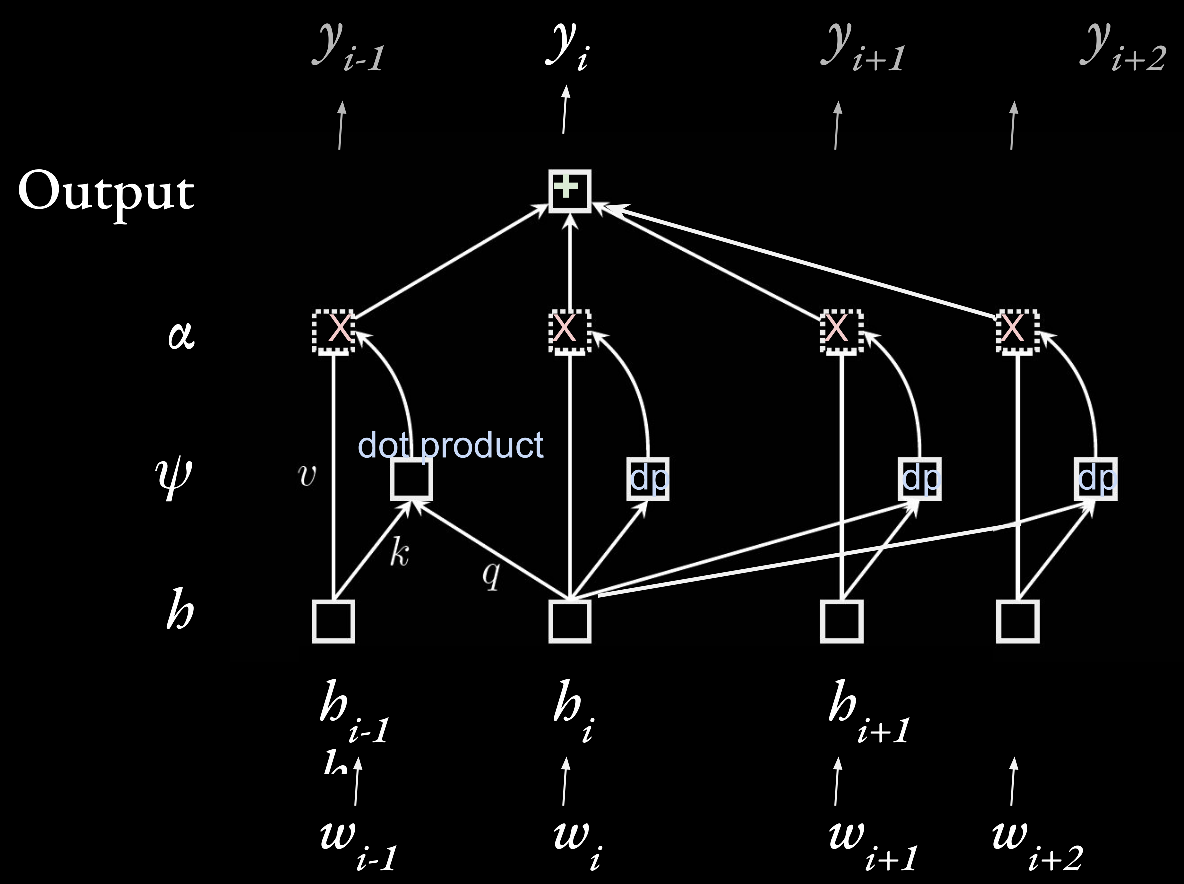
The Transformer's Heart: Self-Attention



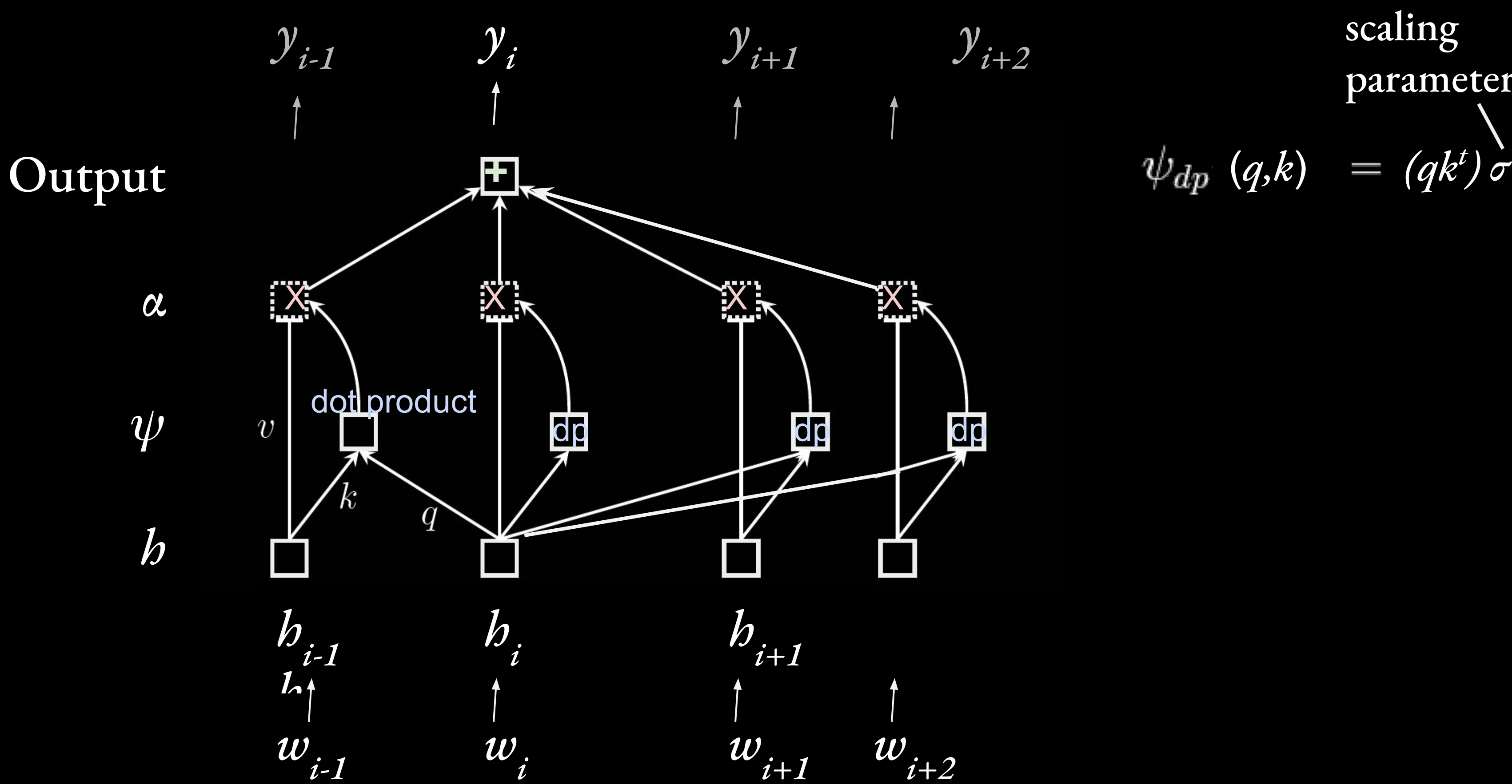
The Transformer's Heart: Self-Attention



The Transformer's Heart: Self-Attention



The Transformer's Heart: Self-Attention



Notations for Self-Attention (Matrix multiplication, Dot Product, Sequence length (s), embedding dimensions)

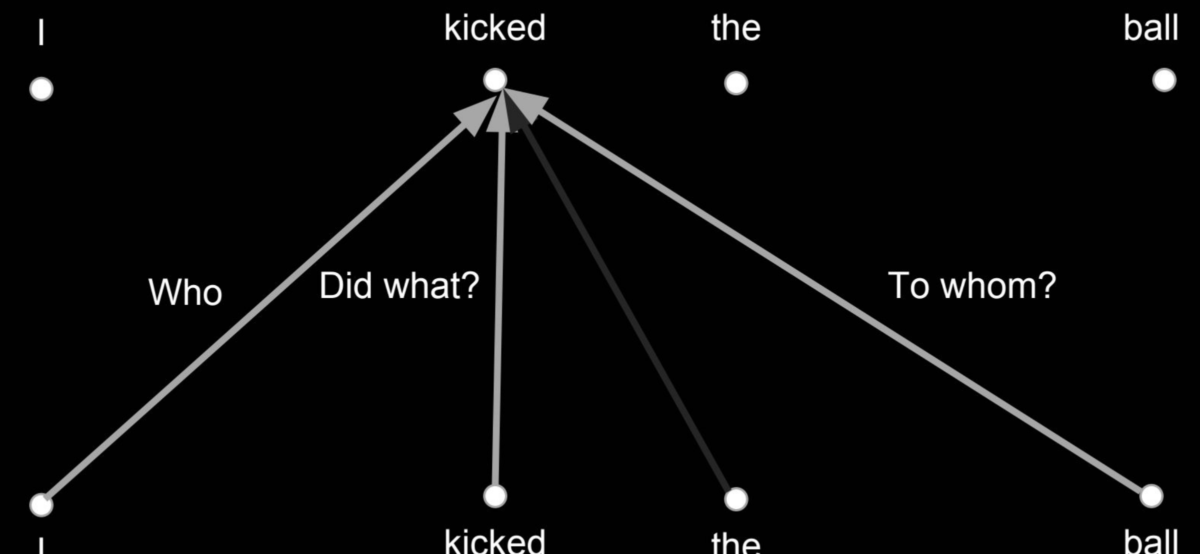
Input matrix: [s, d_{model}]

Self-Attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

The Transformer: Beyond Self-Attention

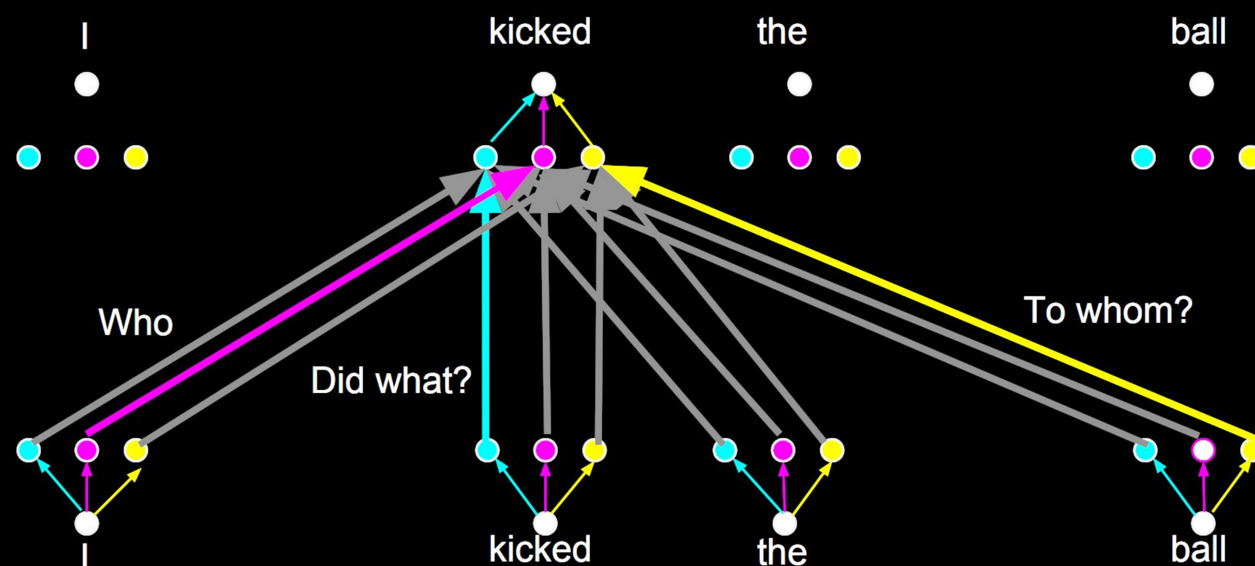
Limitation (thus far): Can't capture multiple types of dependencies between words.



The Transformer: Beyond Self-Attention

Limitation (thus far): Can't capture multiple types of dependencies between words.

Solution: Multi-head attention



Self-Attention: Weights

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-Headed Attention

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

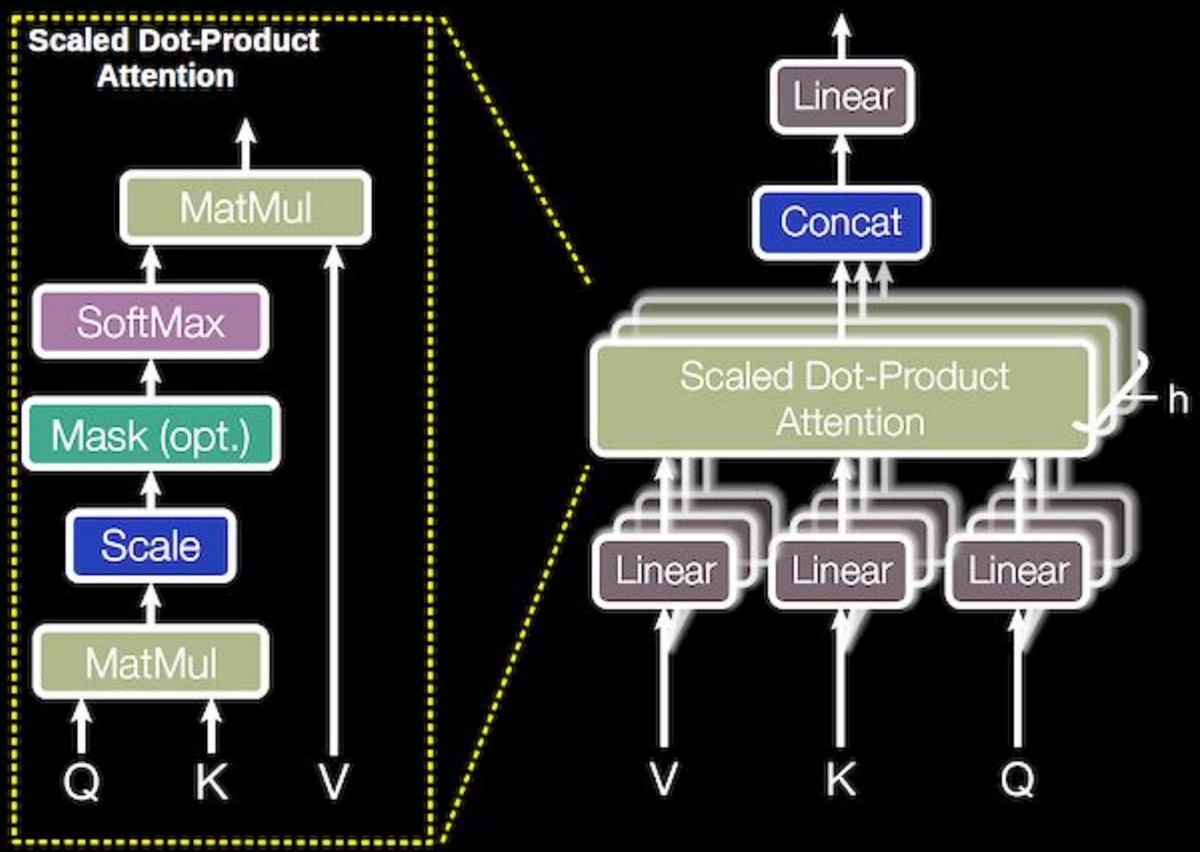
Multi-Headed Attention

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Linear layer:
$W^T X$

One set of weights for each of K, Q, and V

The Transformer: Multi-headed Attention



Self-Attention in PyTorch

```
import nn.functional as f
class SelfAttention(nn.Module):
    def __init__(self, h_dim:int):
        self.Q = nn.Linear(h_dim, h_dim) #1 head
        self.K = nn.Linear(h_dim, h_dim)
        self.V = nn.Linear(h_dim, h_dim)


    def forward(hidden_states:torch.Tensor):
        v = self.V(hidden_states)
        k = self.K(hidden_states)
        q = self.Q(hidden_states)
        attn_scores = torch.matmul(q, k.T)
        attn_probs = f.Softmax(attn_scores)

        context = torch.matmul(attn_probs, v)
        return context
```

$$\psi_{dp}(q,k) = (qk^t)\,\sigma$$

Linear layer:
$W^T X$

One set of weights for each of K, Q, and V

Self-Attention in PyTorch

```python
import nn.functional as f
class SelfAttention(nn.Module):
    def __init__(self, h_dim:int):
        self.Q = nn.Linear(h_dim, h_dim) #1 head
        self.K = nn.Linear(h_dim, h_dim)
        self.V = nn.Linear(h_dim, h_dim)
        self.dropout = nn.dropout(p=0.1)

    def forward(hidden_states:torch.Tensor):
        v = self.V(hidden_states)
        k = self.K(hidden_states)
        q = self.Q(hidden_states)
        attn_scores = torch.matmul(q, k.T)
        attn_probs = f.Softmax(attn_scores)
        attn_probs = self.dropout(attn_probs)
        context = torch.matmul(attn_probs, v)
        return context
```

$$\psi_{dp}(q,k) = (qk^t)\sigma$$

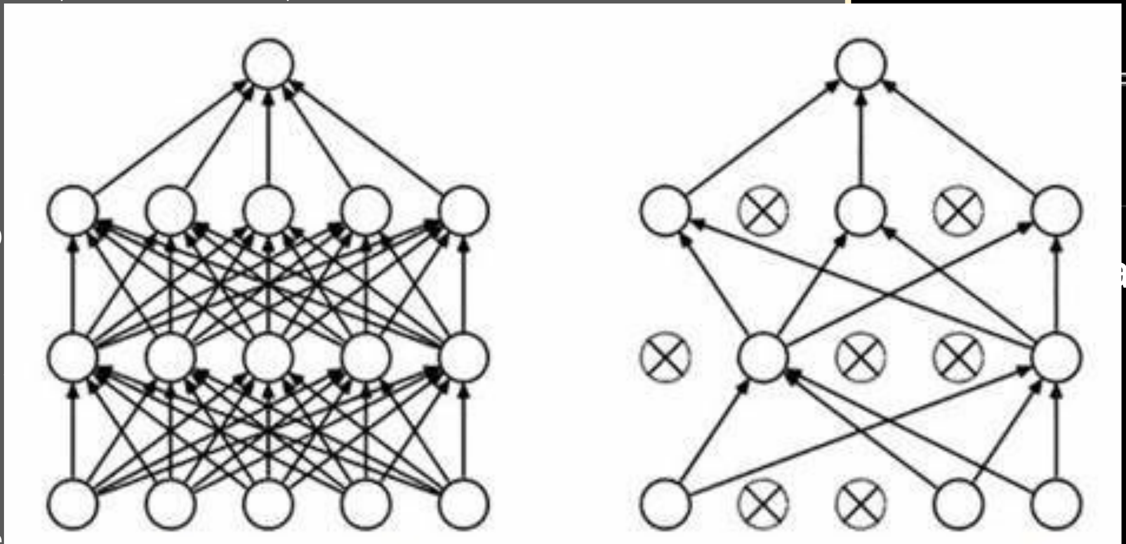
Linear layer:
$W^T X$

One set of weights for each of K, Q, and V

Self-Attention in PyTorch

```
import nn.functional as f
class SelfAttention(nn.Module):
    def __init__(
        self.Q =
        self.K =
        self.V =
        self.drop

    def forward(h
        v = self.
        k = self.
        q = self.
        attn_scor
        attn_prob
        attn_probs = self.dropout(attn_probs)
        context = torch.matmul(attn_probs, v)
        return context
```



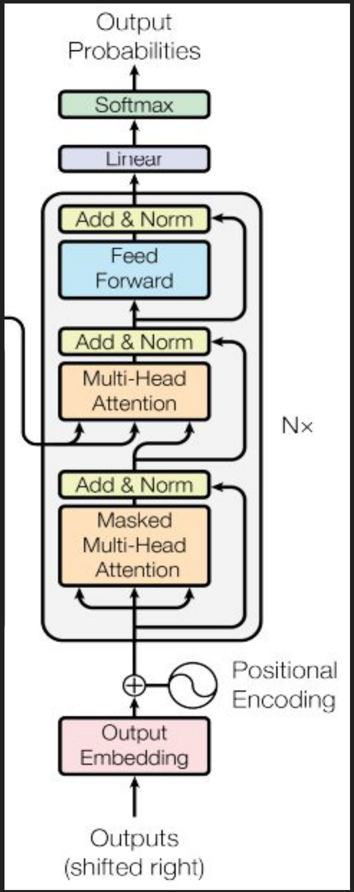
(a) Standard Neural Net (b) After applying dropout.

$= (qk^t)\,\sigma$

ayer:

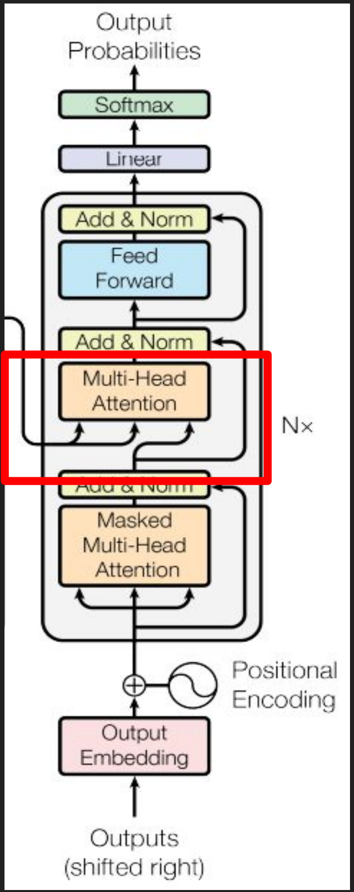
of weights
of for K,
V

Decoder

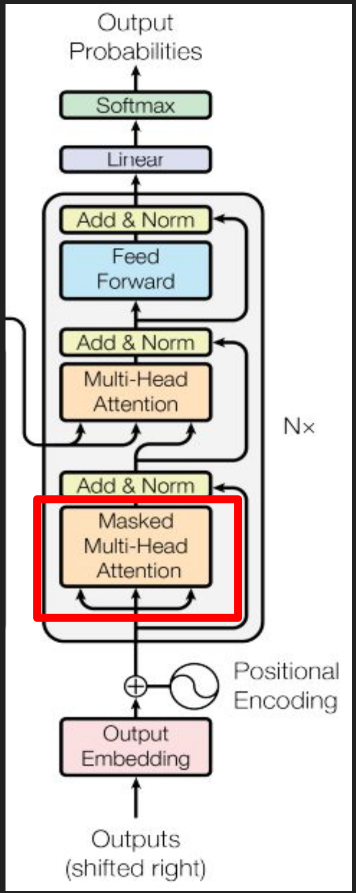


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Positional
Encoding
Output
Embedding
Outputs
(shifted right)

Decoder: Cross Attention



Decoder: Masked Multi-Head Attention



Masked Multi-Head Attention

Training

[English] I love hiking.

[Italian] Adoro le escursioni.

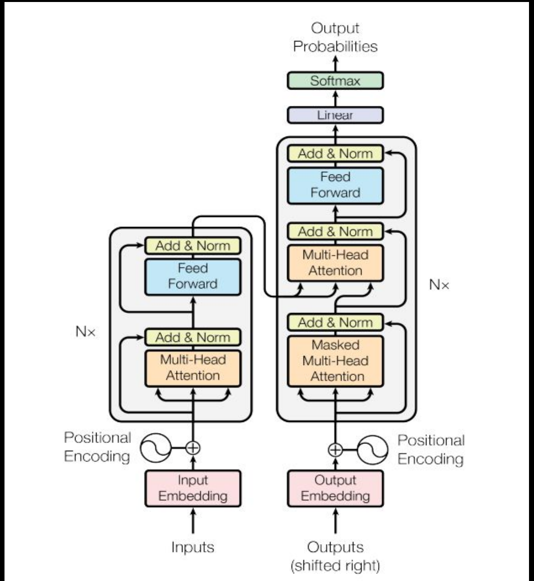


Figure 1: The Transformer - model architecture.

Training

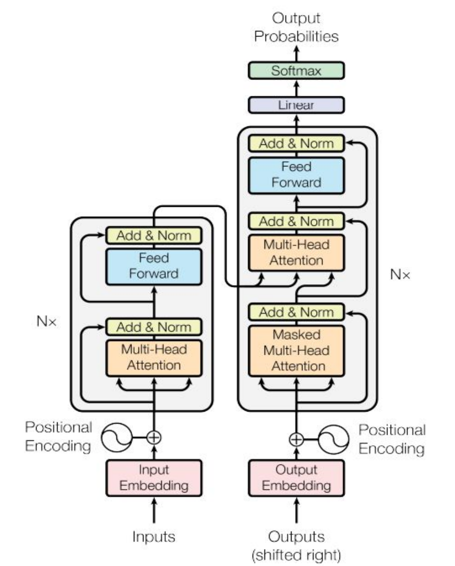
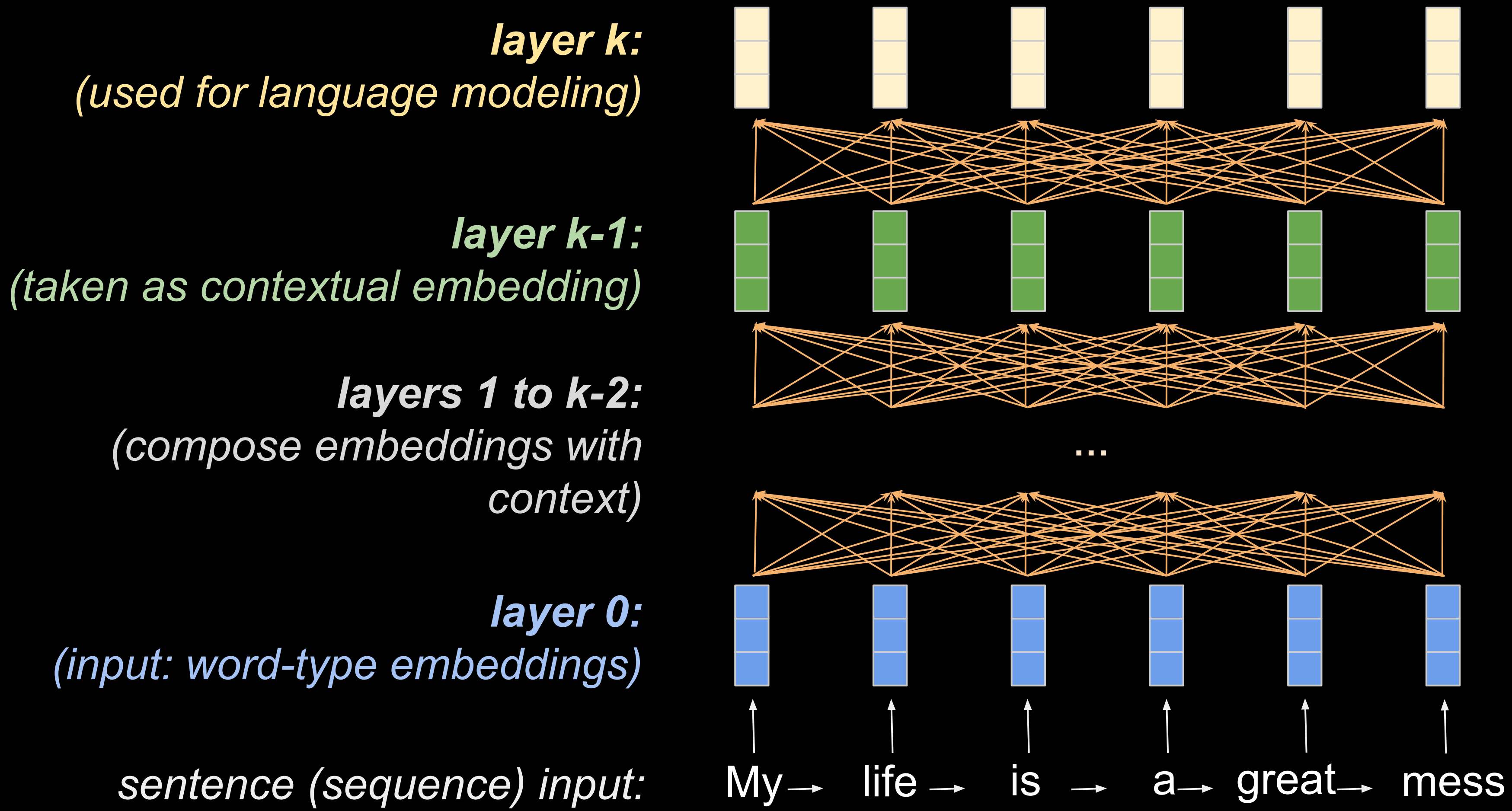


Figure 1: The Transformer - model architecture.

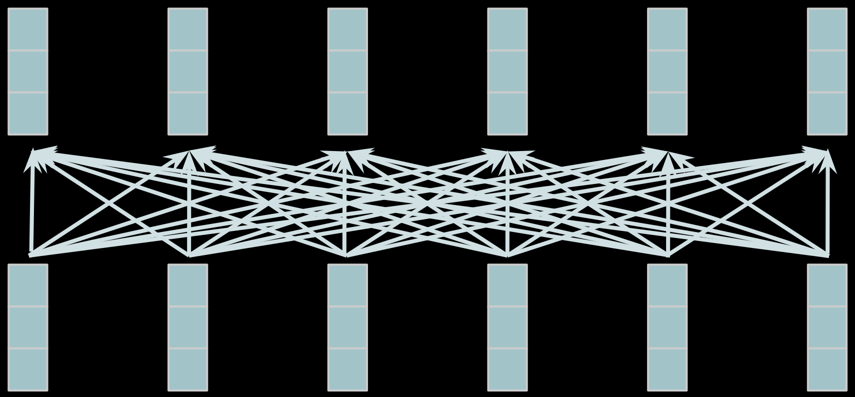
Transformer Language Models: Uses multiple layers of a **transformer**



(Kjell, Kjell, and Schwartz, 2023)

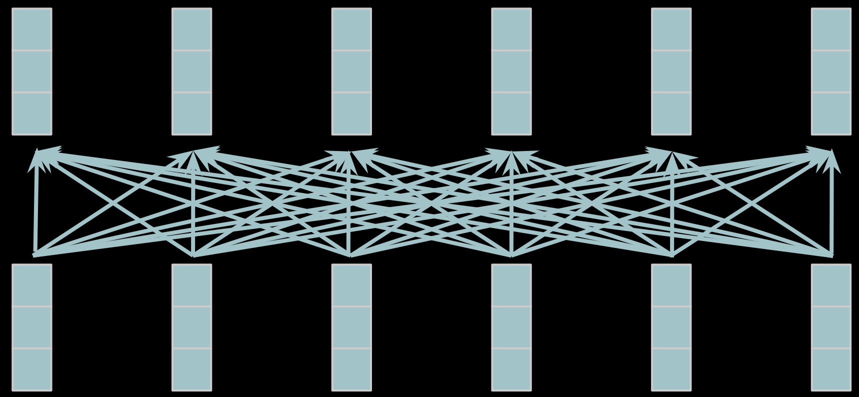
Auto-encoder (MLM):

- Connections go both directions.
- Task is predict word in middle: p(wi| …, pwi-2, wi-1, wi+1, wi+2…)
- Better for:
    - embeddings
    - fine-tuning (transfer learning)



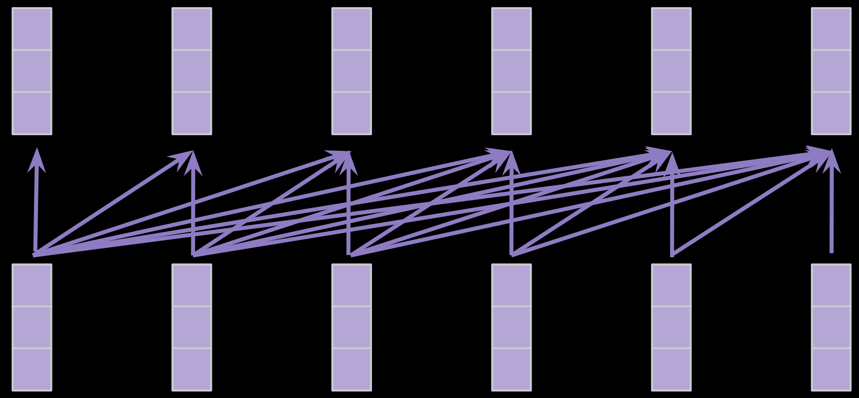
Auto-encoder (MLM):

- Connections go both directions.
- Task is predict word in middle:
  p(wi| …, pwi-2, wi-1, wi+1, wi+2…)
- Better for:
  - embeddings
  - fine-tuning (transfer learning)



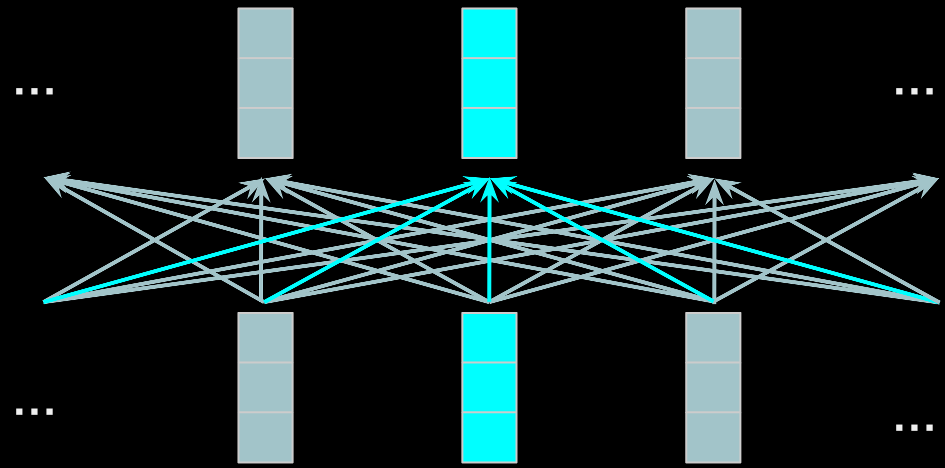
Auto-regressor *(generator):*

- Connections go forward only
- Task is predict word next word:
  p(wi| wi-1, wi-2, …)
- Better for:
  - generating text
  - zero-shot learning



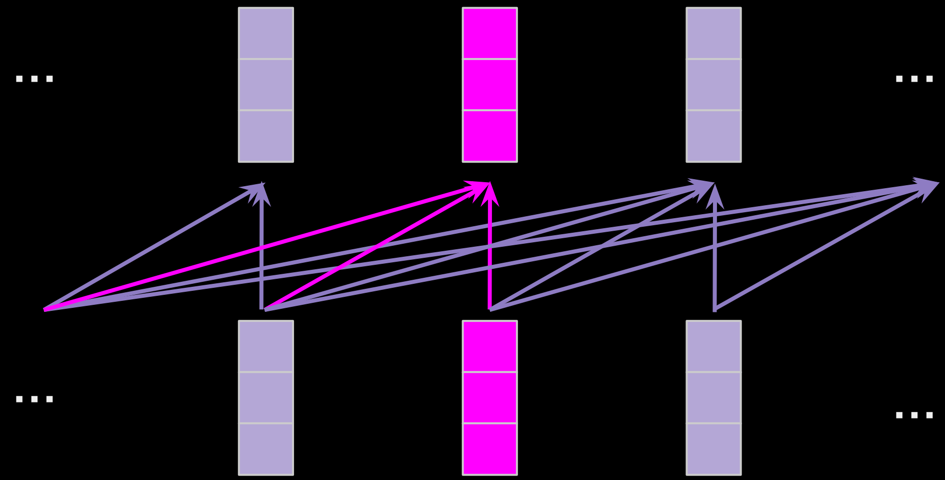
Auto-encoder (MLM):

- Connections go both directions.
- Task is predict word in middle:
  p(wi| …, pwi-2, wi-1, wi+1, wi+2…)
- Better for:
  - embeddings
  - fine-tuning (transfer learning)

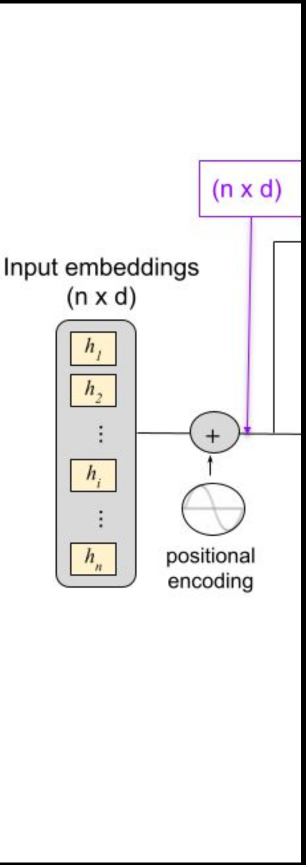


***Auto-regressor** (generator):*

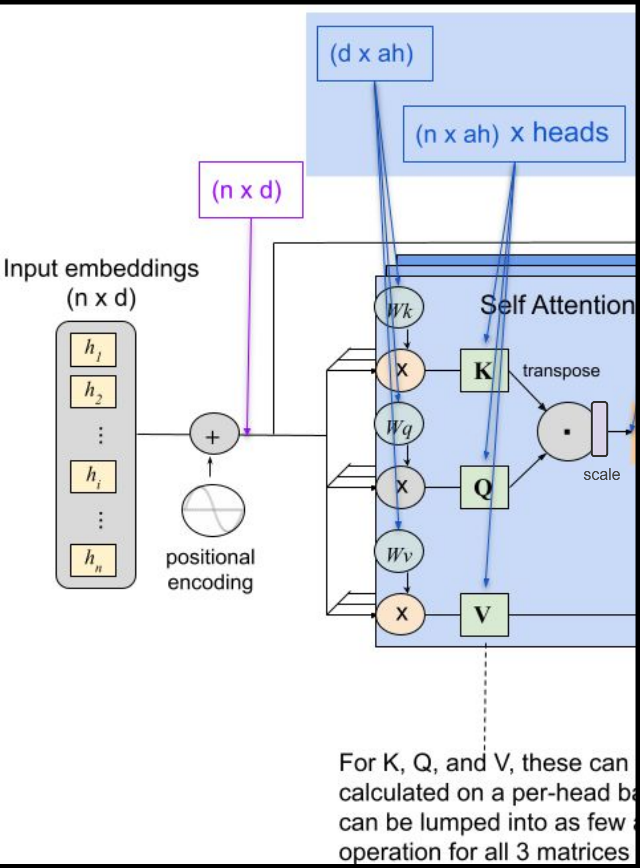
- Connections go forward only
- Task is predict word next word:
  p(wi| wi-1, wi-2, …)
- Better for:
  - generating text
  - zero-shot learning



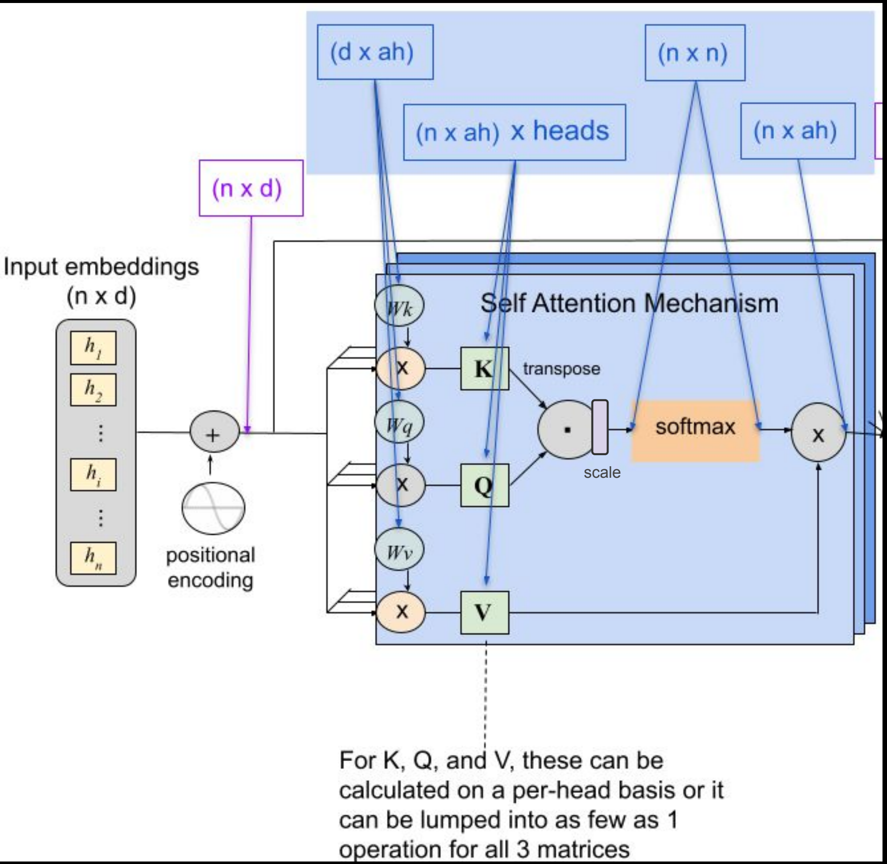
Detailed Overview of (HuggingFace) Transformer Matrices and Computation



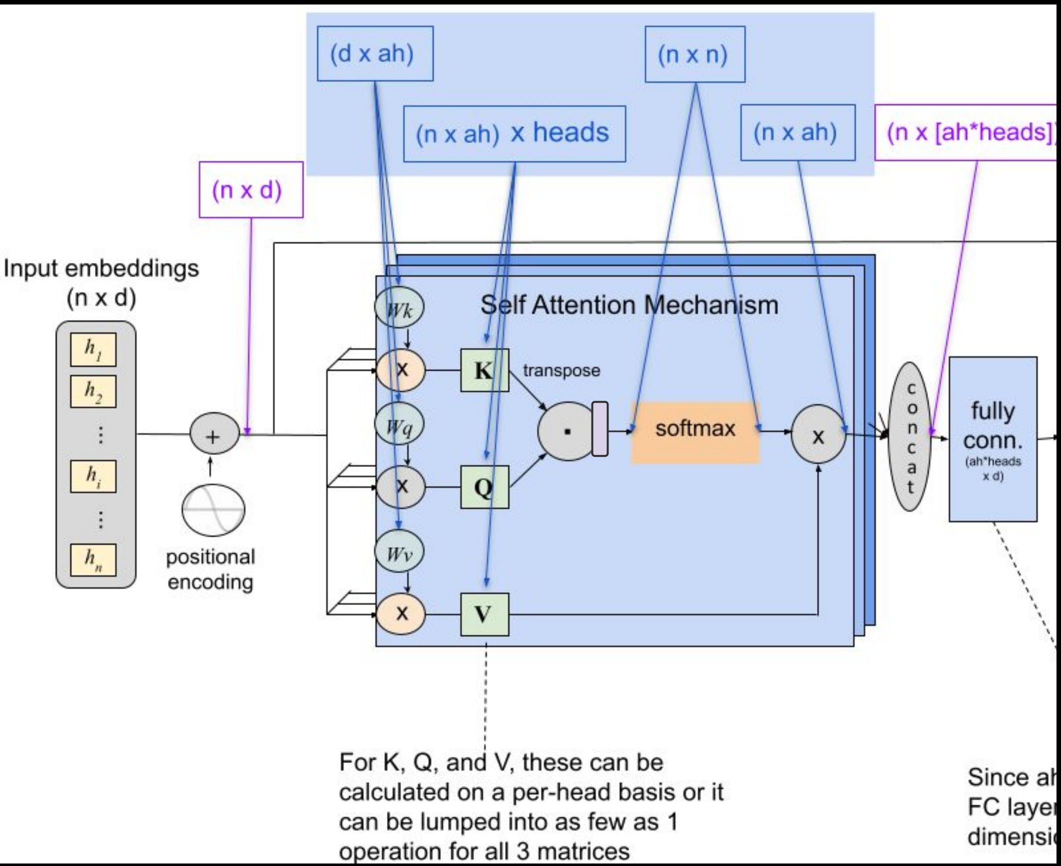
Detailed Overview of (HuggingFace) Transformer Matrices and Computation



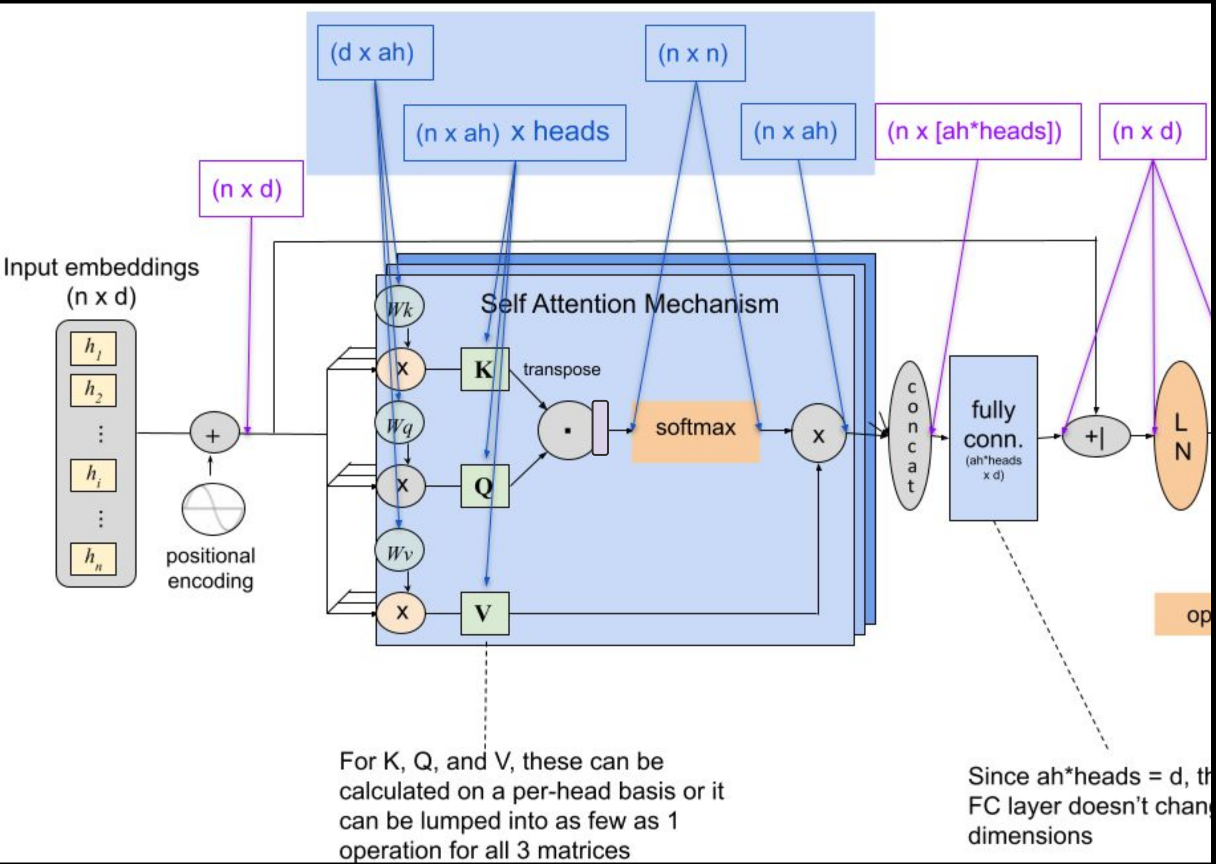
Detailed Overview of (HuggingFace) Transformer Matrices and Computation



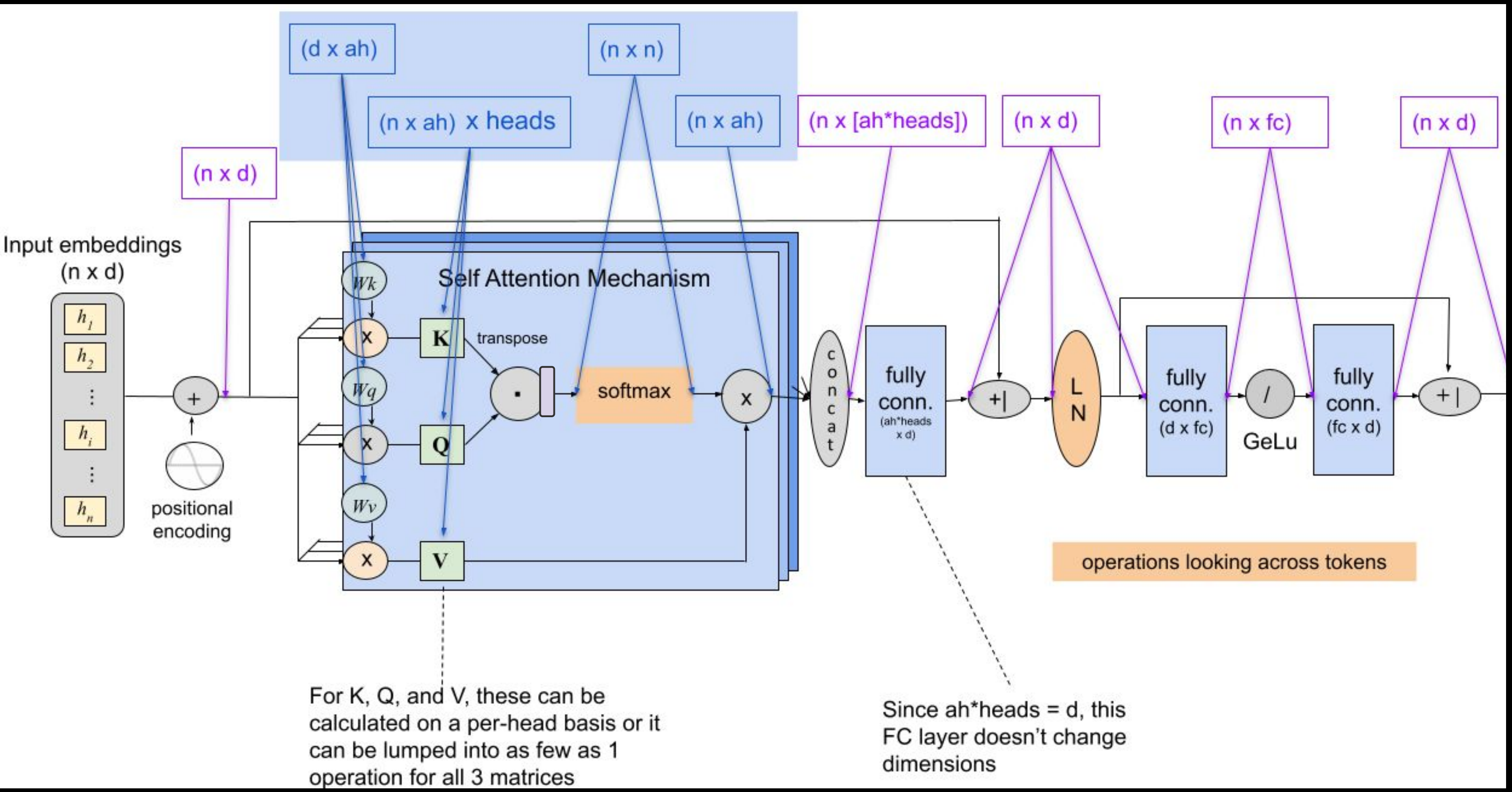
Detailed Overview of (HuggingFace) Transformer Matrices and Computation



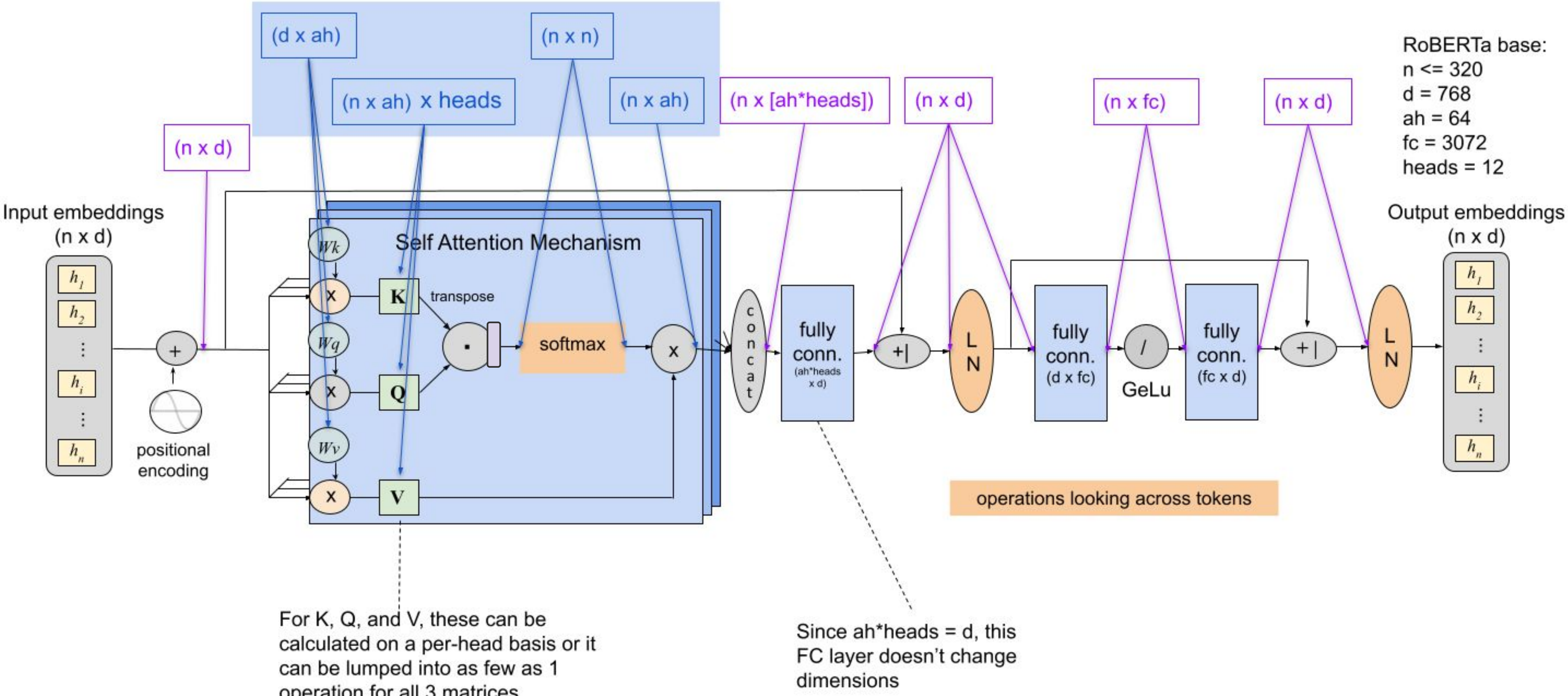
Detailed Overview of (HuggingFace) Transformer Matrices and Computation



Detailed Overview of (HuggingFace) Transformer Matrices and Computation



Detailed Overview of (HuggingFace) Transformer Matrices and Computation



Hugging Face or AllenNLP

https://github.com/huggingface/transformers

```
#example for getting embeddings
from transformers import BertModel, PreTrainedTokenizerFast, pipeline

bert_tokenizer = PreTrainedTokenizerFast.from_pretrained('google-bert/bert-base-uncased')
bert_model = BertModel.from_pretrained('google-bert/bert-base-uncased')
pipe = pipeline('feature-extraction', model=bert_model, tokenizer=bert_tokenizer)
emb = pipe(text)
print(emb[0][0])
```

https://docs.allennlp.org/v2.10.1/api/modules/transformer/transformer_module/

Transformer (as of 2017)

"WMT-2014" Data Set. BLEU scores:

	EN-DE	EN-FR
GNMT (orig)	24.6	39.9
ConvSeq2Seq	25.2	40.5
Transformer*	**28.4**	**41.8**

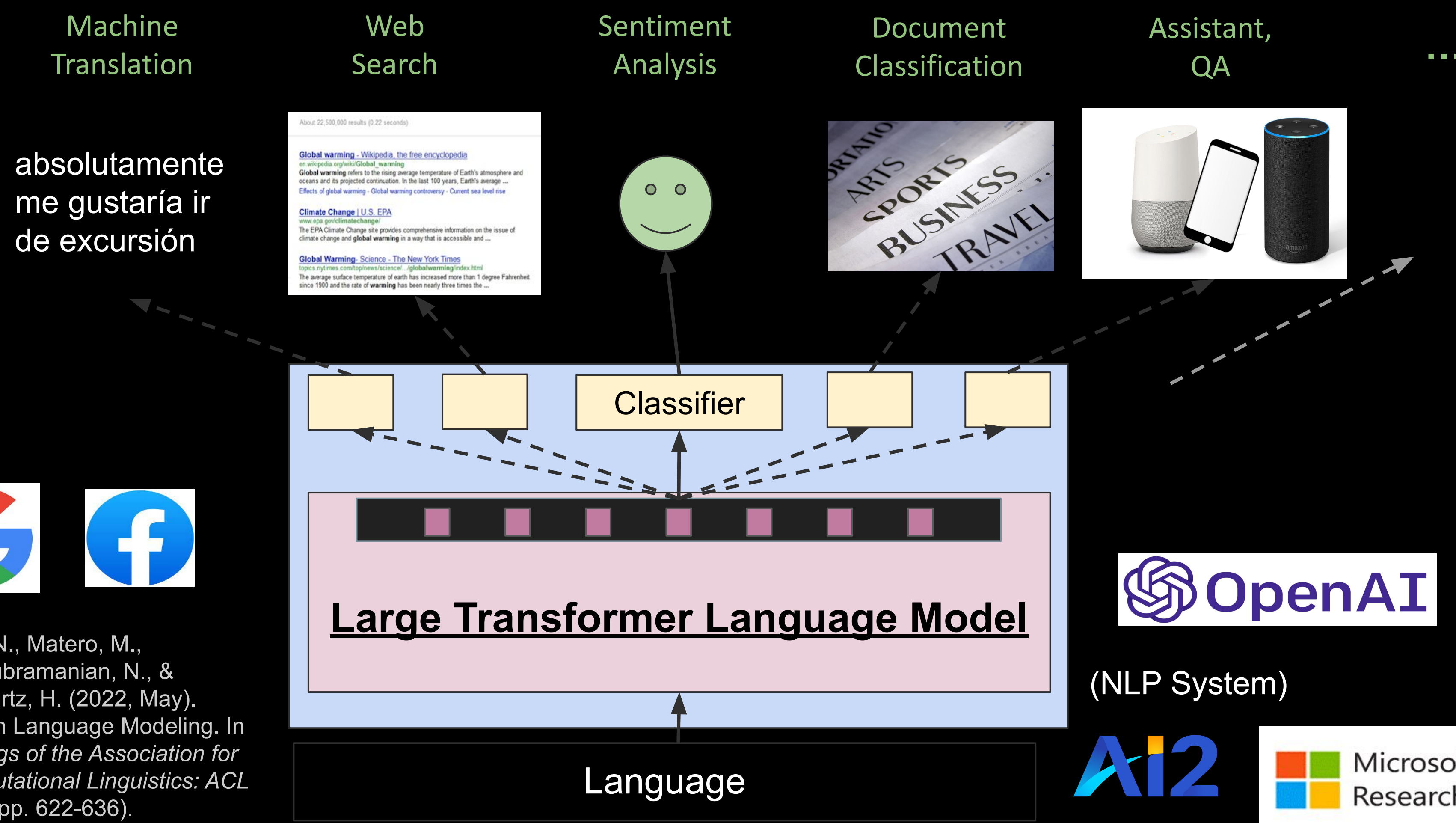
Transformers as of 2023

General Language Understanding Evaluations:

https://gluebenchmark.com/leaderboard

https://super.gluebenchmark.com/leaderboard/

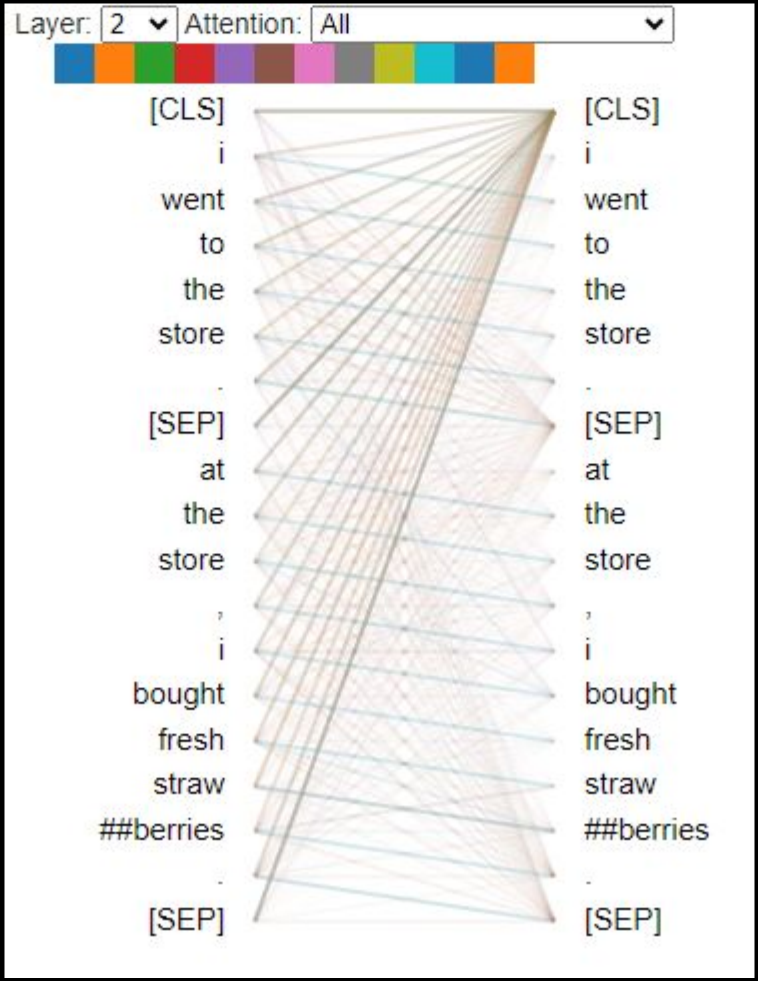
Transformers as of 2023



Soni, N., Matero, M., Balasubramanian, N., & Schwartz, H. (2022, May). Human Language Modeling. In *Findings of the Association for Computational Linguistics: ACL 2022* (pp. 622-636).

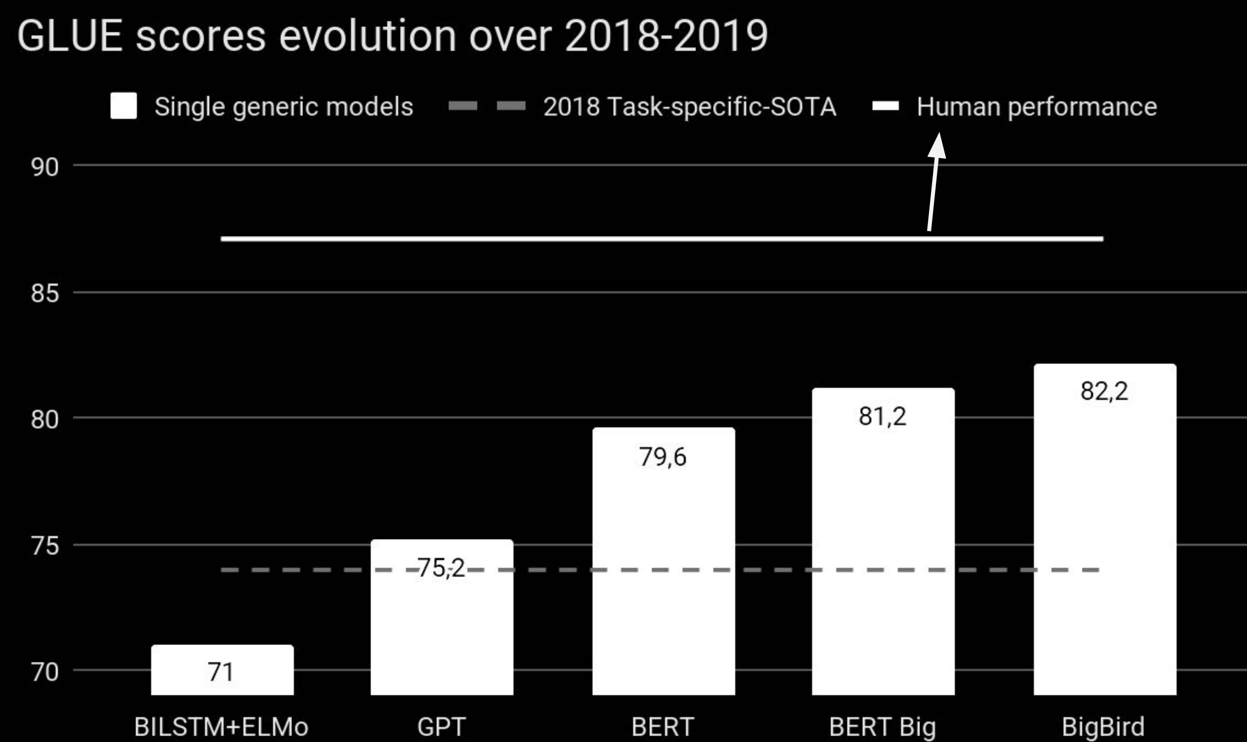
Bert: Attention by Layers

https://colab.research.google.com/drive/1vlOJ1lhdujVjfH857hvYKIdKPTD9Kid8



(Vig, 2019)

BERT Performance: e.g. Question Answering



https://rajpurkar.github.io/SQuAD-explorer/

The Transformer: Take Away

Challenges to sequential representation learning

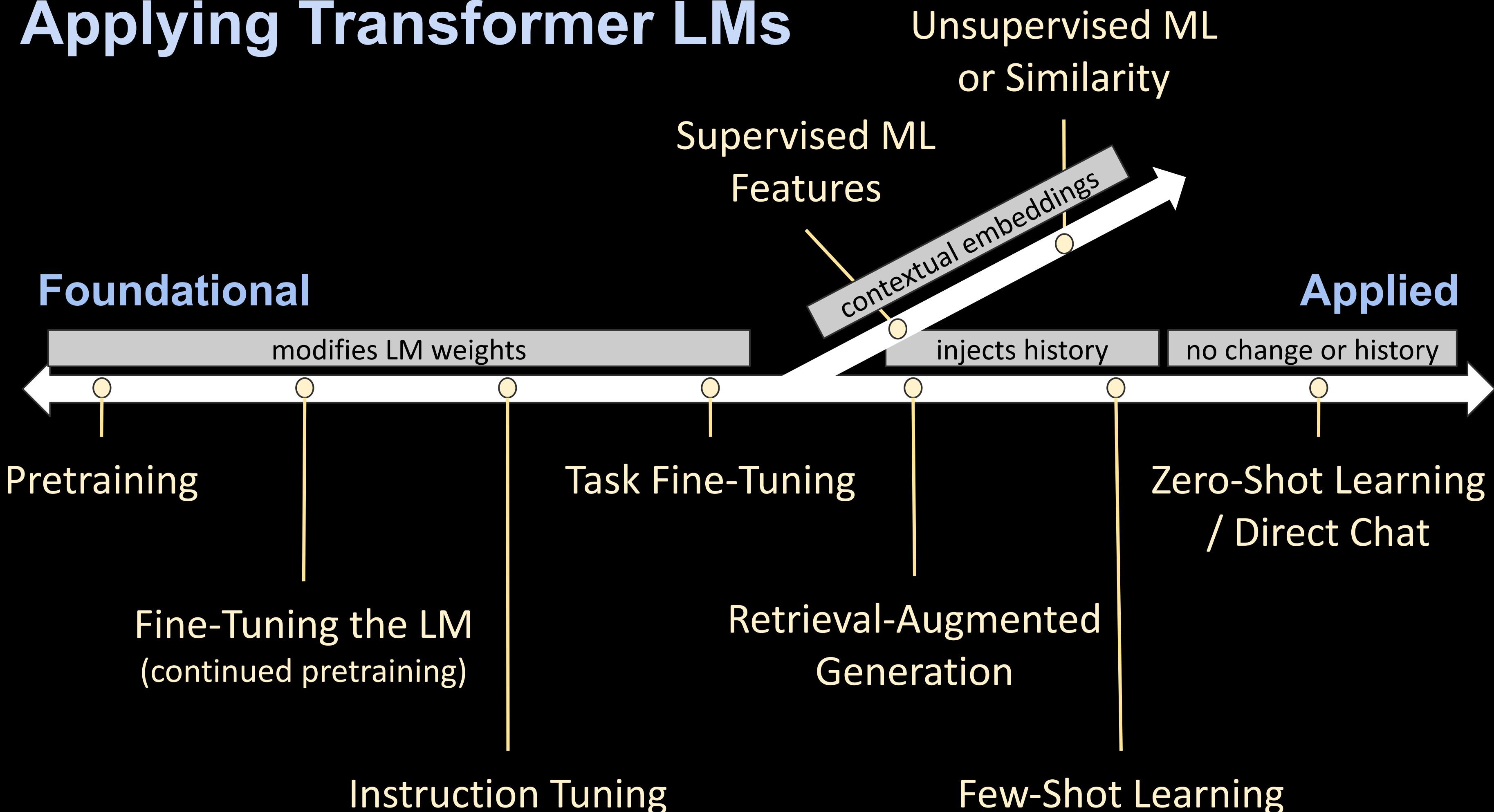
- Capture long-distance dependencies
  Self-attention *treats far away words similar to those close.*
- Preserving sequential distances / periodicity
  Positional embeddings *encode distances/periods.*
- Capture multiple relationships
  Multi-headed attention *enables multiple compositions.*
- Easy to parallelize -- don't need sequential processing.
  Entire layer can be computed at once. Is only matrix multiplications + standardizing.

Part 3: Applying Transformer LMs

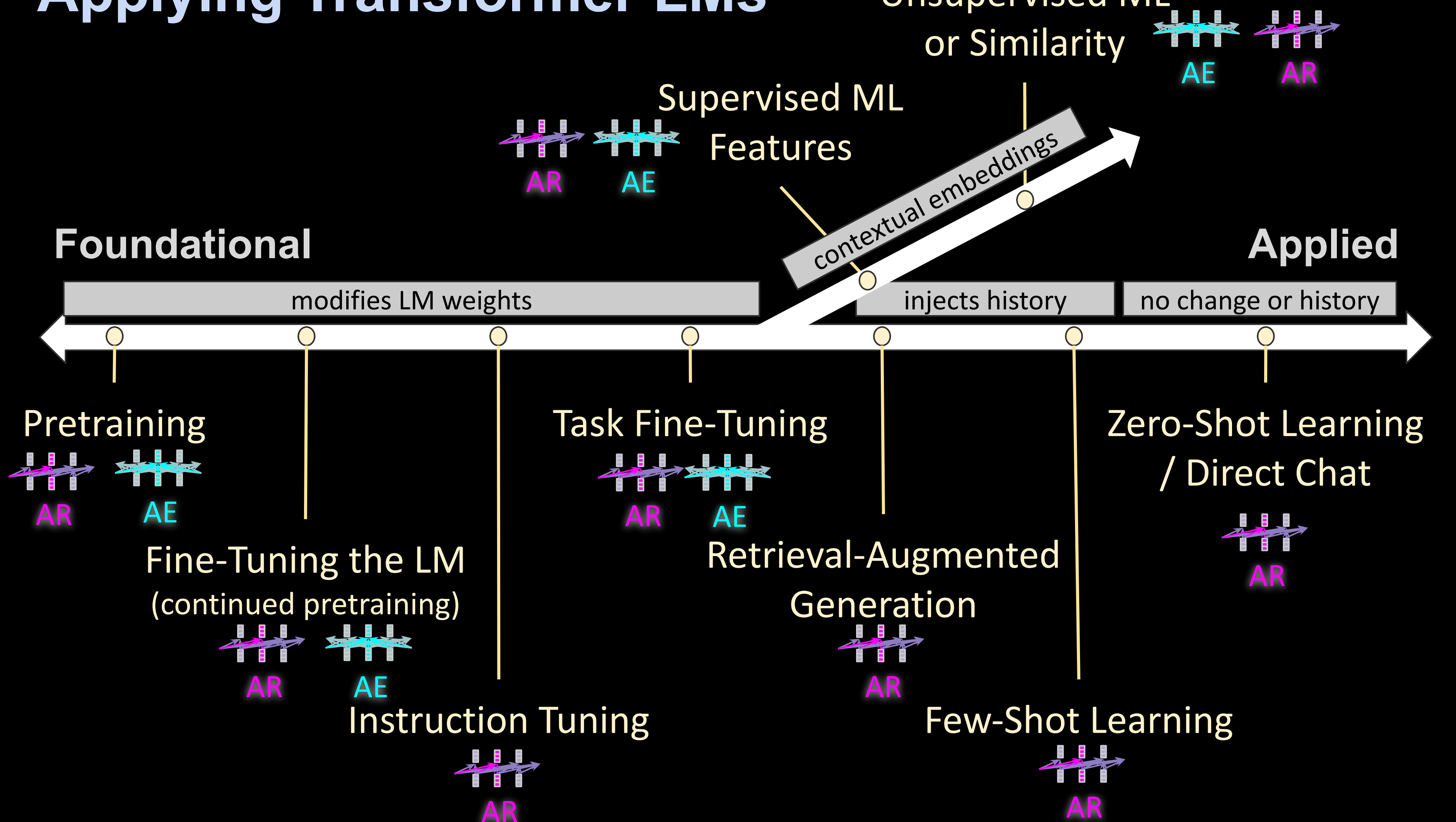
Foundational

Applied

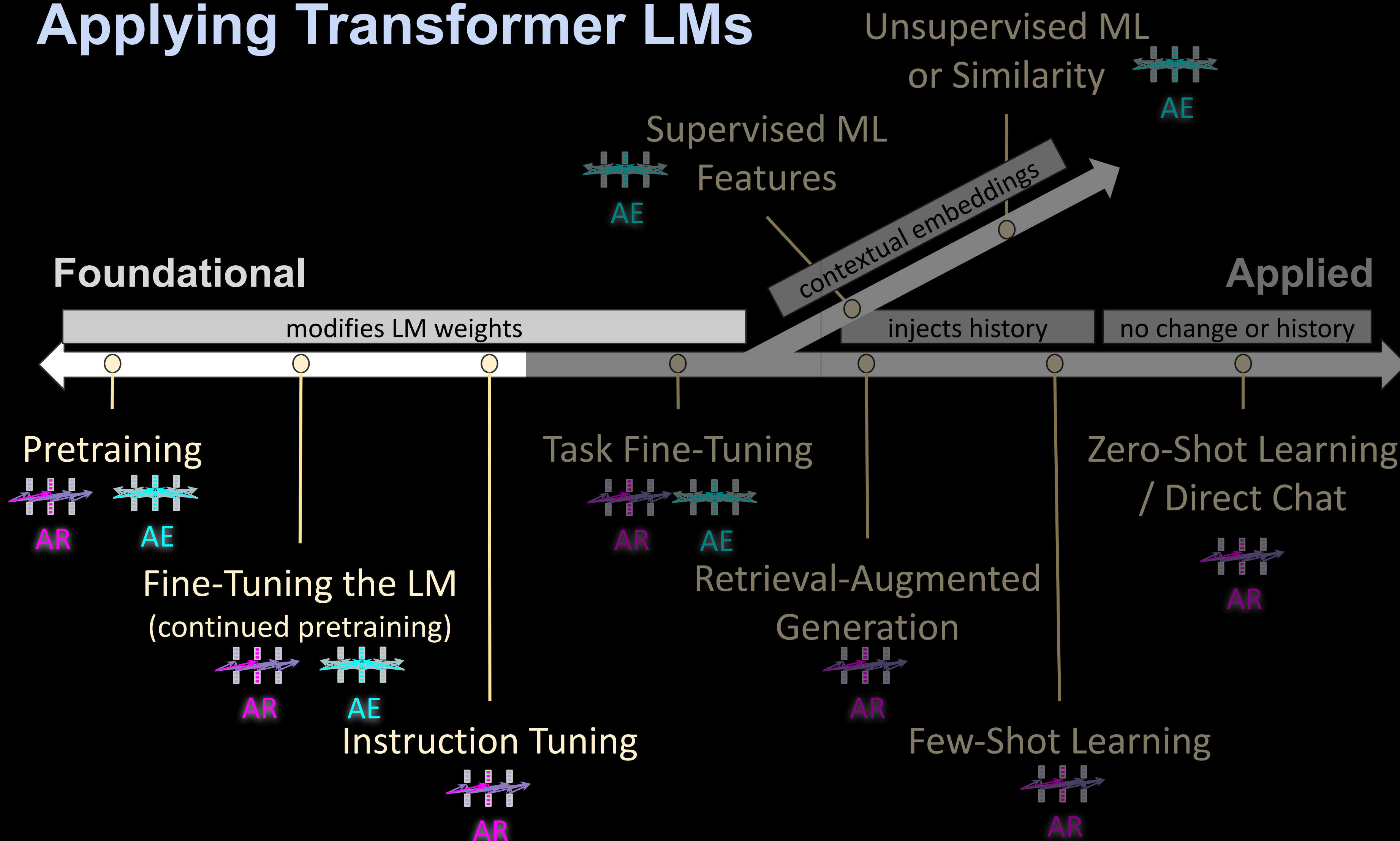
Applying Transformer LMs



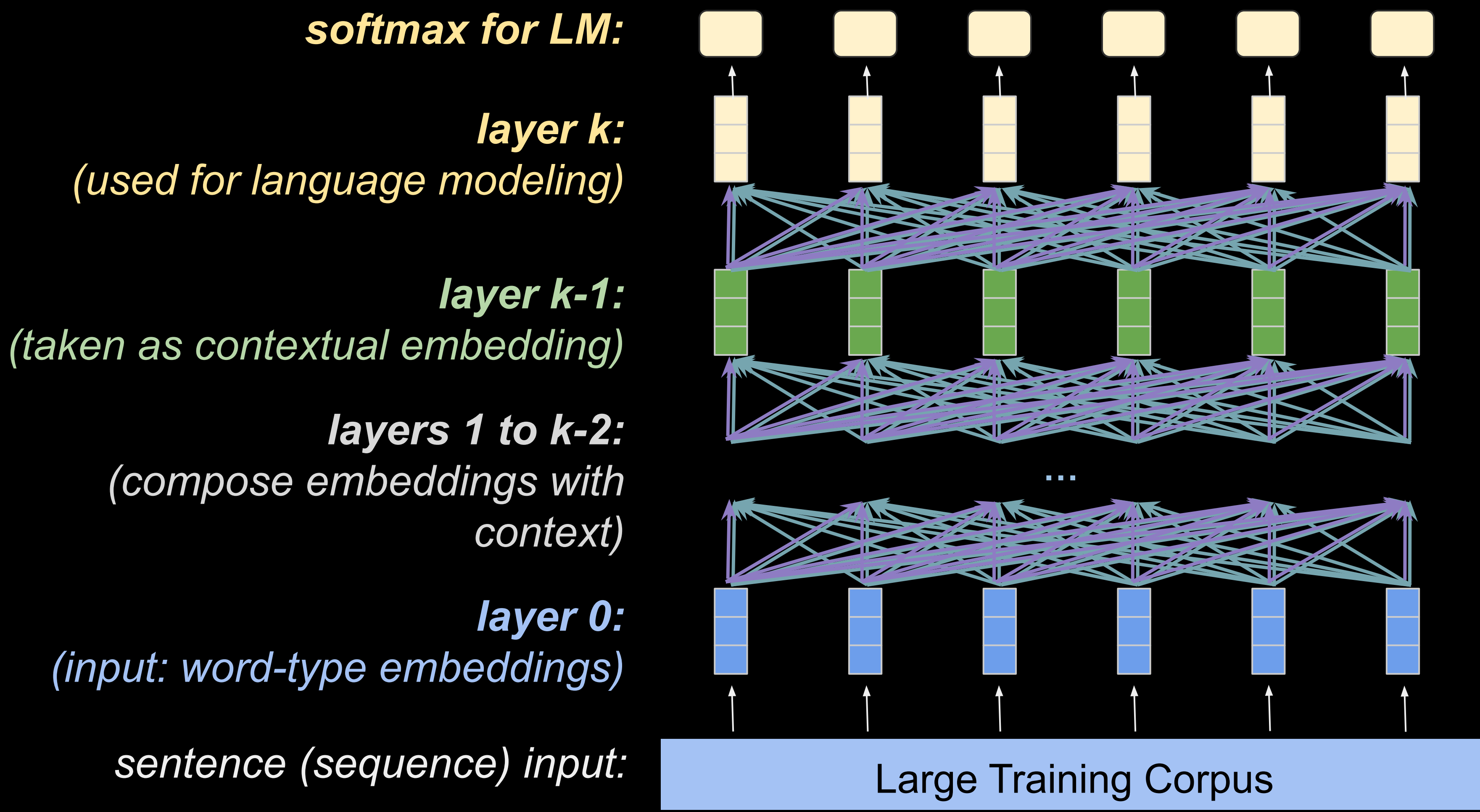
Applying Transformer LMs



Applying Transformer LMs

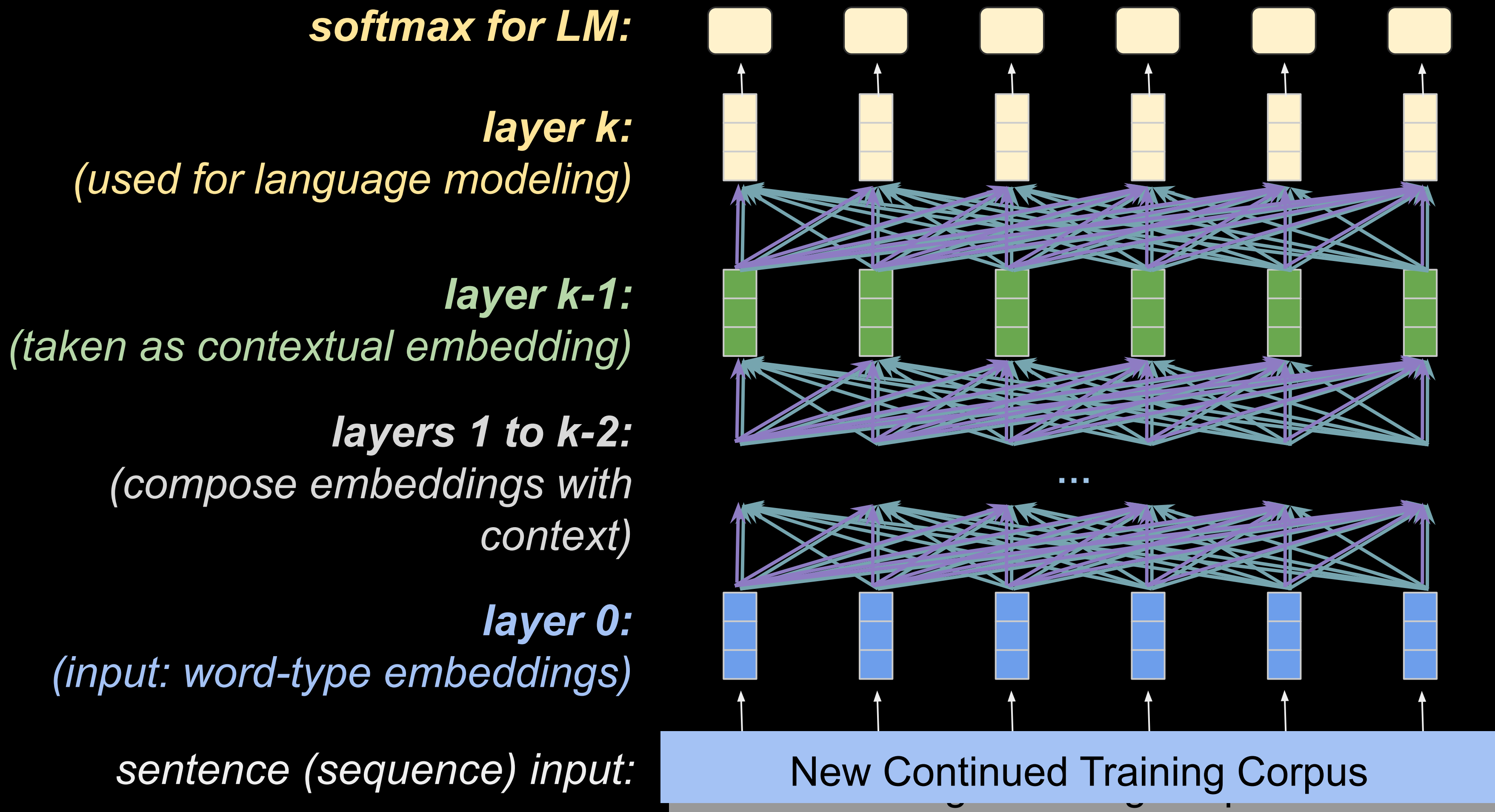


Pretraining; FTing the LM; Instruction Tuning

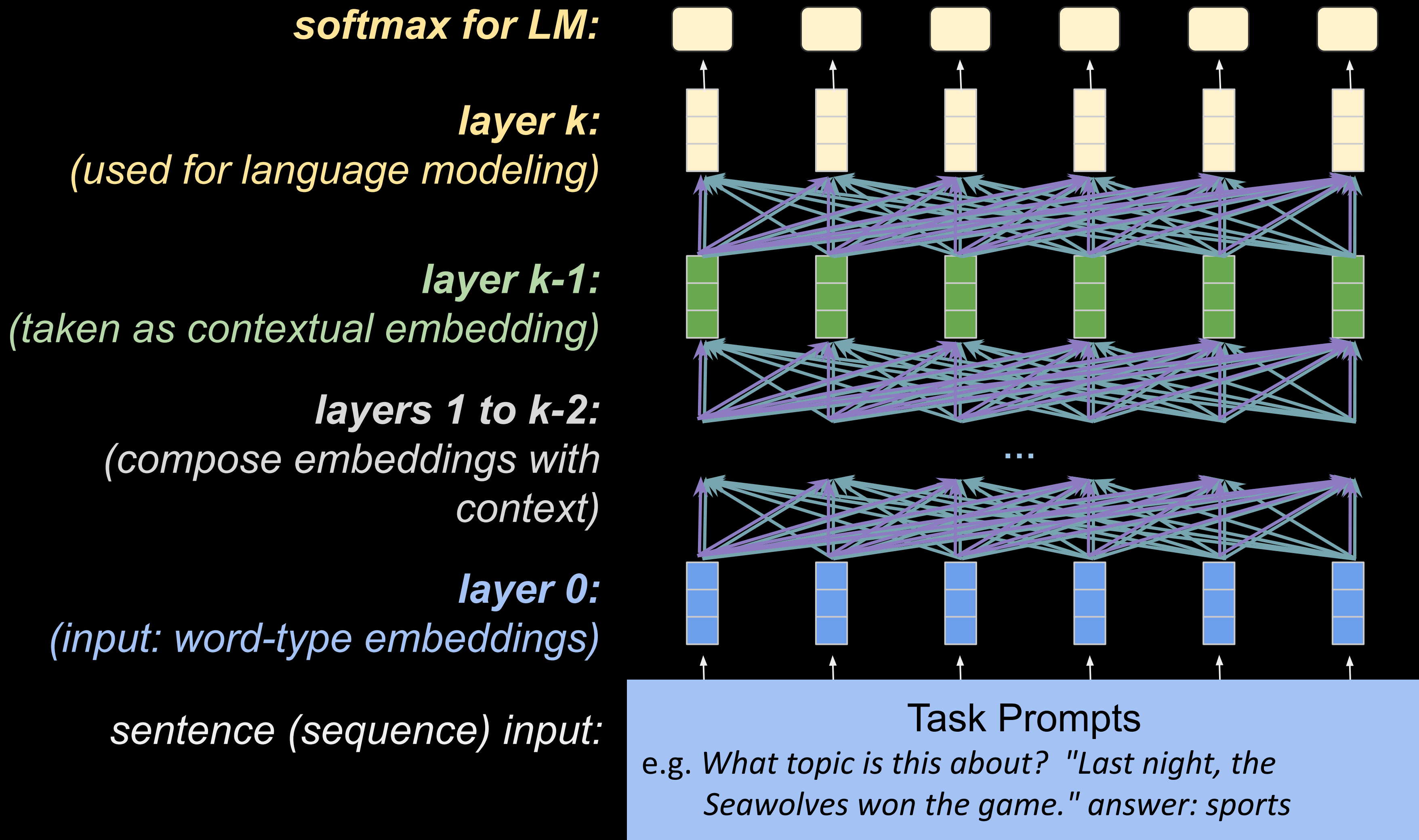


(Kjell, Kjell, and Schwartz, 2023)

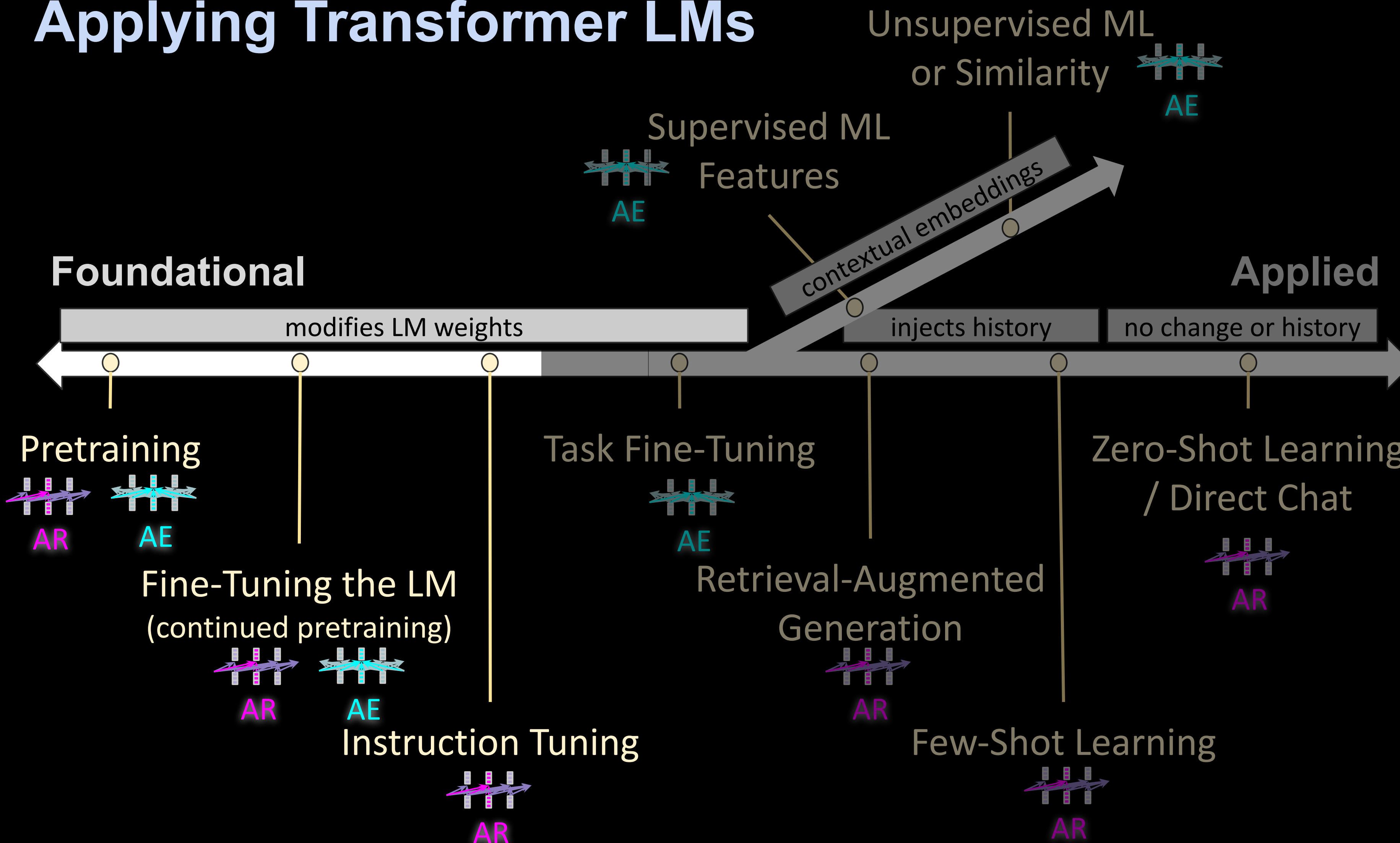
Pretraining; **FTing the LM**; Instruction Tuning



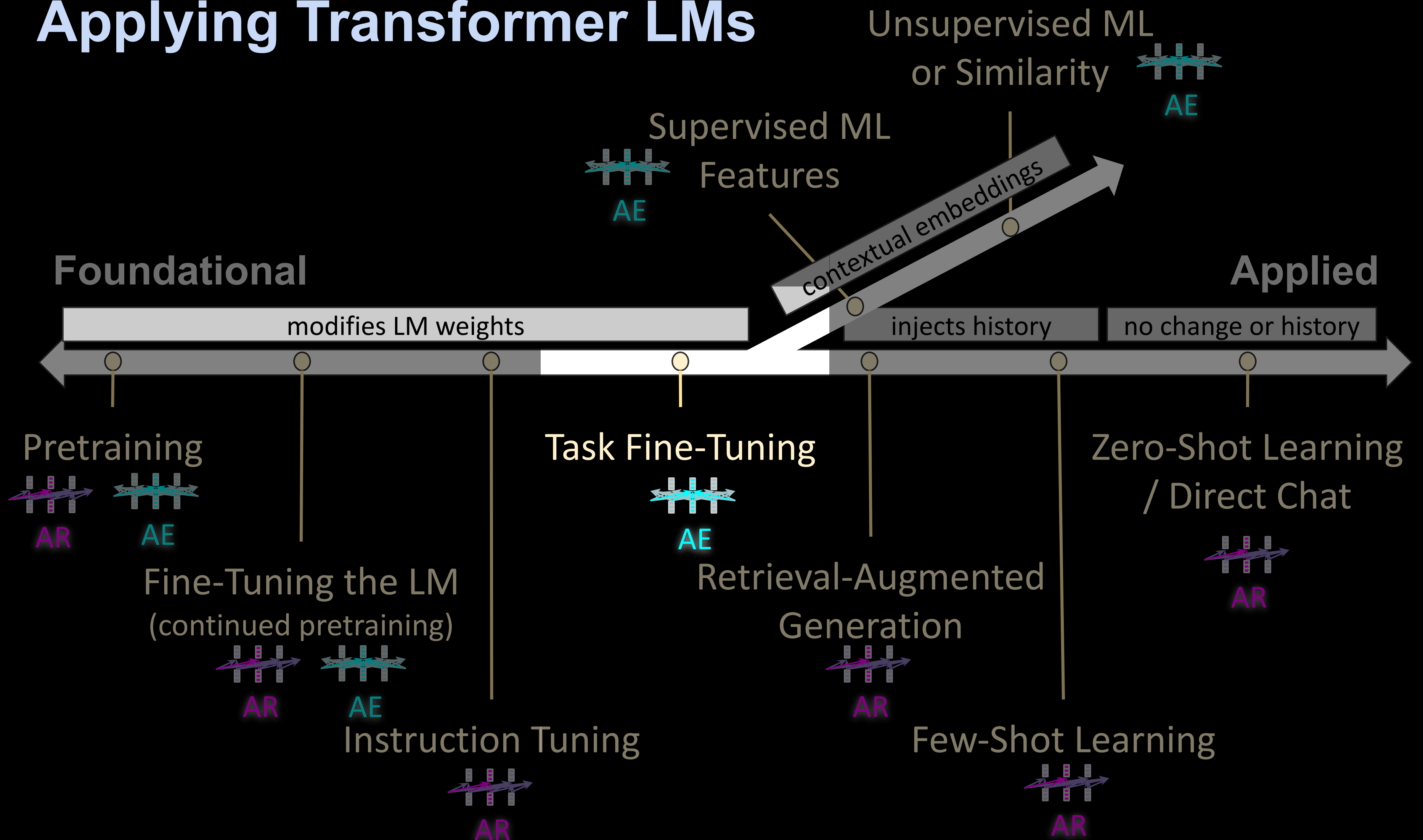
Pretraining; FTing the LM; **Instruction Tuning**



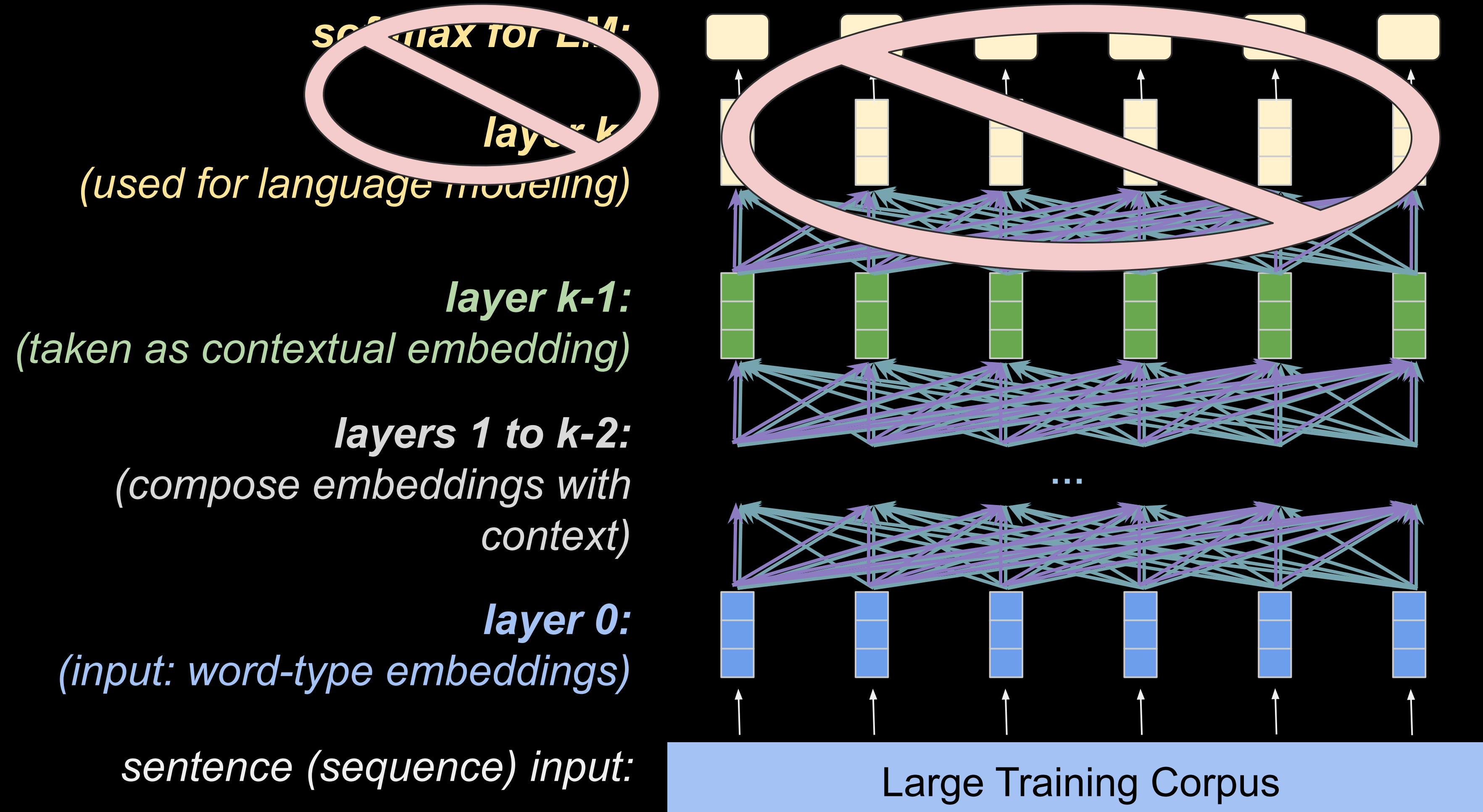
Applying Transformer LMs



Applying Transformer LMs

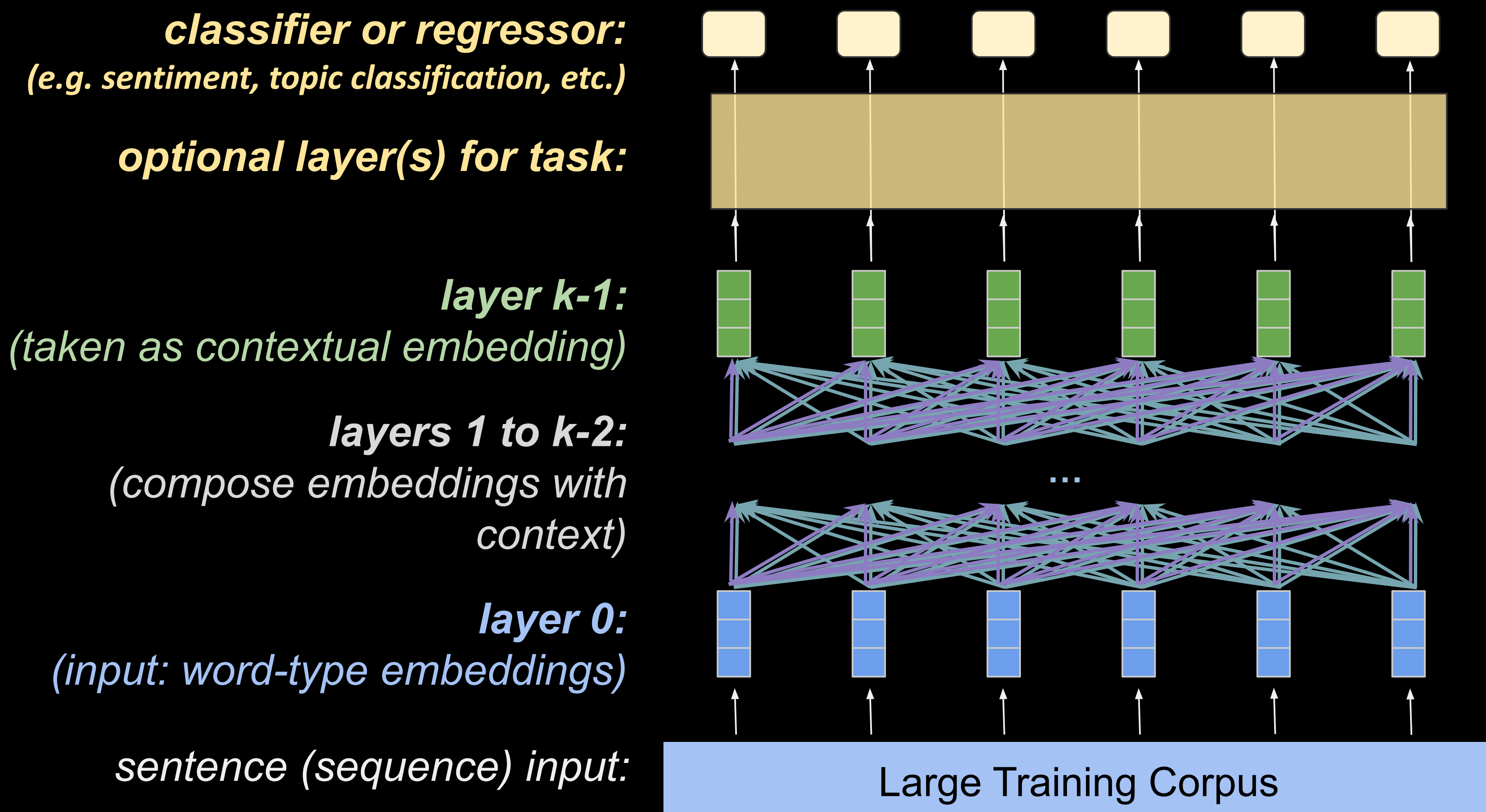


Task Fine-Tuning



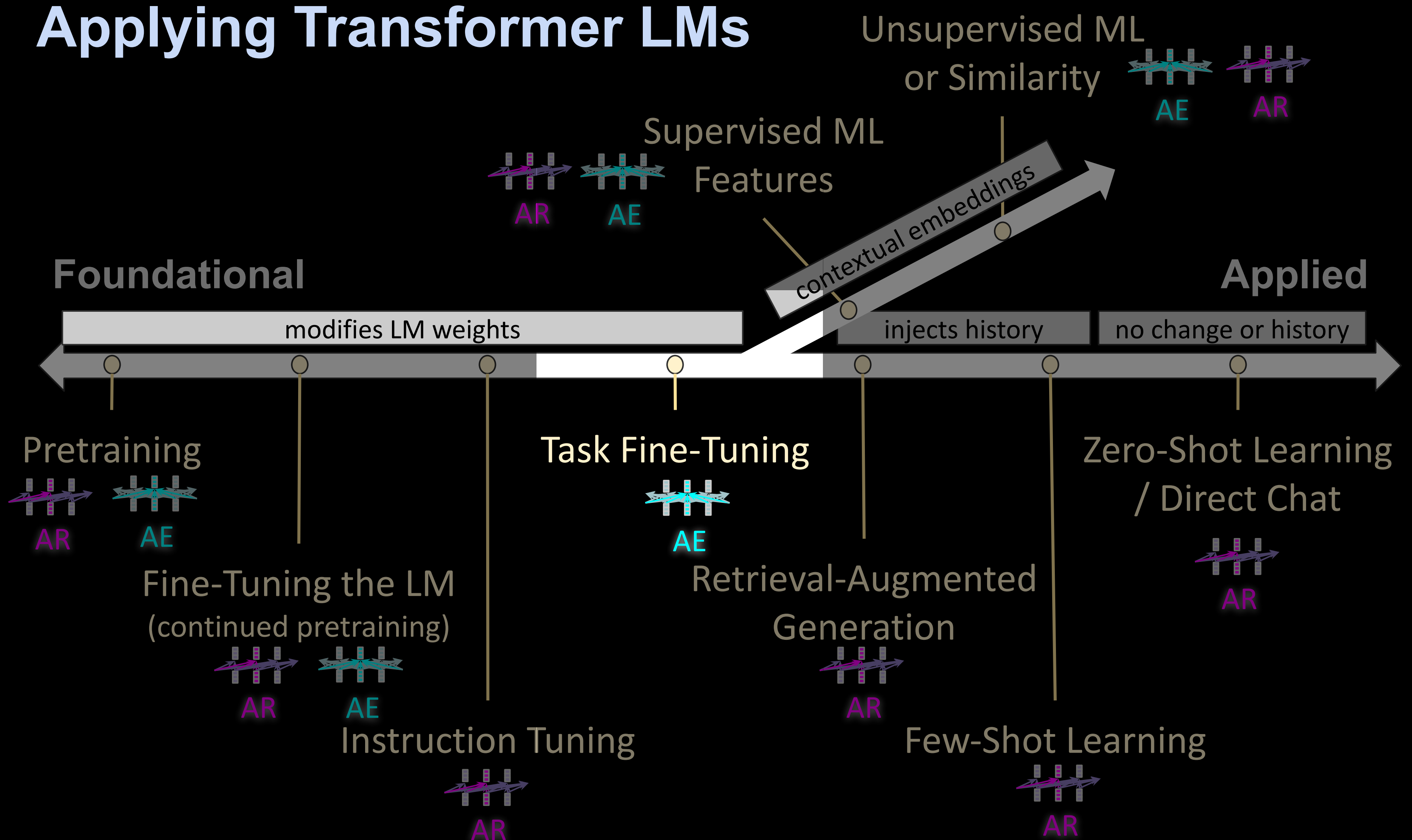
(Kjell, Kjell, and Schwartz, 2023)

Task Fine-Tuning

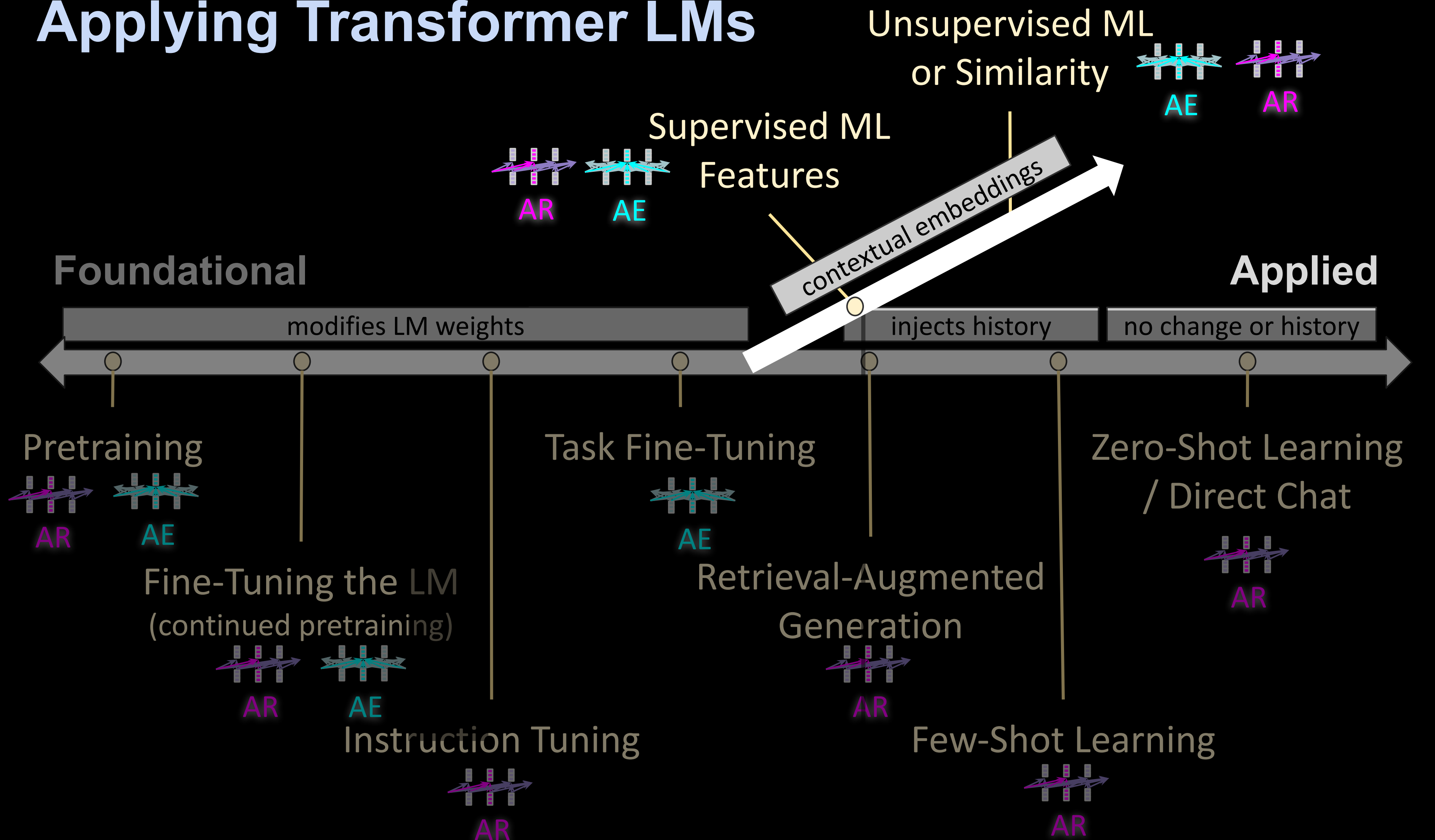


(Kjell, Kjell, and Schwartz, 2023)

Applying Transformer LMs



Applying Transformer LMs



Contextual Embeddings: for Supervised ML; for Similarity (unsup)

softmax for LM:

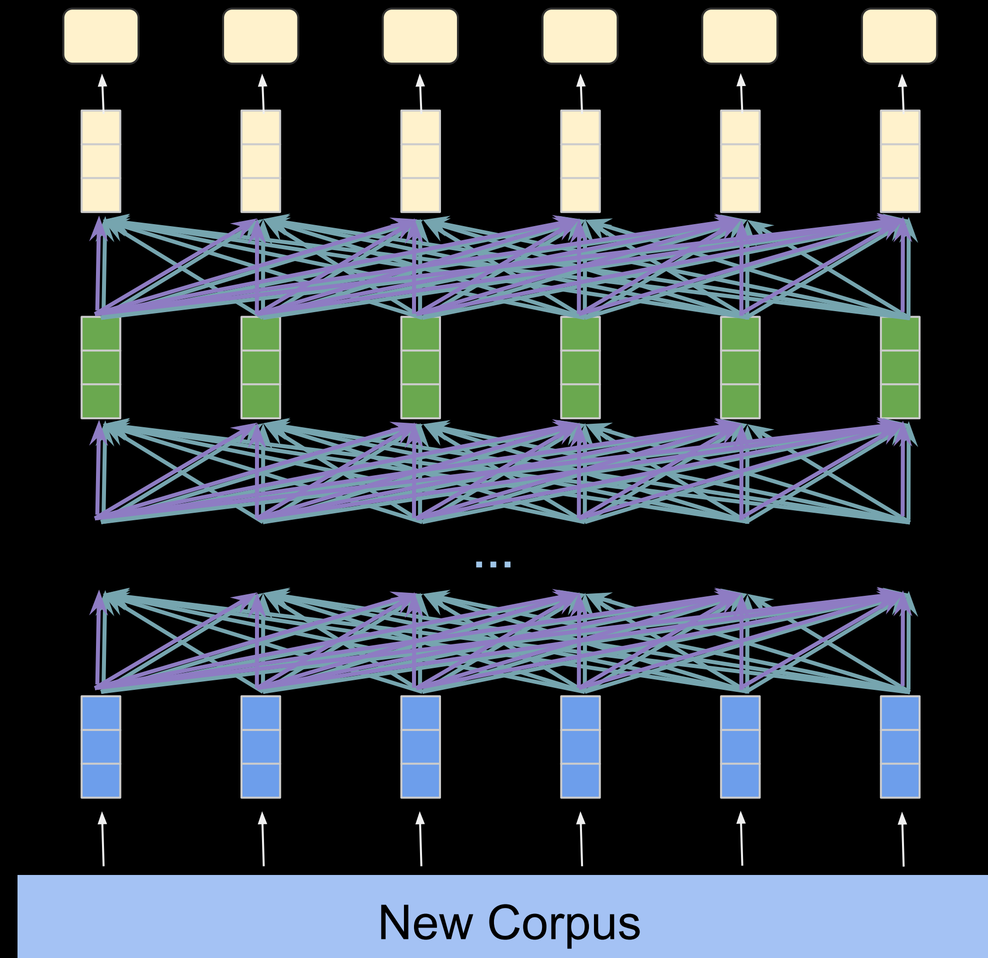
layer k:
(used for language modeling)

layer k-1:
(taken as contextual embedding)

layers 1 to k-2:
(compose embeddings with context)

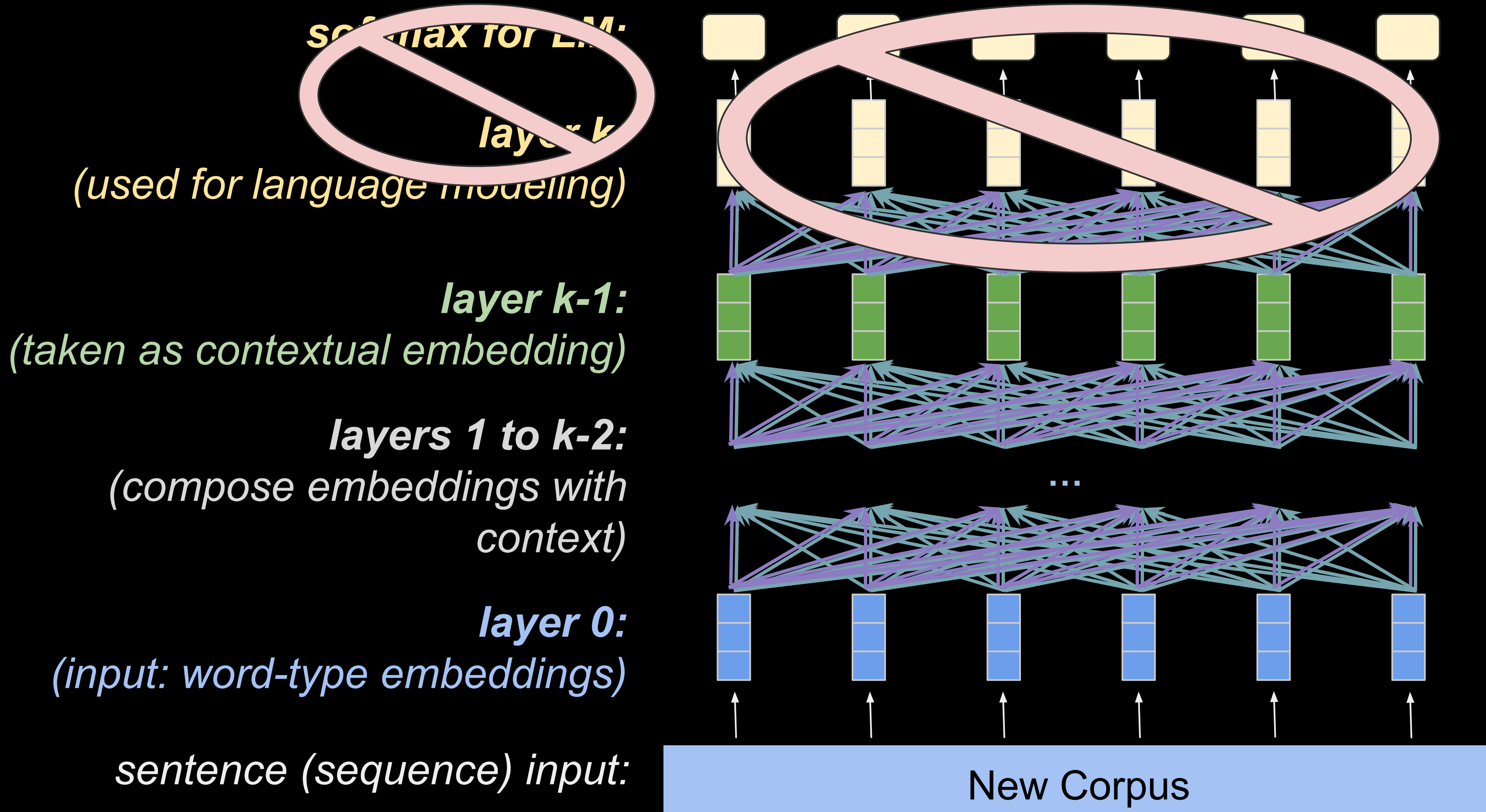
layer 0:
(input: word-type embeddings)

sentence (sequence) input:



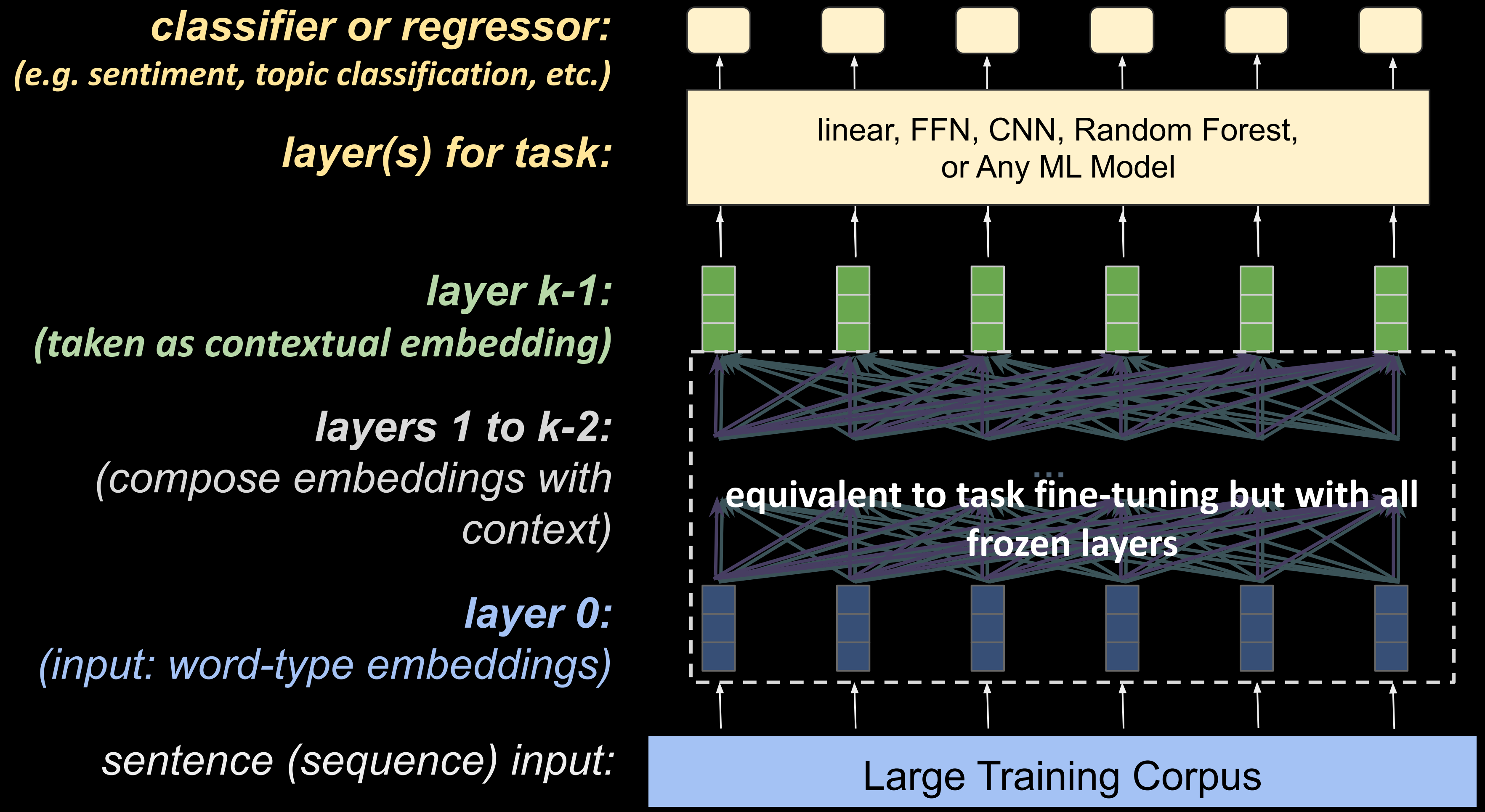
(Kjell, Kjell, and Schwartz, 2023)

Contextual Embeddings: for Supervised ML; for Similarity (unsup)



(Kjell, Kjell, and Schwartz, 2023)

Contextual Embeddings: for Supervised ML



(Kjell, Kjell, and Schwartz, 2023)

Contextual Embeddings: for Similarity (unsup)

classifier or regressor:
(e.g. sentiment, topic classification, etc.)

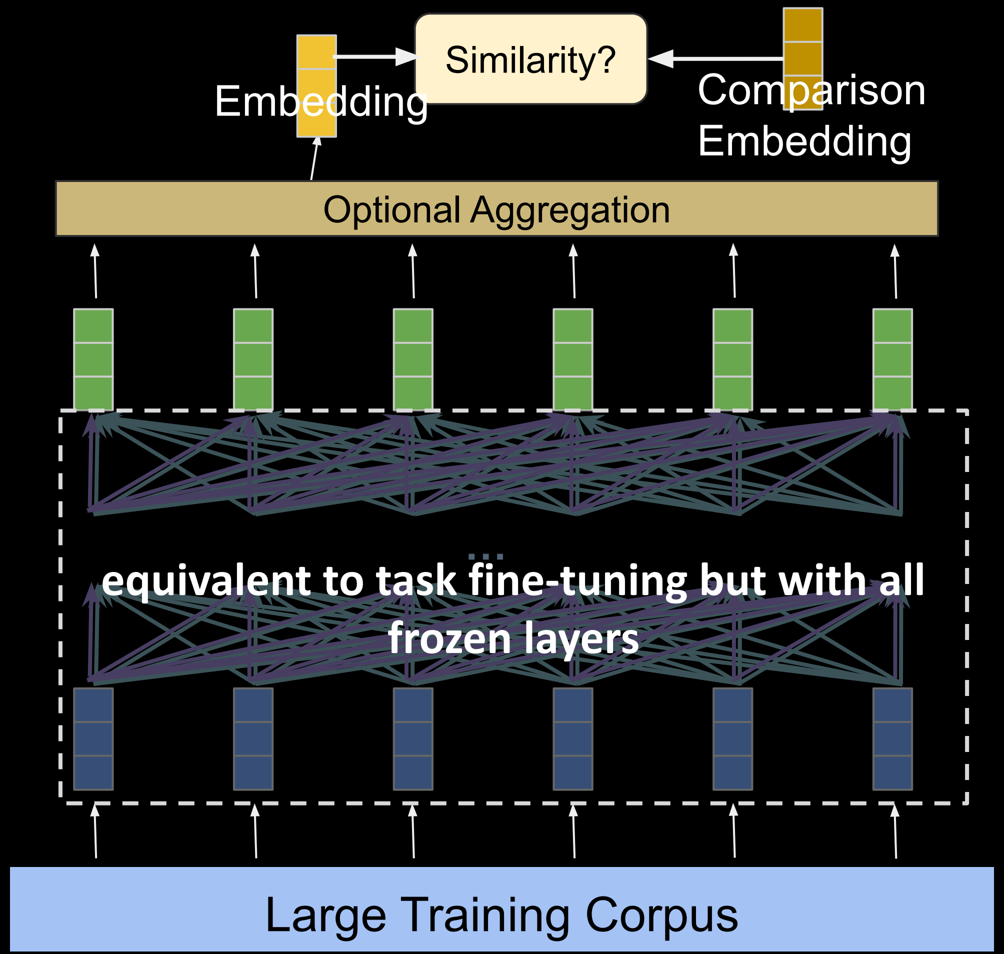
layer(s) for task:

layer k-1:
(taken as contextual embedding)

layers 1 to k-2:
(compose embeddings with context)

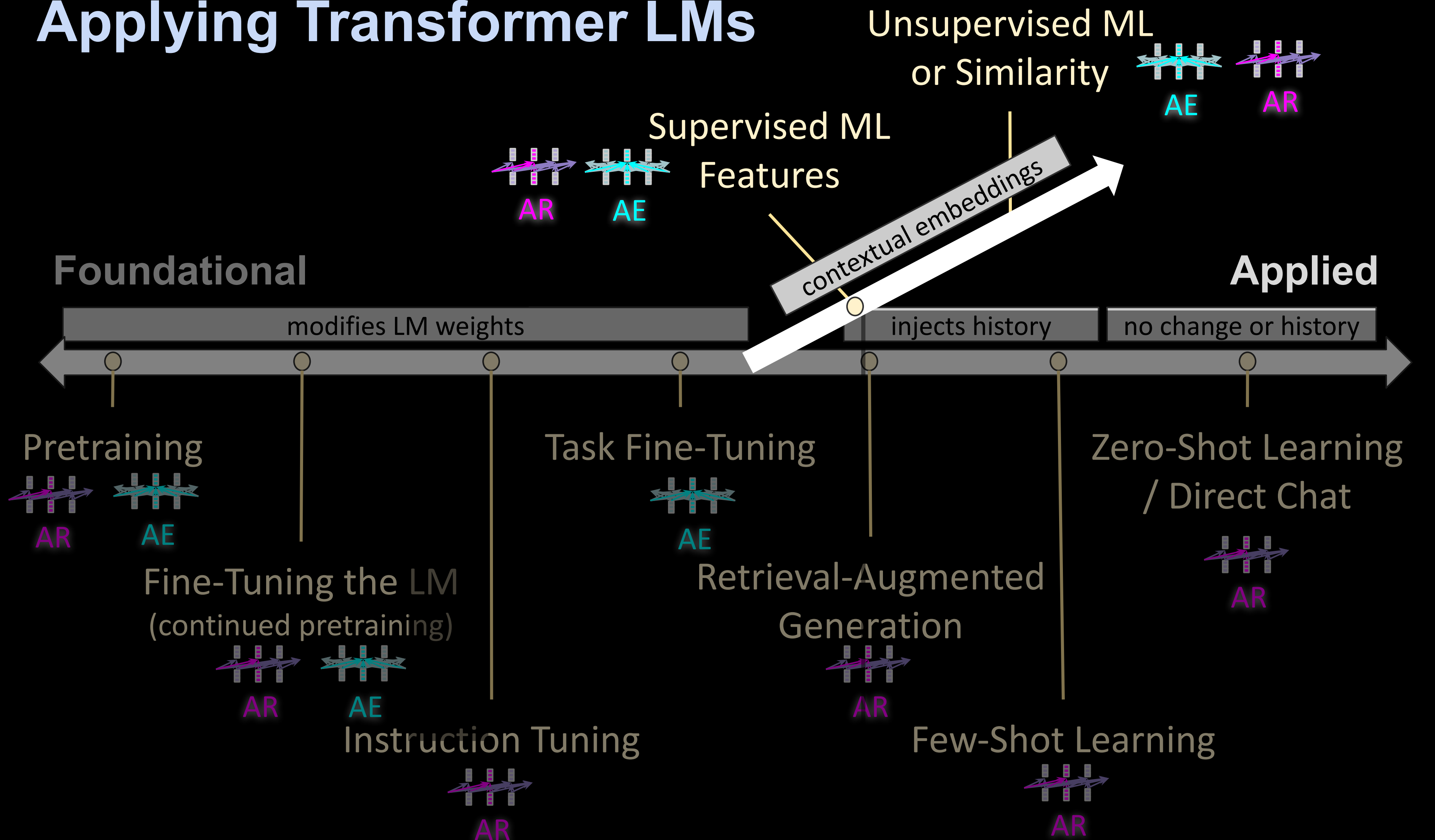
layer 0:
(input: word-type embeddings)

sentence (sequence) input:

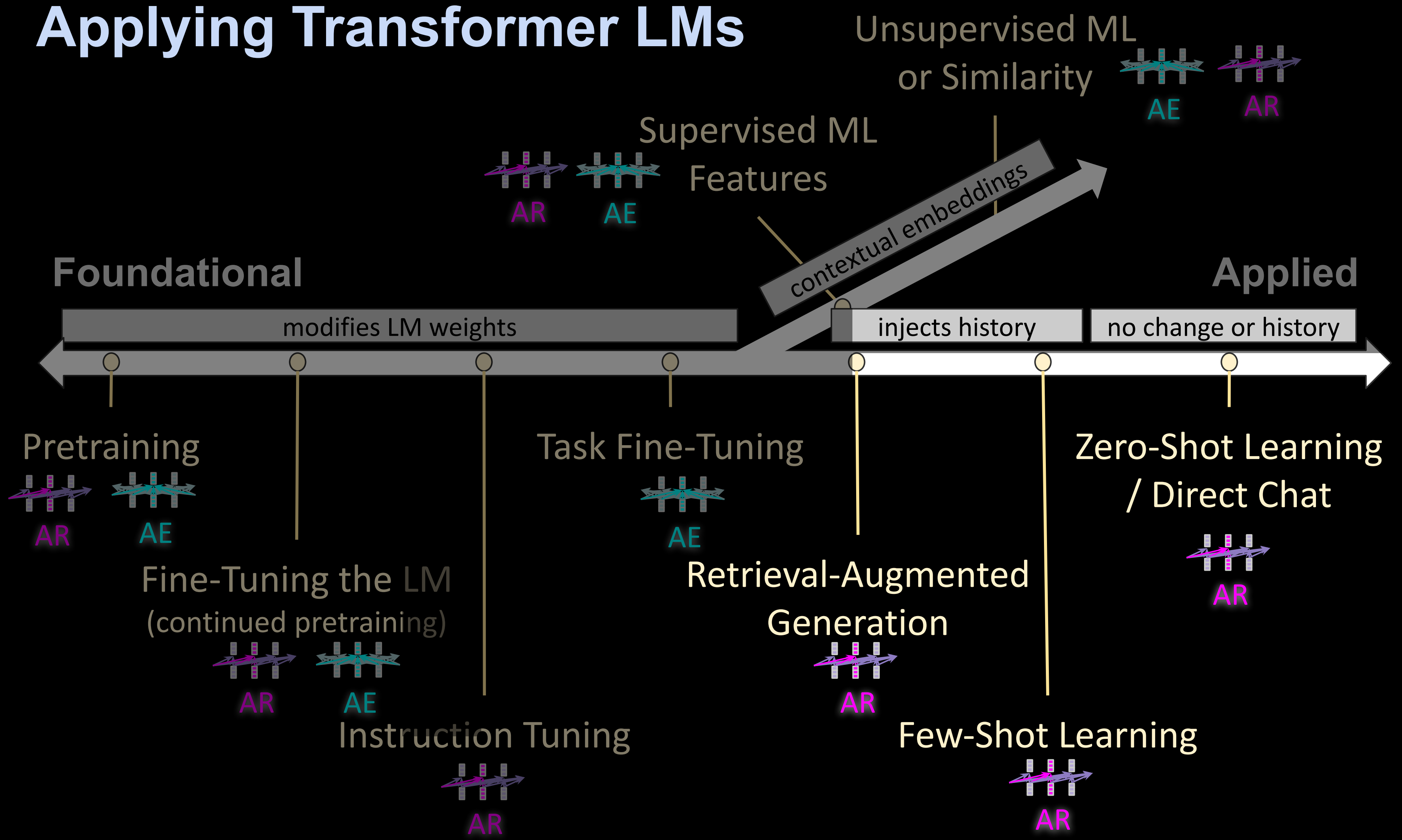


(Kjell, Kjell, and Schwartz, 2023)

Applying Transformer LMs



Applying Transformer LMs



RAG, Few-Shot, Zero-Shot

softmax for LM:

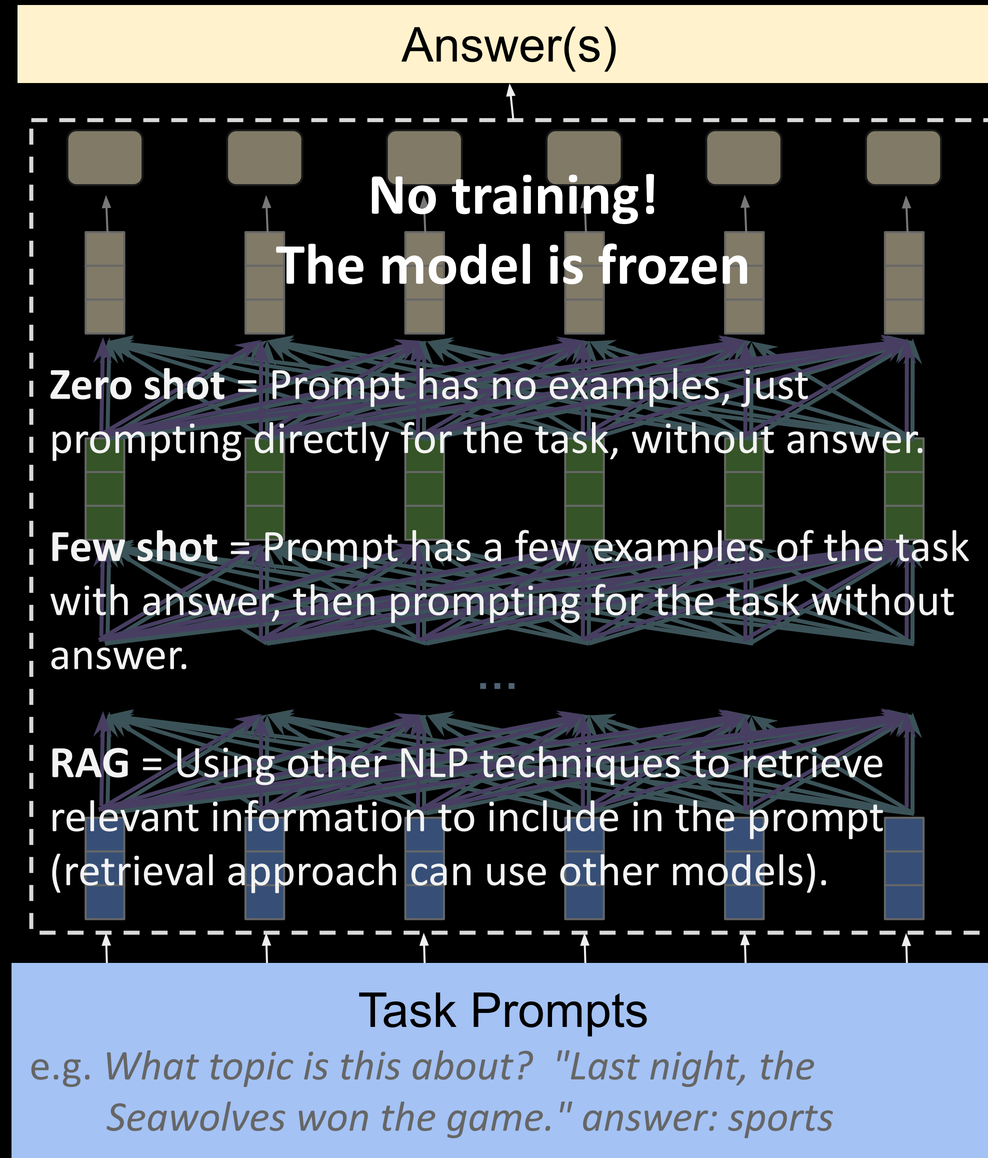
layer k:
(used for language modeling)

layer k-1:
(taken as contextual embedding)

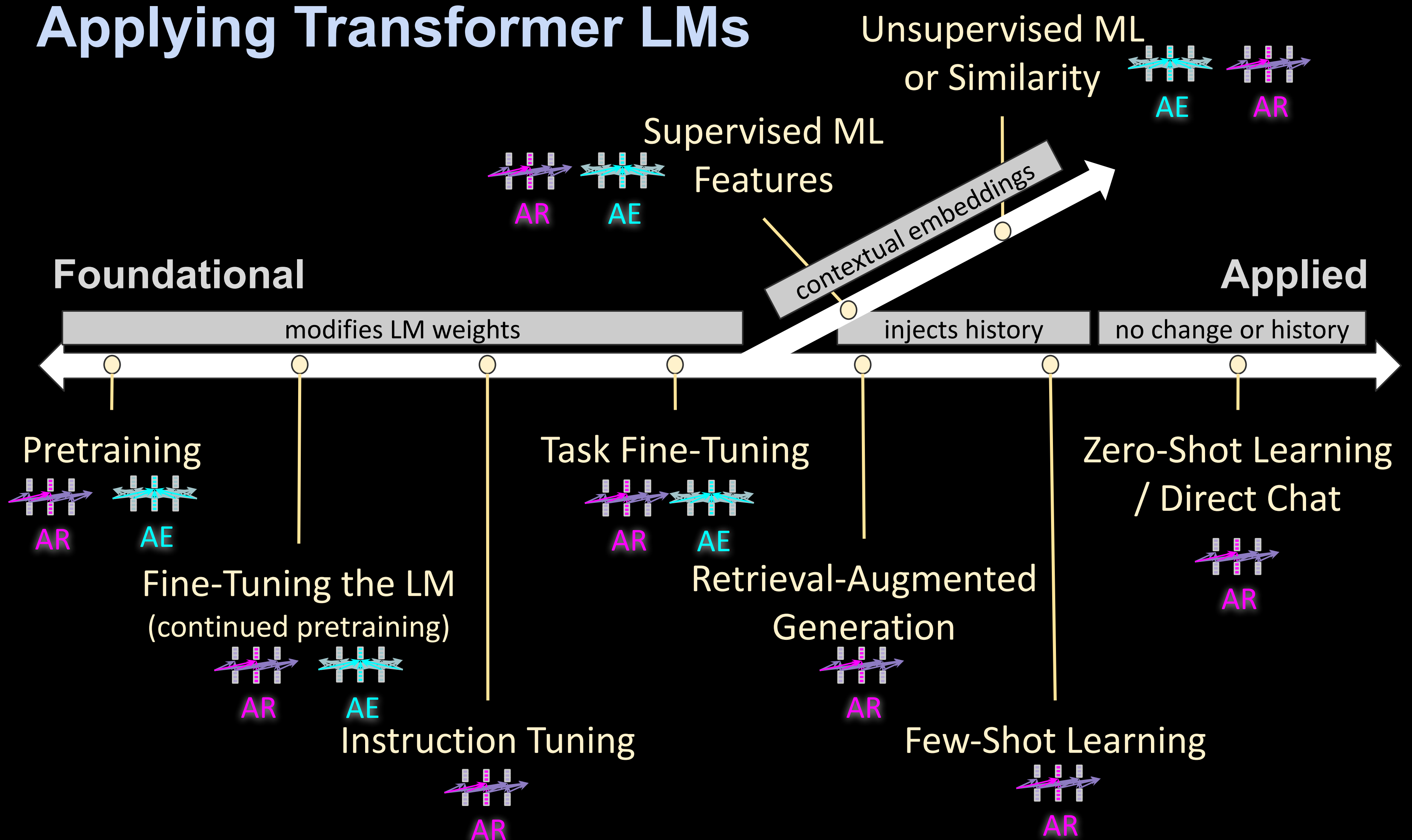
layers 1 to k-2:
(compose embeddings with context)

layer 0:
(input: word-type embeddings)

sentence (sequence) input:



Applying Transformer LMs



How to use an LM for Generation

- Greedy Search
- Beam Search
- Random Walk

How to use an LM for Generation

- **Greedy Search**
- Beam Search
- Random Walk

Always take the most probable next word:

$$\hat{w}_t = \text{argmax}_{w \in V} P(w|\mathbf{w}_{<t})$$

```
def generateGreedy(model, history='<s>'):
    vocabProbs = model.getNextProbs(history)
    history += argmax(vocabProbs)
                 #word with maximum prob
    if history[-1] == '</s>': return history
    else: return generateGreedy(model, history)
```

How to use an LM for Generation

- **Greedy Search**
- Beam Search
- Random W

Always take the most probable next word:

Problem:

p('<s> ok ok </s>')
=.28

p('<s> yes yes </s>'
=.20

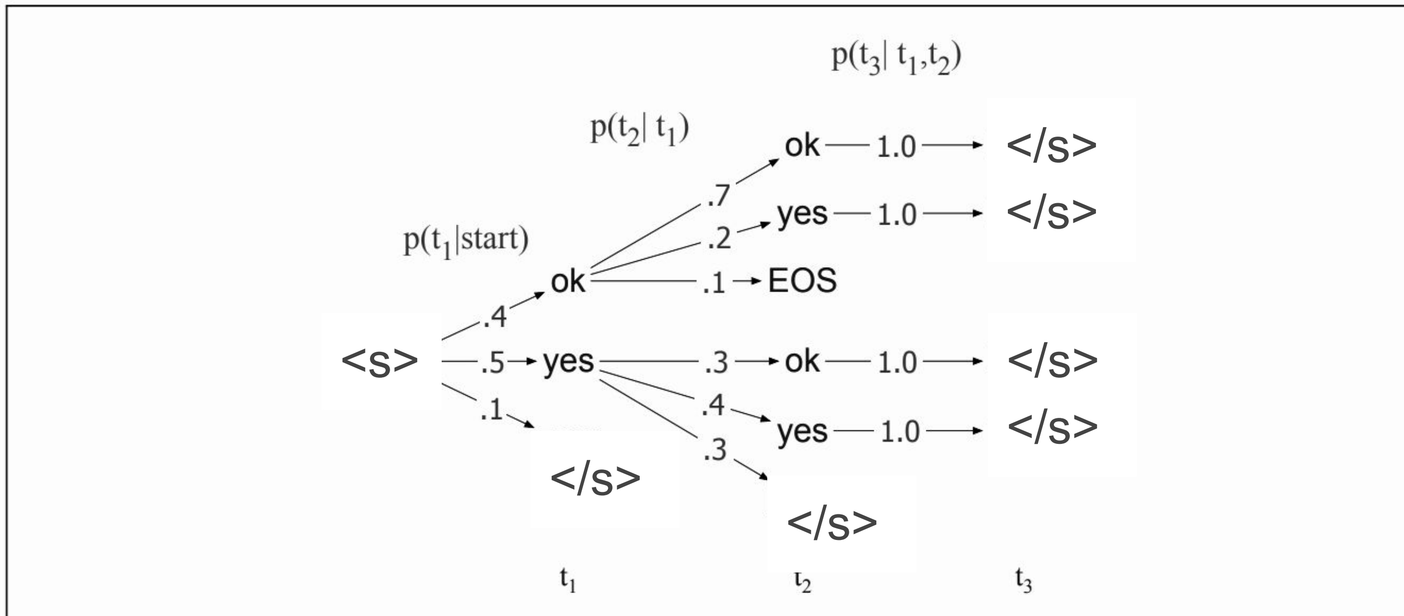


Figure 13.7 A search tree for generating the target string $T = t_1, t_2, \ldots$ from vocabulary $V = \{\text{yes}, \text{ok}, \text{<s>}\}$, showing the probability of generating each token from that state. Greedy search chooses *yes* followed by *yes*, instead of the globally most probable sequence *ok ok*.

How to use an LM for Generation

- Greedy Search
- **Beam Search**
- Random Walk

Evaluate among multiple sequences.

Restrict to consider the top k (*beam width*) most probable per step.

```
def generateBeam(model, history='<s>', init_prob=1, k=4):
  frontier = [(history, init_prob)]
  max_path = []
  max_path_p = -1.0
    while path, path_p in frontier:
      if path[-1] == "</s>": #current max
         if path_p > max_path_p:
             max_path = path
             map_path_p = path_p
      else:
         vocabProbs = model.getNextProbs(path)
         nextWPs = topK(vocabProbs, k)
         for w, p in nextWPs.items():
             frontier.append((path+w, path_p*p))
  return max_path, max_path_p
```

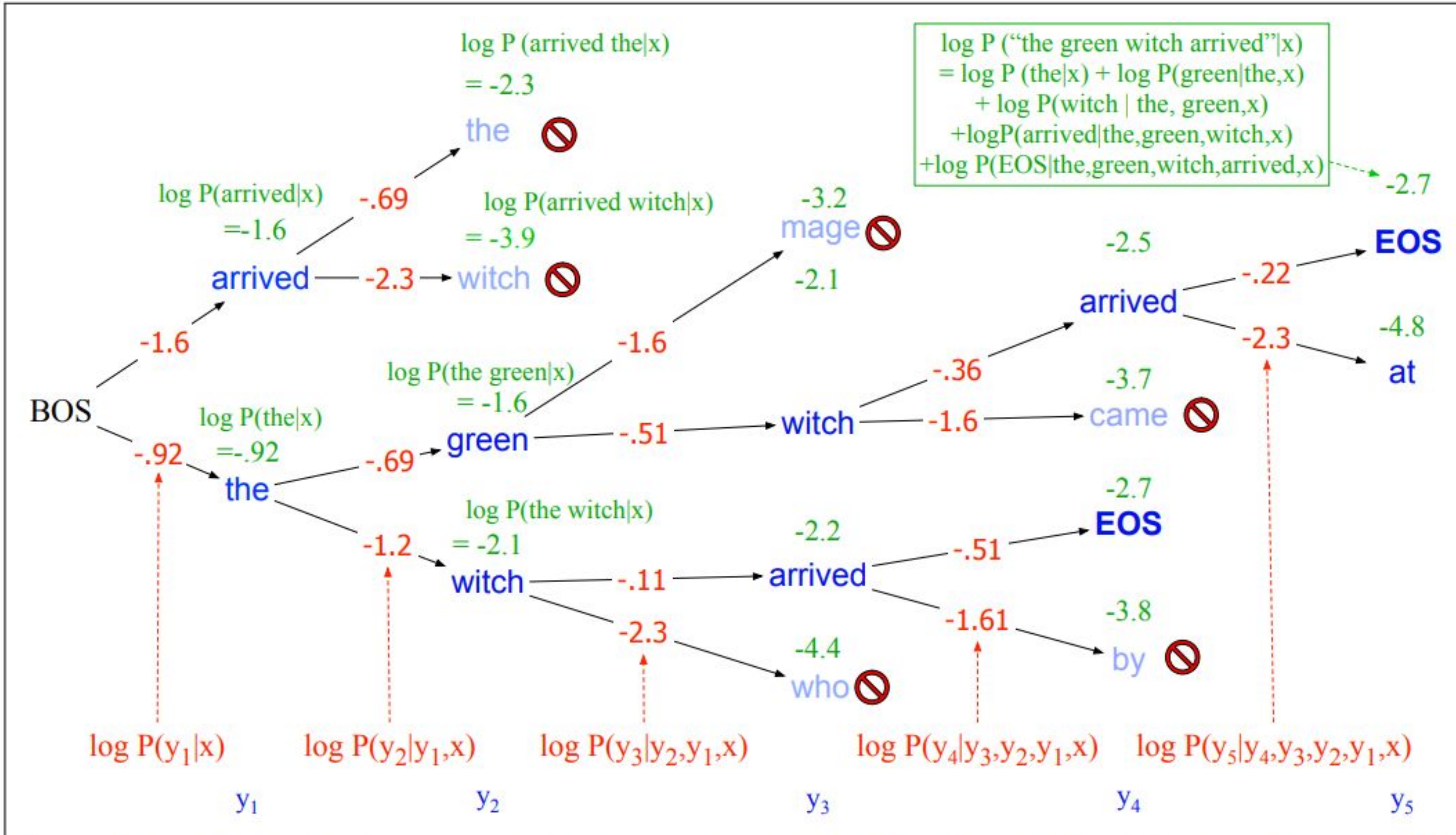
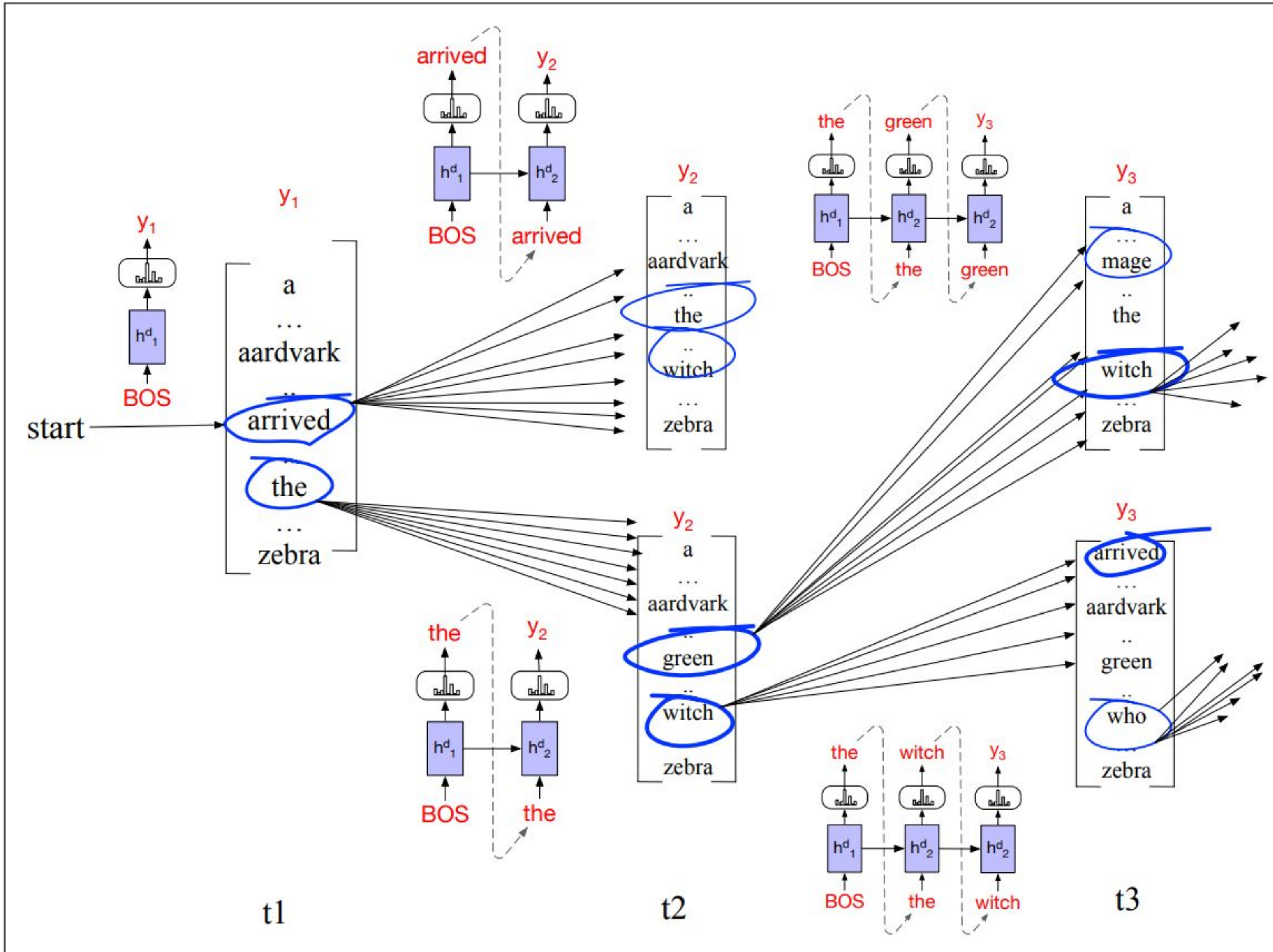


Figure 13.9 Scoring for beam search decoding with a beam width of $k = 2$. We maintain the log probability of each hypothesis in the beam by incrementally adding the logprob of generating each next token. Only the top k paths are extended to the next step.



start
BOS
y_1
h^d_1
a
…
aardvark
..
arrived
..
the
…
zebra
arrived
y_2
h^d_1
h^d_2
BOS
arrived
the
y_2
BOS
the
a
…
aardvark
..
the
..
witch
…
zebra
green
witch
the
green
y_3
h^d_2
BOS
the
green
witch
a
…
mage
..
the
witch
…
zebra
arrived
…
aardvark
..
green
..
who
…
zebra
t1
t2
t3

How to use an LM for Gene

- Greedy Search
- **Beam Search**
- Random Walk

Evaluate among multiple sequences.

Restrict to consider the top k (*beam width*) most probable per step.

```
def generateBeam(mode
  frontier = [(histor
  max_path = []
  max_path_p = -1.0
    while path, path_p in fron     :
      if path[-1] == "</s>": #current potential end
         if path_p > max_path_p:
              max_path = path
              map_path_p = path_p
      else:
         vocabProbs = model.getNextProbs(path)
         nextWPs = topK(vocabProbs, k)
         for w, p in nextWPs.items():
              frontier.append((s+w, path_p*p))
  return max_path, max_path_p
```

Disadvantage: Focuses on the most probable, which is the most typical. Results in very "average sounding" utterances.

How to use an LM for Generation

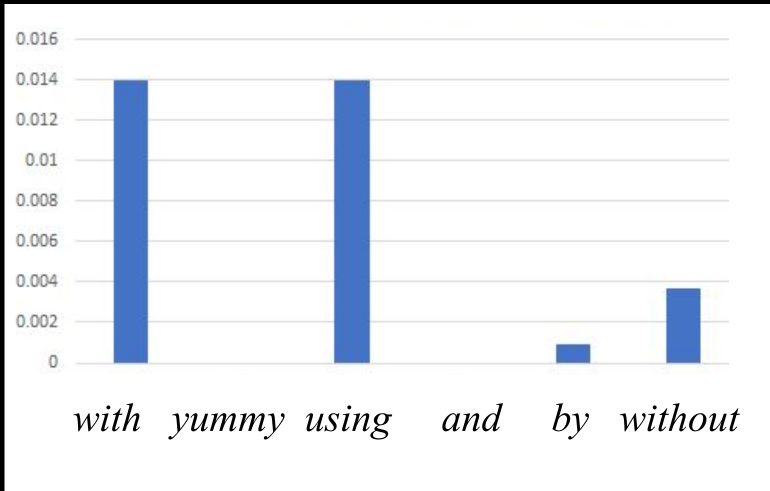
- Greedy Search
- Beam Search
- **Random Walk**

```
def generateRandWalk(model, history='<s>'):
    vocabProbs = model.getNextProbs(history)
    history += multinomial.draw(vocabProbs)
                #random multinomial draw by probs
    if history[-1] == '</s>': return history
    else: return generateRandWalk(model, history)
```

Task: Estimate $P(w_i \mid w_1, \ldots w_{i-1})$

:P(masked word given history)

P(with | He ate the cake <M>) = ?



How to use an LM for Generation

Practical Points

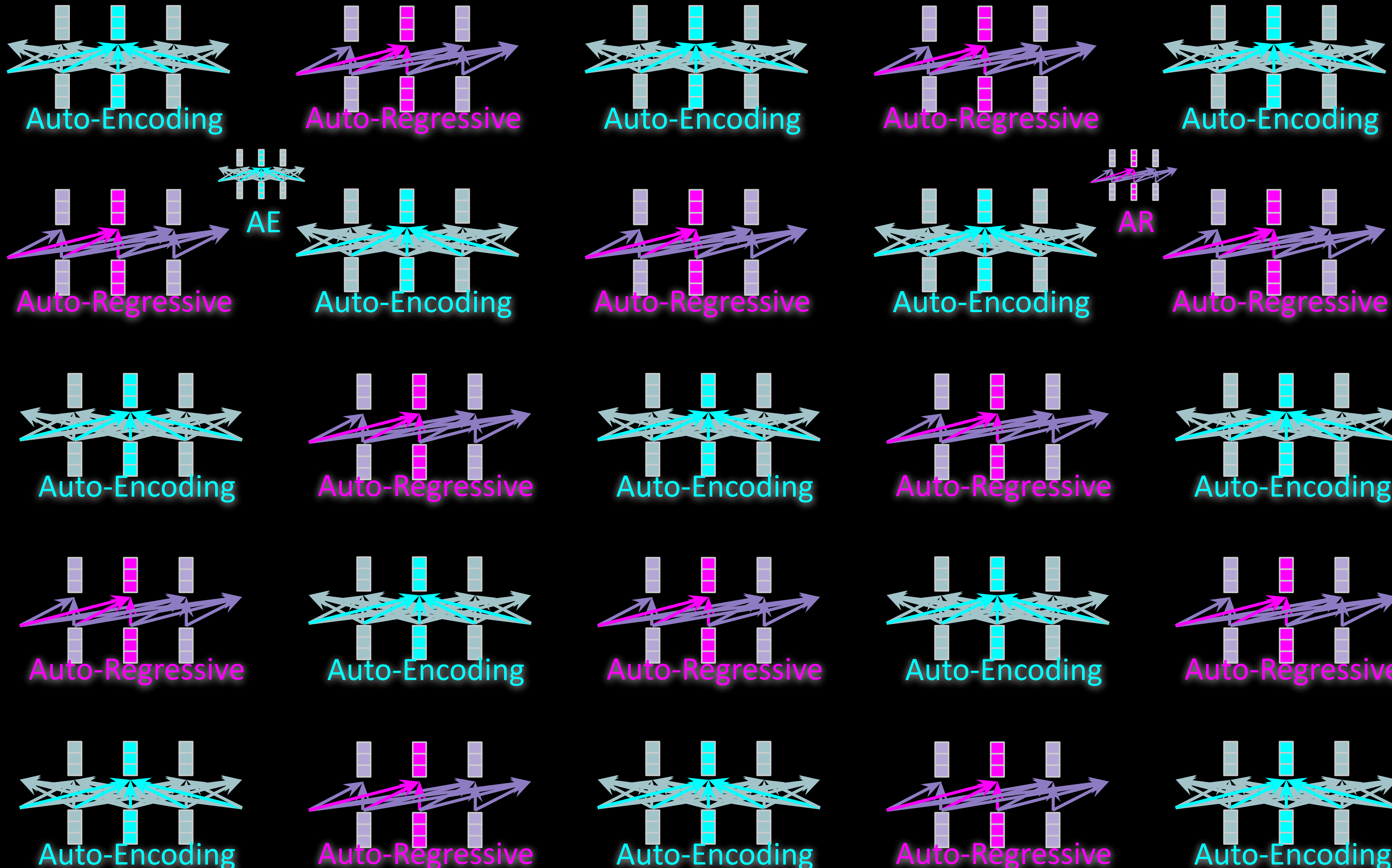
- Use log probs for faster computation tracking maximums.

- Can normalize by length to not favor shorter sequences:

$$score(y) = \log P(y|x) \;=\; \frac{1}{t}\sum_{i=1}^{t} \log P(y_i|y_1,...,y_{i-1},x) \qquad (13.16)$$

- Combine beam and random walk for more novelty.

Supplemental Review Material



Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
AE
Auto-Encoding
Auto-Regressive
Auto-Encoding
AR
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding

Linear Regression as DAG

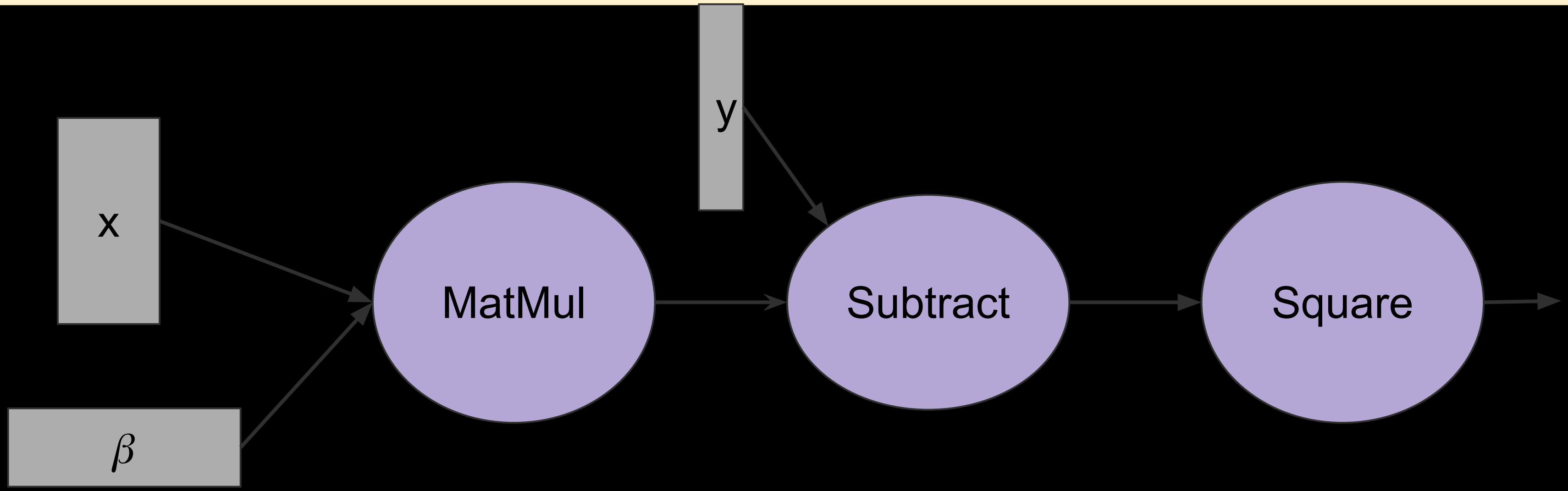
How do Machine learning/ Deep learning frameworks represent these models?

Linear Regression as DAG

How do Machine learning/ Deep learning frameworks represent these models?

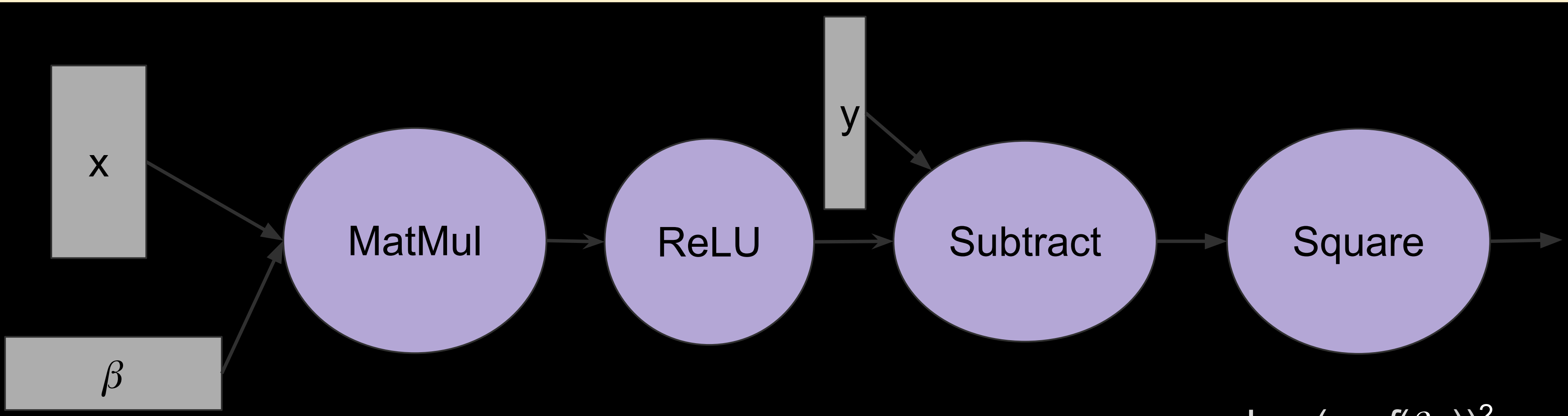
Computational Graph!

Linear Regression as DAG



$L = (y - \beta x)^2$

Linear Regression as DAG



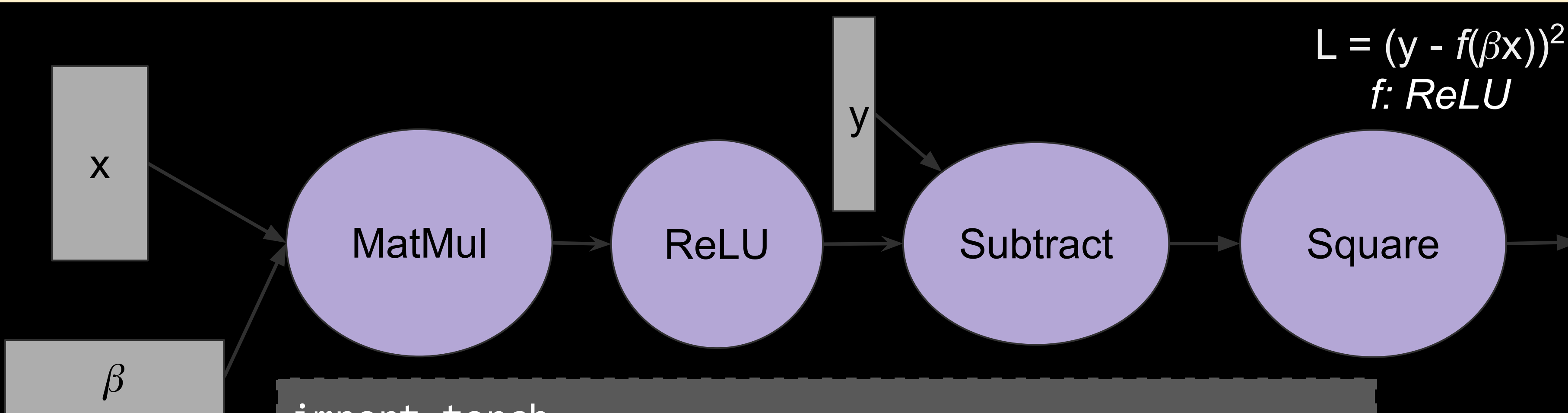
$L = (y - f(\beta x))^2$

f: ReLU

Linear Regression as DAG

$$L = (y - f(\beta x))^2$$

f: ReLU




```
import torch
from torch import nn

x = torch.Tensor(input)
beta = torch.random.randn(X.shape, 1)
z = torch.matmul(x, beta)
yhat = nn.functional.relu(z)
loss = nn.MSELoss(yhat, torch.Tensor(y))
```

PyTorch Demo

Native Linear Regression Implementation (Link)

Torch.nn Linear Regression Implementation (Link)

Linear Regression

Linear Regression: $\hat{y} = \beta X$

Objective: *Learn w, such that* $(y - \beta X)^2$ *is minimized*

Linear Regression

Linear Regression: $\hat{y} = \beta X$

Objective: *Learn w, such that* $(y - \beta X)^2$ *is minimized*

How do we solve for β?

Linear Regression

Linear Regression: $\hat{y} = \beta X$

Objective: *Learn w, such that $(y - \beta X)^2$ is minimized*

How do we solve for β?

1. Analytic Gradient: Differentiate the objective, solve the system of equations by equating it to 0

Linear Regression

Linear Regression: $\hat{y} = \beta X$

Objective: *Learn w, such that* $(y - \beta X)^2$ *is minimized*

How do we solve for β?

1. Analytic Gradient: Differentiate the objective, solve the system of equations by equating it to 0

$$\beta_{opt} = (X^T X)^{-1} X^T y$$

Linear Regression

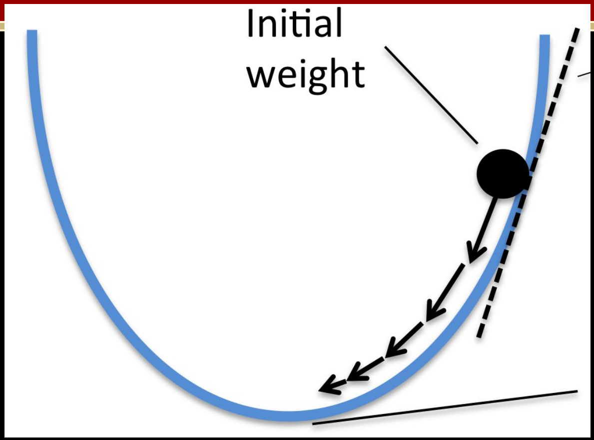
Linear Regression: $\hat{y} = \beta X$

Objective: *Learn w, such that* $(y - \beta X)^2$ *is minimized*

How do we solve for β?

1. Analytic Gradient: Differentiate the objective, solve the system of equations by equating it to 0
2. Numerical Gradient: Start at a random point and move in the direction of minima until optima is reached

Linear Regression



Linear Regression: $\hat{y} = \beta X$

Objective: *Learn w, such that* $(y - \beta X)^2$ *is minimized*

How do we solve for β?

1. Analytic Gradient: Differentiate the objective, solve the system of equations by equating it to 0
2. **Numerical Gradient: Start at a random point and move in the direction of minima until optima is reached**

Numerical Gradient Approach

Linear Regression: Trying to find “betas” that minimize:

$$\beta^* = \mathrm{argmin}_\beta \{\sum_i (y_i - \hat{y}_i)^2\}$$

Numerical Gradient Approach

Linear Regression: Trying to find “betas” that minimize:

$$\beta^* = \text{argmin}_\beta \{\sum_i (y_i - \hat{y}_i)^2\}$$

matrix multiply

$$\hat{y}_i = X_i\beta$$

Numerical Gradient Approach

Linear Regression: Trying to find "betas" that minimize:

$$\beta^* = \text{argmin}_\beta \{\sum_i (y_i - \hat{y}_i)^2\}$$

matrix multiply

$\hat{y}_i = X_i\beta$

Thus: $$\beta^* = \text{argmin}_\beta \{\sum_i (y_i - X_i\beta)^2\}$$

Numerical Gradient Approach

Linear Regression: Trying to find "betas" that minimize:

$$\beta^* = \text{argmin}_\beta \{\textstyle\sum_i (y_i - \hat{y}_i)^2\}$$

matrix multiply

$\hat{y}_i = X_i\beta$

Thus: $\beta^* = \text{argmin}_\beta \{\sum_i (y_i - X_i\beta)^2\}$

How to update?

$$\beta_{new} = \beta_{old} - \alpha * \text{grad}$$

Numerical Gradient Approach

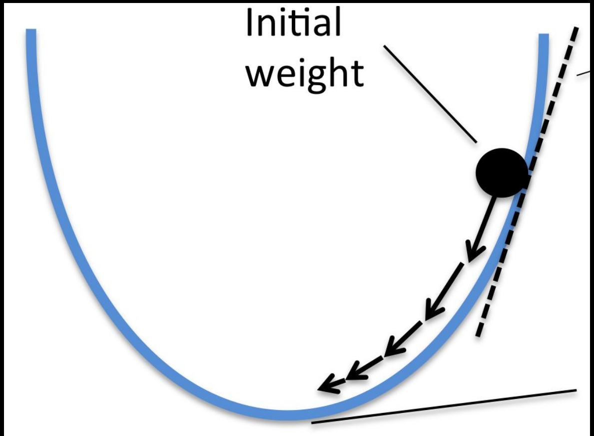
Linear Regression: Trying to find "betas" that minimize:

$$\beta^* = \text{argmin}_\beta \{\textstyle\sum_i (y_i - \hat{y}_i)^2\}$$

matrix multiply

$\hat{y}_i = X_i\beta$

Thus: $$\beta^* = \text{argmin}_\beta \{\textstyle\sum_i (y_i - X_i\beta)^2\}$$



How to update?

$$\beta_{new} = \beta_{old} - \boldsymbol{\alpha} * \text{grad}$$

$\boldsymbol{\alpha}$: Learning Rate

Numerical Gradient Approach

Linear Regression: Trying to find "betas" that minimize:

Gradient Descent: $\beta_{new} = \beta_{old} - \alpha * \text{grad}$

Numerical Gradient Approach

Linear Regression: Trying to find “betas” that minimize:

Gradient Descent: $\beta_{new} = \beta_{old} - \alpha * \text{grad}$

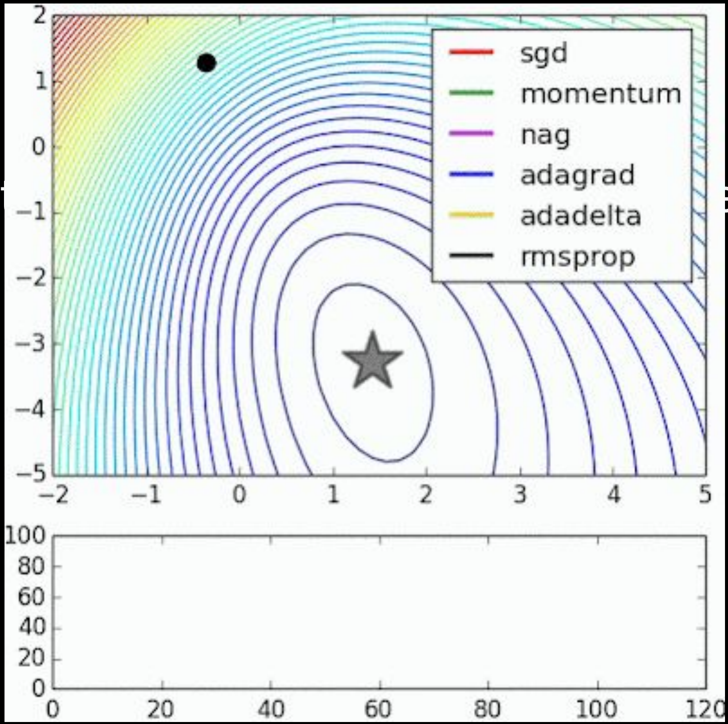
But there are other gradient descent based optimization methods which are better*

Numerical Gradient Approach

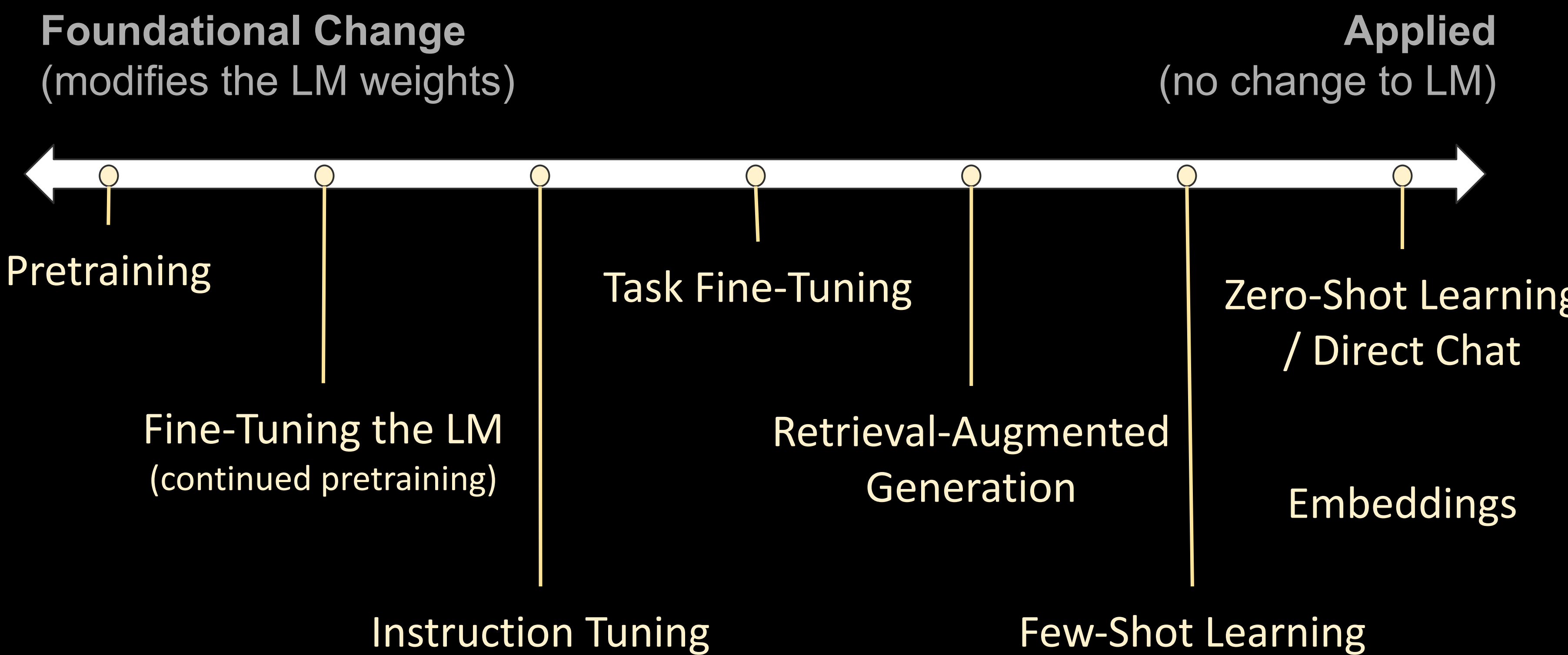
Linear Regression: Trying to find "betas" that minimize:

Gradient Descent: $\beta_{new} = \beta_{old} - \alpha * grad$

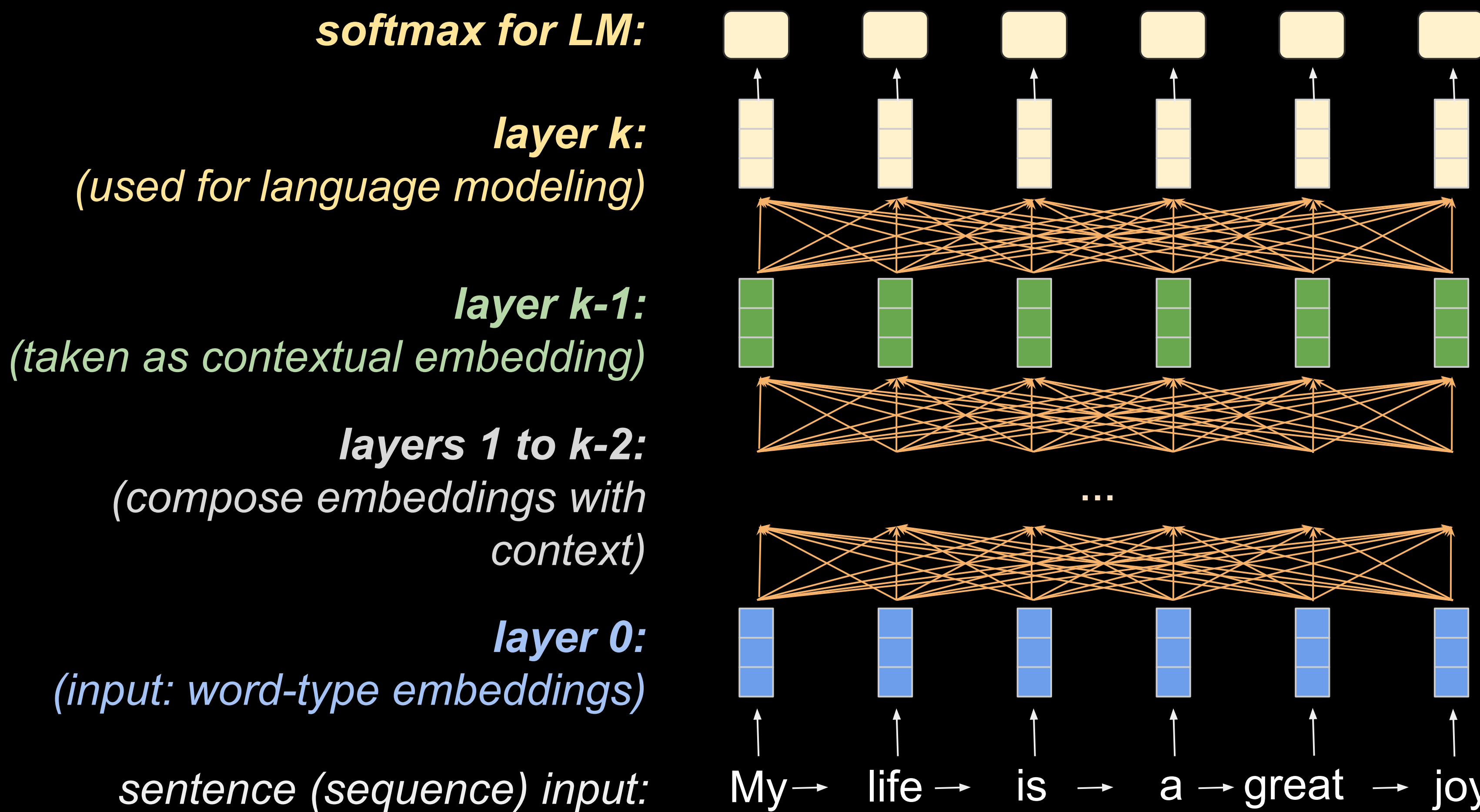
But there are other gradient descent based op… er*



simpler version

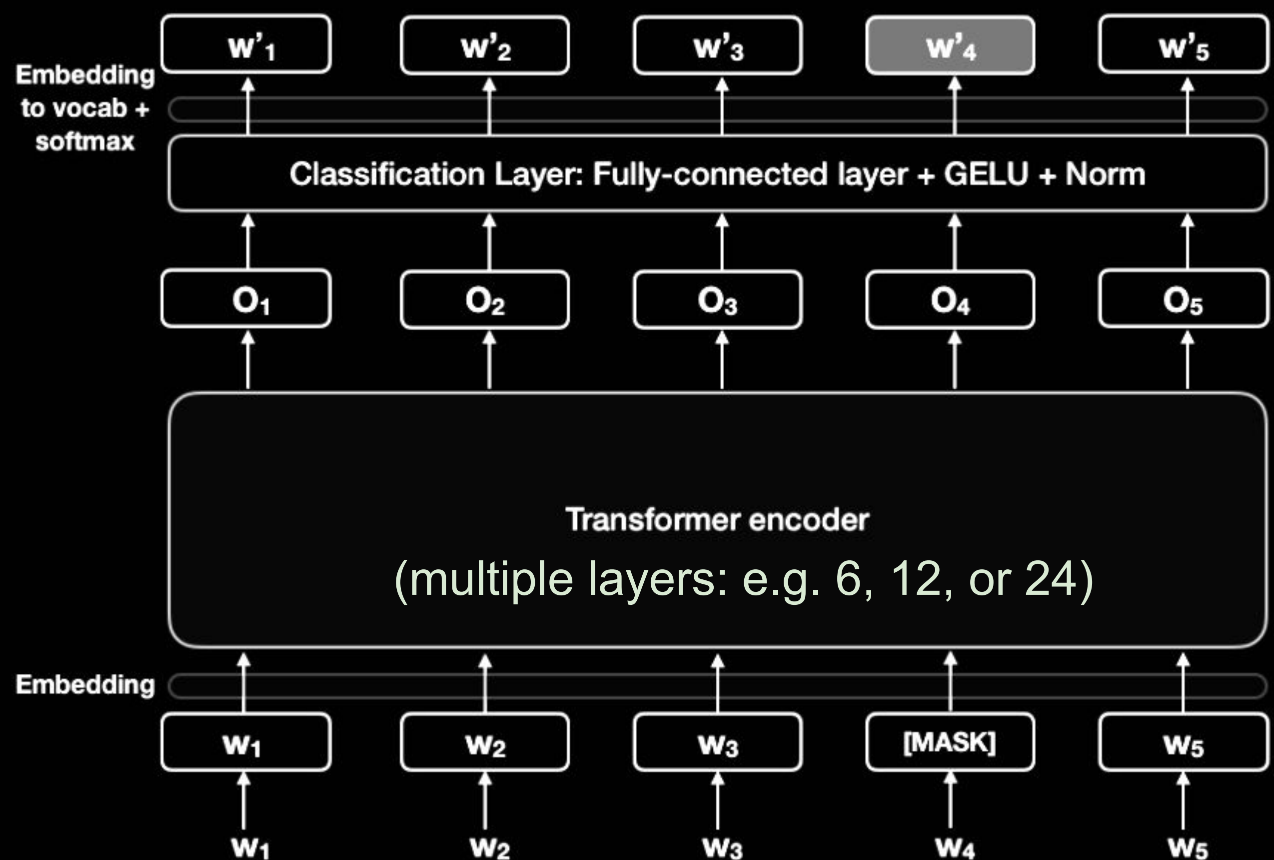


Pretraining; FTing the LM; Instruction Tuning

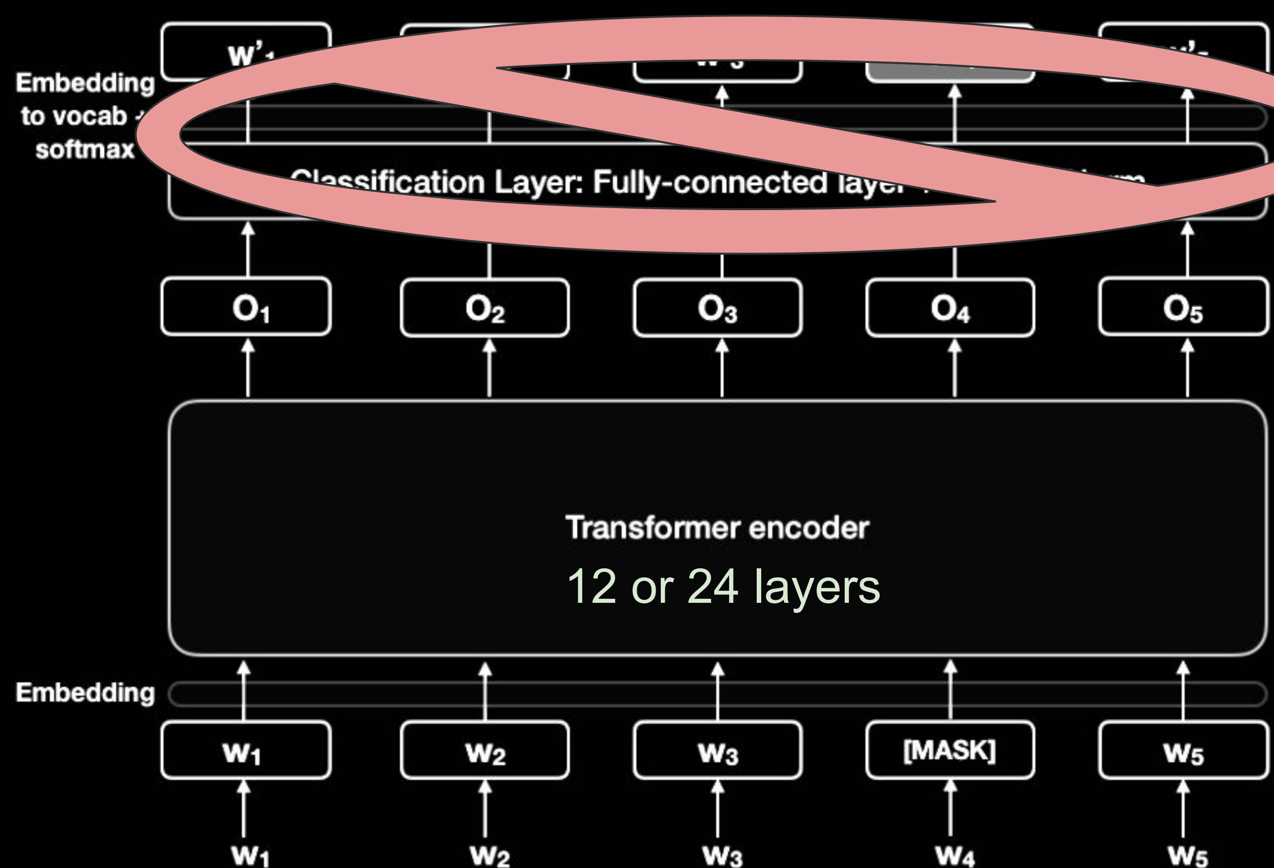


(Kjell, Kjell, and Schwartz, 2023)

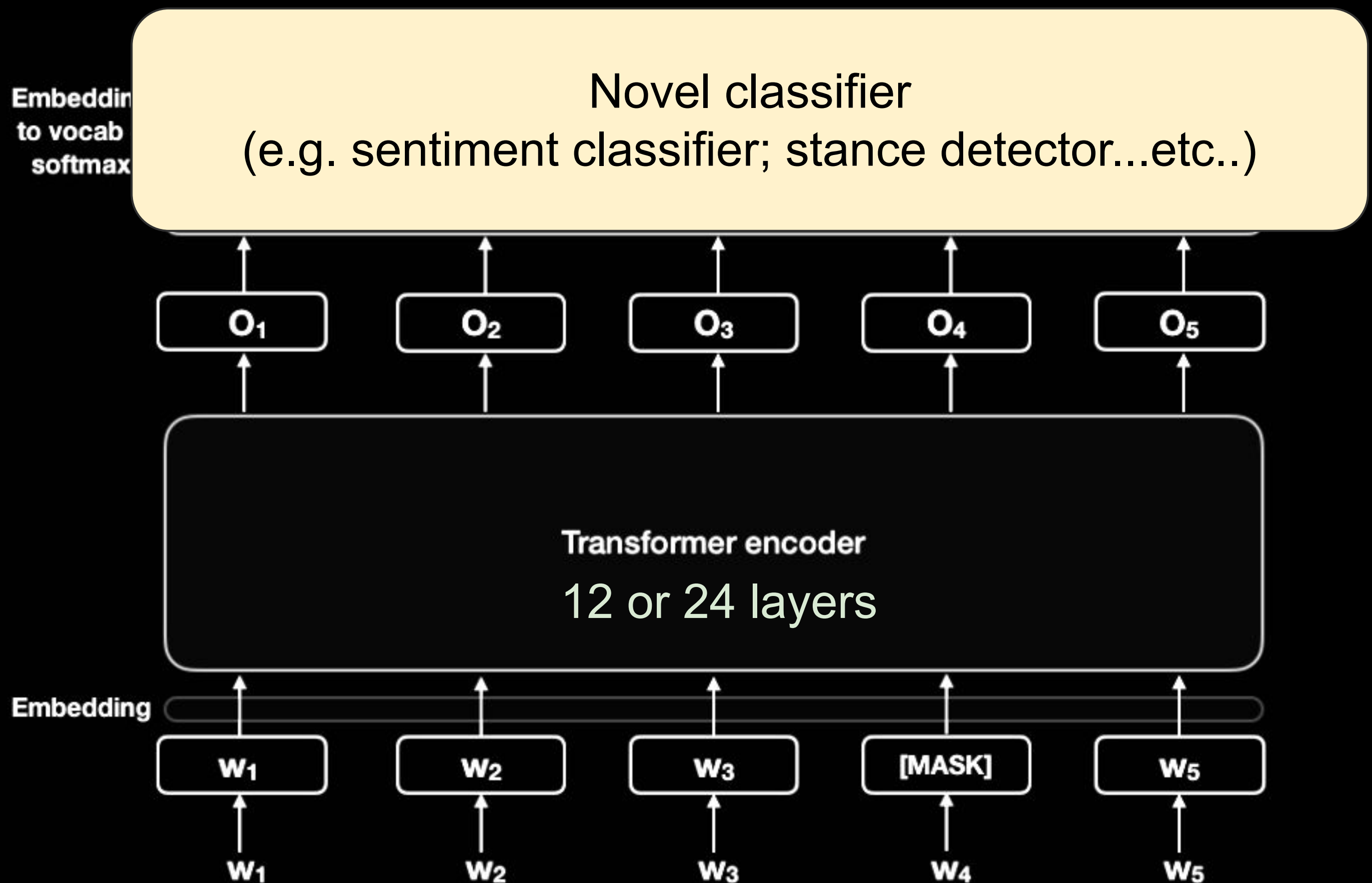
Pre-training; Fine-tuning the LM; Instruction Tuning



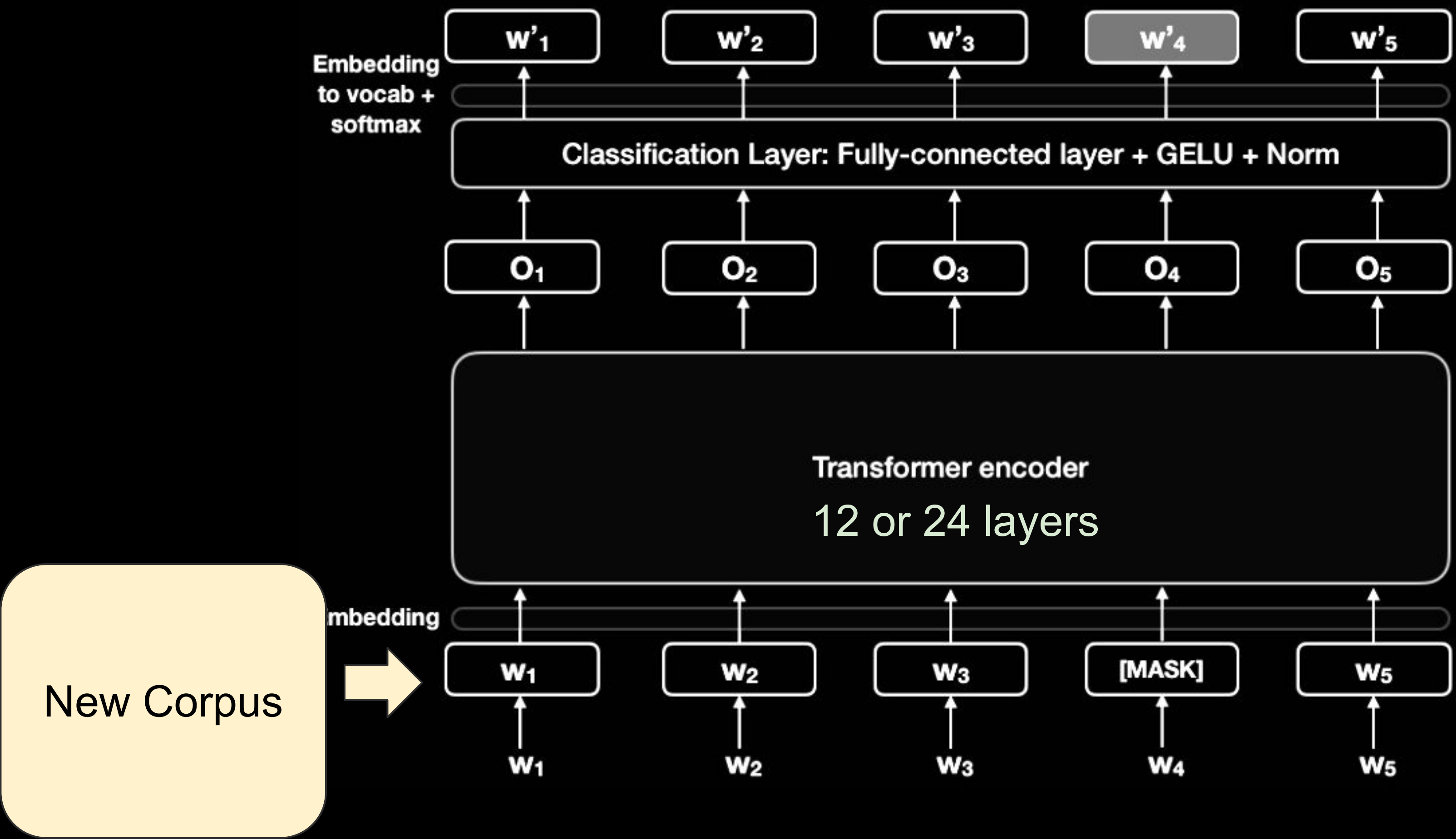
BERT: Pre-training; Fine-tuning

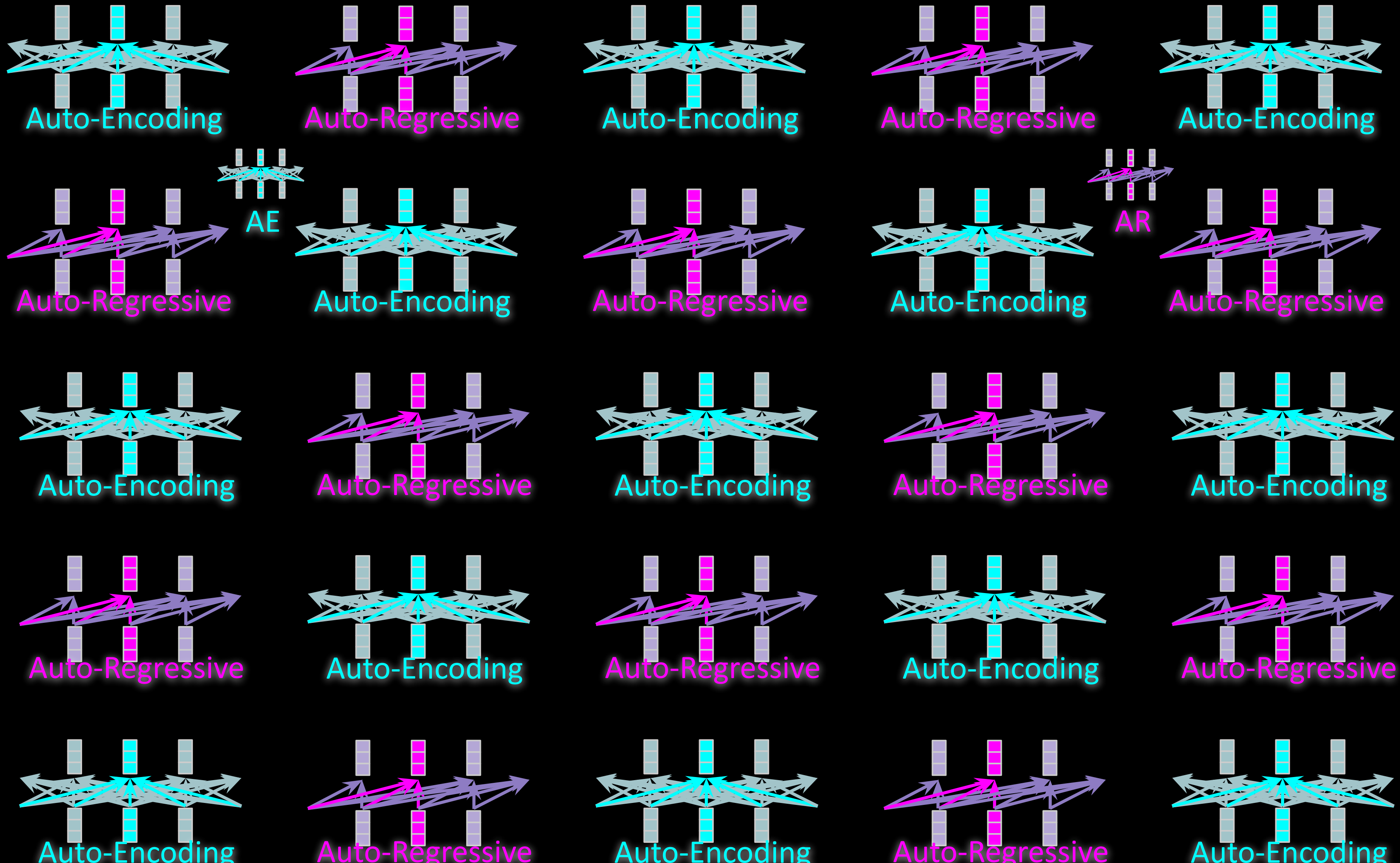


BERT: Pre-training -> Task Fine-tuning



BERT: Pre-training -> LM Fine-tuning




Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
AE
Auto-Encoding
Auto-Regressive
Auto-Encoding
AR
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding
Auto-Regressive
Auto-Encoding