

Contextualized Word Representations

CS 490A, Fall 2021

Applications of Natural Language Processing

https://people.cs.umass.edu/~brenocon/cs490a_f21

Brendan O'Connor & Laure Thompson

College of Information & Computer Sciences
University of Massachusetts Amherst

Administrivia

- Project Progress Report due **Monday**, 11/22
- Hugging Face (transformers) virtual tutorial this Friday, 11/19
- HW4 will be released next week

Recall Word Embeddings

So far, we've talked about **vocabulary-level** word embeddings

Other adjectives: **static, type, noncontextual**

Idea: Each word **type** is represented by a k -dimensional vector
where **distance** reflects **semantic** relationships

Q: What are their pros and cons?

- Pre-trained embeddings are powerful! Transfer learning
- Enable many numeric operations
- Easy programming

- Types aren't token
 - ↳ the same word form can have different meanings
- "opposite" encoding
- Rare words, out-of-vocabulary

fastText: Dealing with out-of-vocabulary terms

Idea: Consider a word to be a **bag of n-grams**

Instead of learning representations for each word, learn the representations of their n-grams

Note: the full word is also included (no matter its length)!

fastText: Dealing with out-of-vocabulary terms

Formally, a word representation is the sum of its n-grams' learned representations

$$w = \sum_{g \in G_w} \text{vec}_g$$

fastText: Dealing with out-of-vocabulary terms

Word: **making**

3-grams: <ma mak aki kin ing ng>

4-grams: <mak maki akin king ing>

5-grams: <maki makin aking king>

6-grams: <makin making aking>

Full sequence: <making>

Contextual Word Representations

Idea: Represent each word **token** as a k -dimensional vector that reflects the token's **local context**

Contextual Word Representations

Idea: Represent each word **token** as a k -dimensional vector that reflects the token's **local context**

1. She lived off the *land*
2. She had just enough fuel to *land*

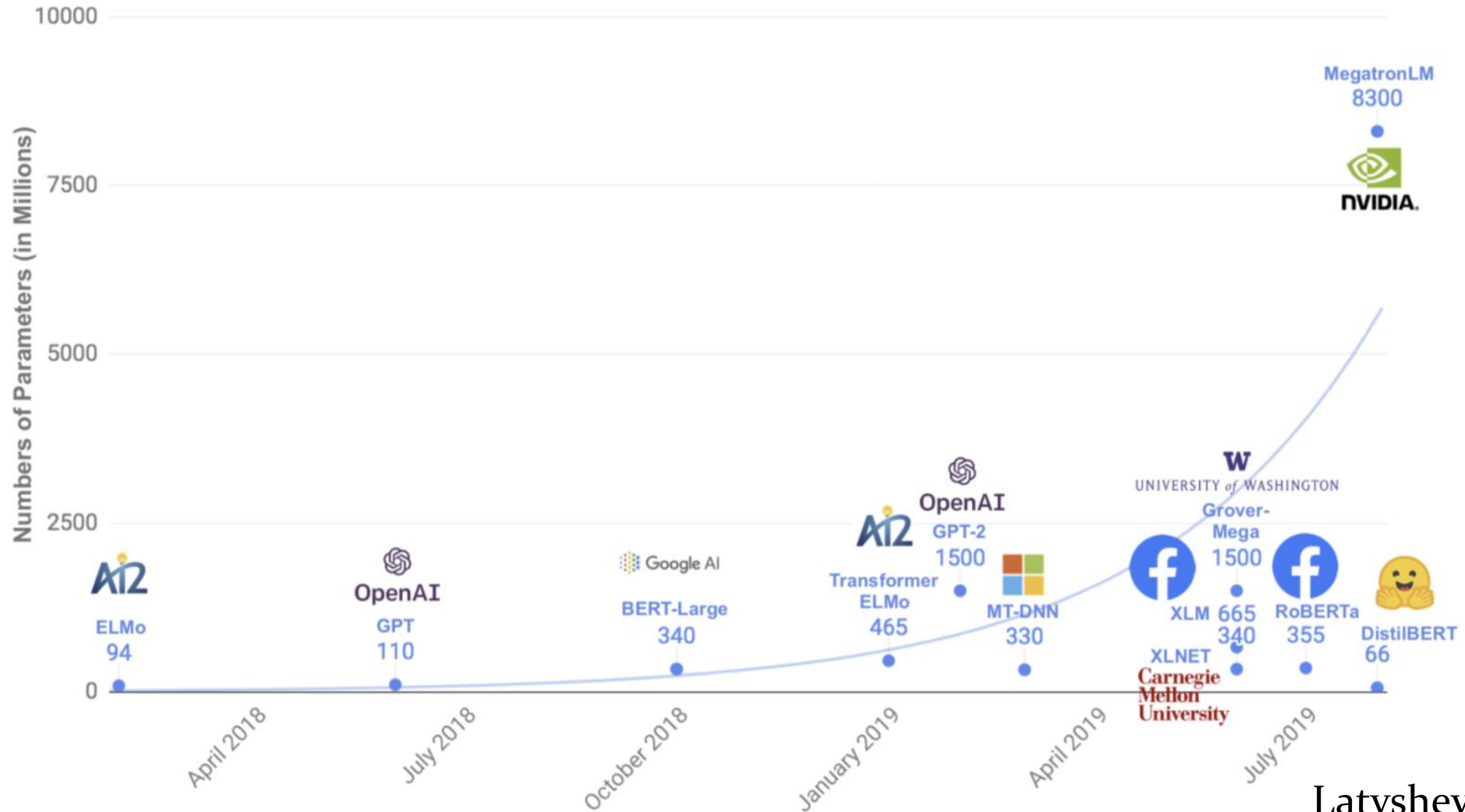
Contextual Word Representations

Idea: Represent each word **token** as a k -dimensional vector that reflects the token's **local context**

1. Caesium is the most reactive alkali *metal*
2. Folk *metal* is a fusion of music genres

Large Pretrained Language Models

Large Pretrained Language Models



Large Pretrained Language Models

- Large model size (millions of parameters)
- Large (pre)training corpus

Model	Size	Data Sources
BERT	16 GB	BookCorpus, Wikipedia (en)
RoBERTa	161GB	BookCorpus, Wikipedia (en), Stories, CommonCrawl News, OpenWebText
GPT-3	570 GB	CommonCrawl (filtered), WebText2, Books1, Books2, Wikipedia (en)
T5	750 GB	C4: Colossal Clean Crawled Corpus

Large Pretrained Language Models

Paradigm Shift:

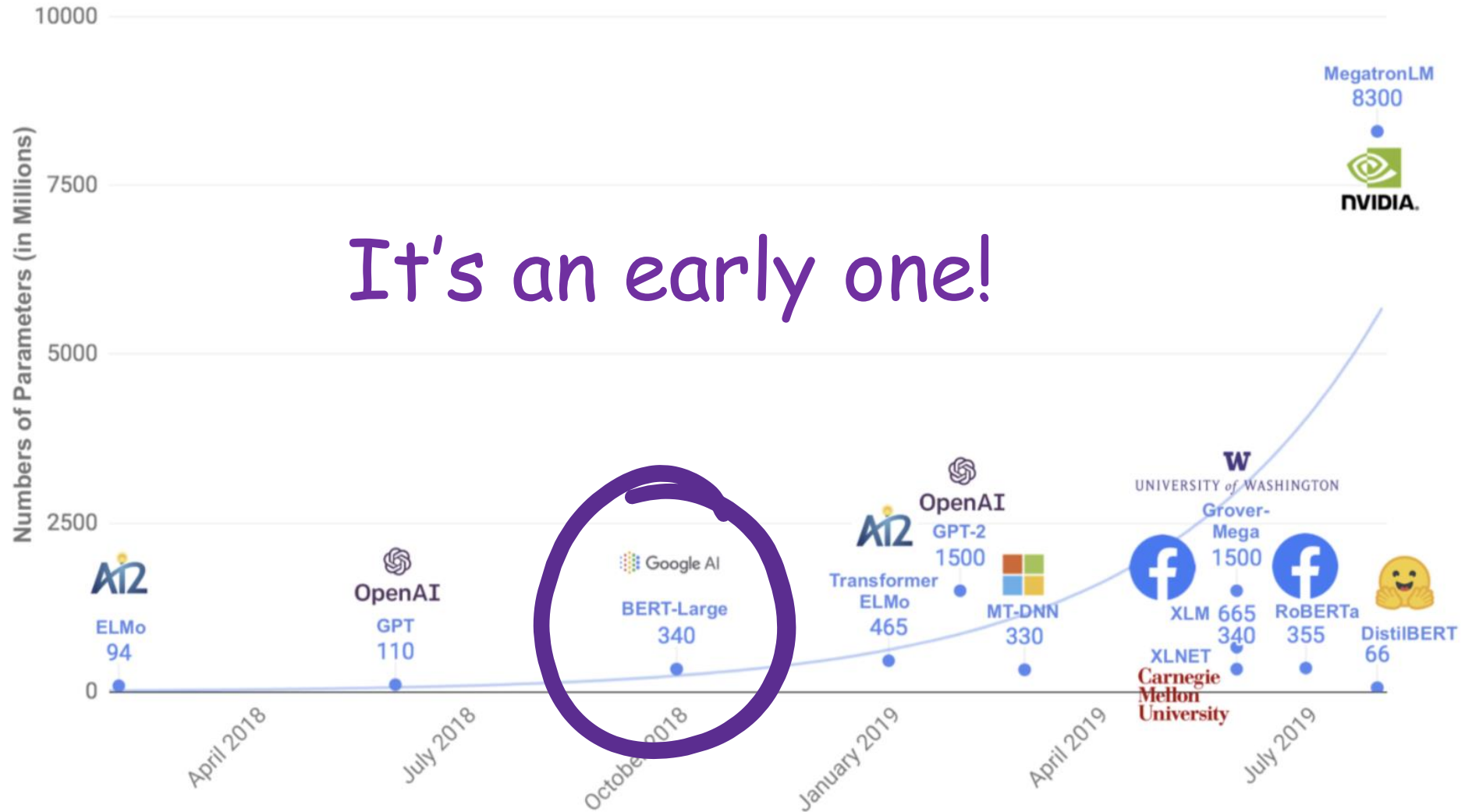
- (a) **Pretrain** a model on general tasks where labels can easily be generated for massive text corpora
- (b) **Fine-tune** the pretrained model (as needed) for a more specialized, harder task



BERT: Bidirectional Encoder Representations for Transformers

Devlin et al. 2019

Why BERT?



Why BERT?

Highly influential!

25,048 Citations

Highly Influential Citations ⓘ 7,670

Background Citations 10,124

Methods Citations 13,635

Results Citations 463

[View All](#)

Why BERT?

Highly influential!

25,048 Citations

Highly Influential Citations ⓘ **7,670**

Background Citations **10,124**

Methods Citations **13,635**

Results Citations **463**

[View All](#)

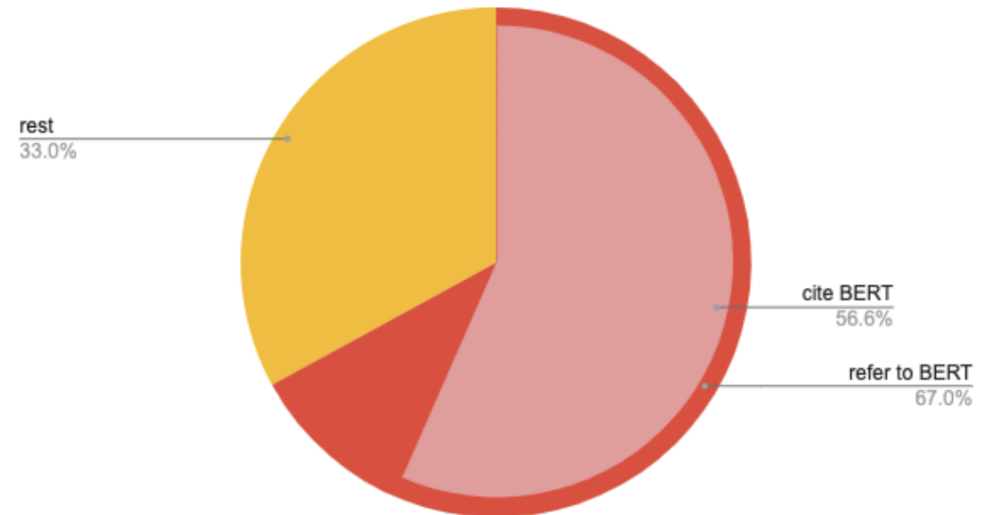


Gabriel Stanovsky
@GabiStanovsky

I skimmed through many papers from [@emnlpmeeting](#), which got me thinking - what % of papers refer to BERT, and out of those, how many cite it? Here's the answer*: 67% of papers refer to BERT (!), and 56% cite it.

*computed automatically, exact #'s may vary
[#EMNLP2021](#) [#NLProc](#)

EMNLP2021 papers (main + findings)



12:21 PM · Nov 9, 2021 · Twitter Web App

BERT is a **Transformer**-based model.

What's a Transformer?



Guido 2005

Let's talk about *attention*

How do we use word embeddings for document classification?

How do we use word embeddings for document classification?



I



loved



the



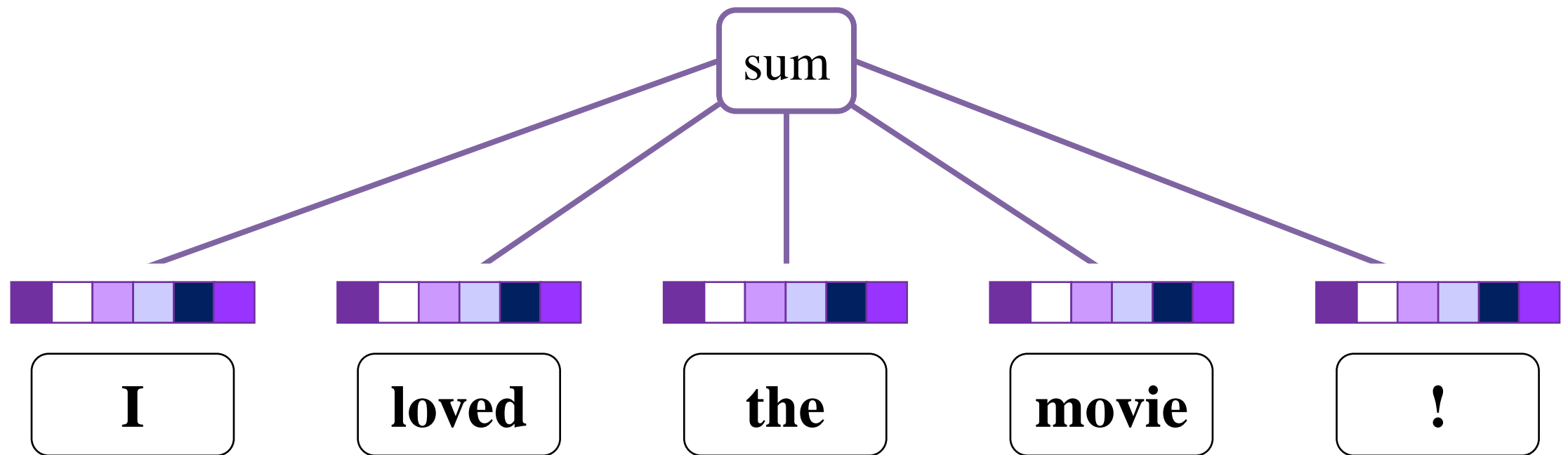
movie



!

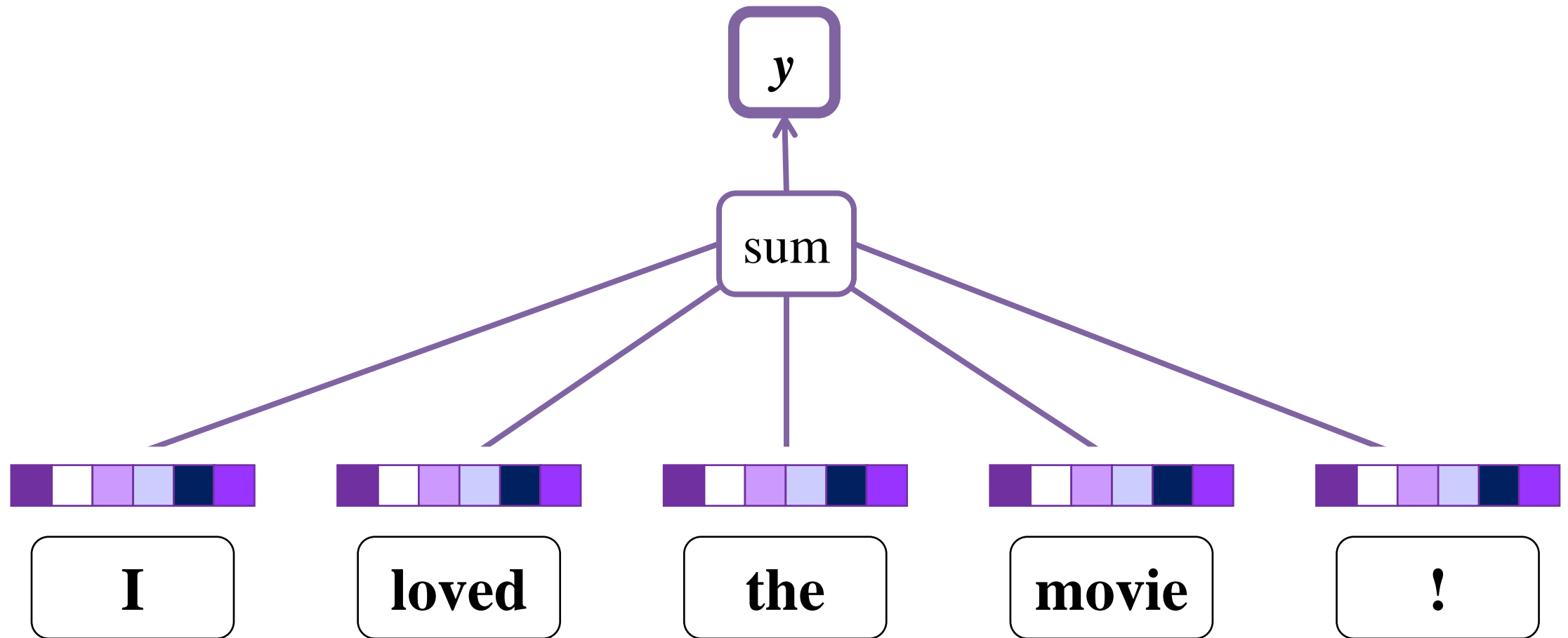
Adapted from the slides by Bamman (2021)

How do we use word embeddings for document classification?



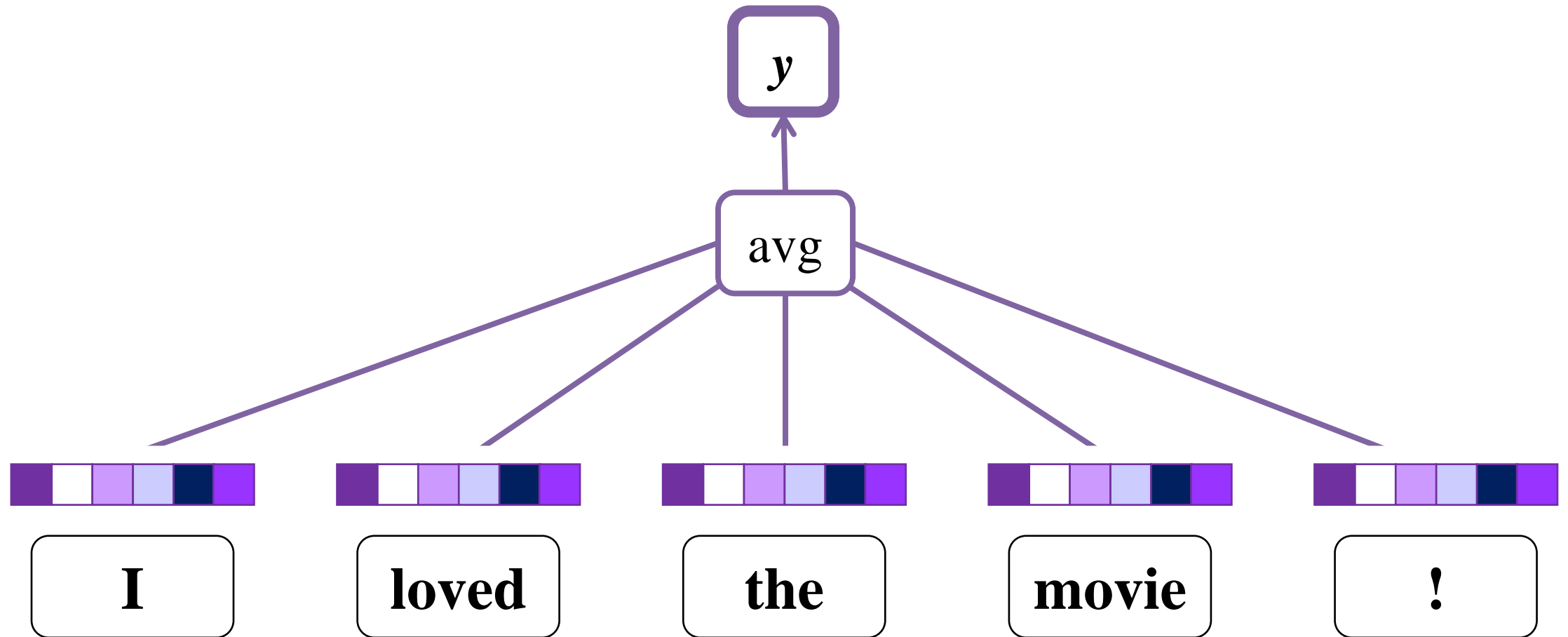
Adapted from the slides by Bamman (2021)

How do we use word embeddings for document classification?



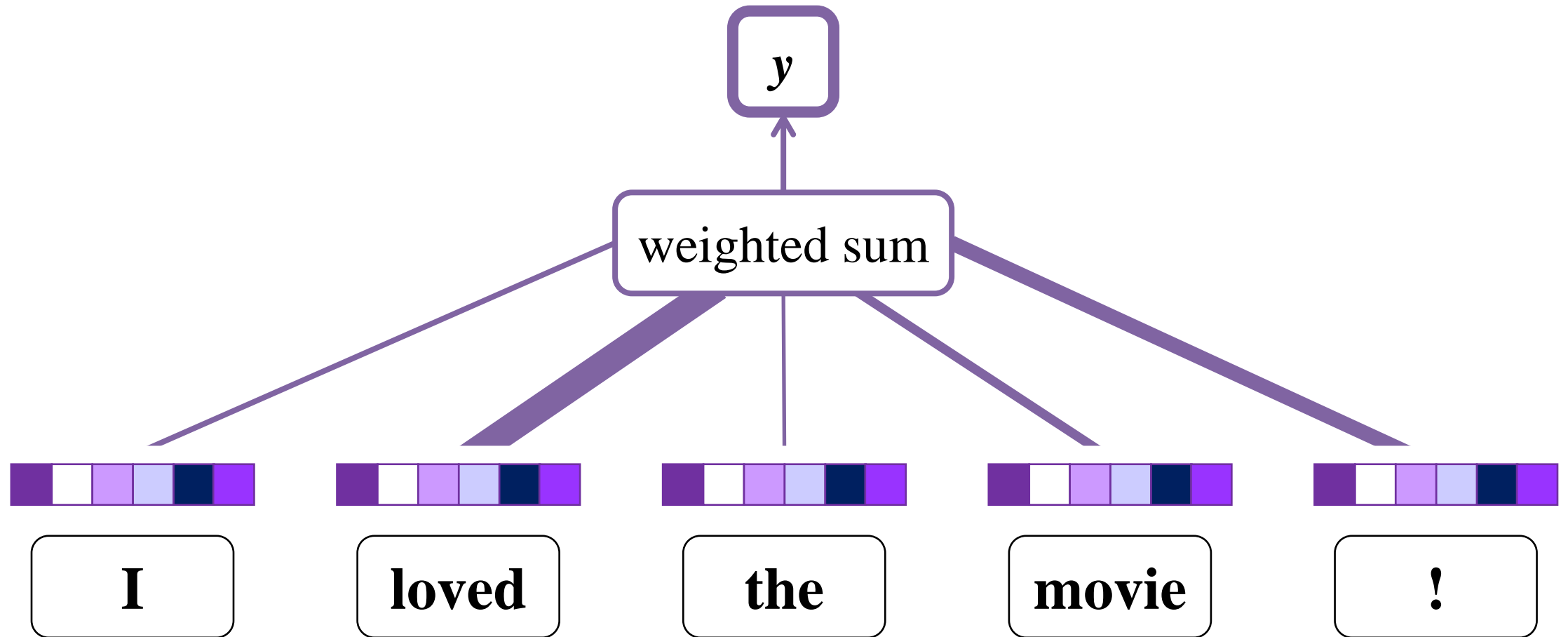
Adapted from the slides by Bamman (2021)

...or we could average them



Adapted from the slides by Bamman (2021); Iyyer et al. 2015

...or take a weighted sum



Adapted from the slides by Bamman (2021)

What is attention?

Idea: Give neural networks structure (and parameter) to learn which input elements we should **attend** to and which we can ignore

What is attention?

Let's learn a task-specific vector \mathbf{v}

Intuition: think of \mathbf{v} like an “**important word**” vector



x_1 : I



x_2 : loved



x_3 : the



x_4 : movie



x_5 : !

Adapted from the slides by Bamman (2021)

What is attention?

We'll measure the “**importance**” of each input vector \mathbf{x} by its similarity (dot product) to \mathbf{v}

$$r_1 = \mathbf{v} \cdot \mathbf{x}_1$$



\mathbf{x}_1 : I

$$r_2 = \mathbf{v} \cdot \mathbf{x}_2$$



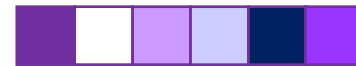
\mathbf{x}_2 : loved

$$r_3 = \mathbf{v} \cdot \mathbf{x}_3$$



\mathbf{x}_3 : the

$$r_4 = \mathbf{v} \cdot \mathbf{x}_4$$



\mathbf{x}_4 : movie

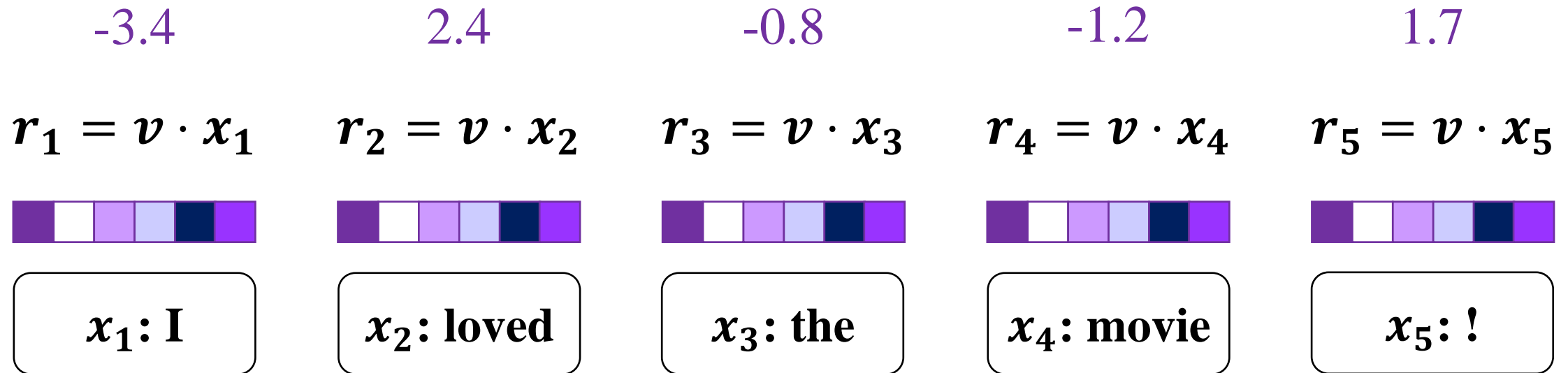
$$r_5 = \mathbf{v} \cdot \mathbf{x}_5$$



\mathbf{x}_5 : !

Adapted from the slides by Bamman (2021)

What is attention?



Adapted from the slides by Bamman (2021)

What is attention?

$$a = \text{softmax}(r)$$

0

-3.4

$$r_1 = v \cdot x_1$$



x_1 : I

0.64

2.4

$$r_2 = v \cdot x_2$$



x_2 : loved

0.02

-0.8

$$r_3 = v \cdot x_3$$



x_3 : the

0.02

-1.2

$$r_4 = v \cdot x_4$$



x_4 : movie

0.32

1.7

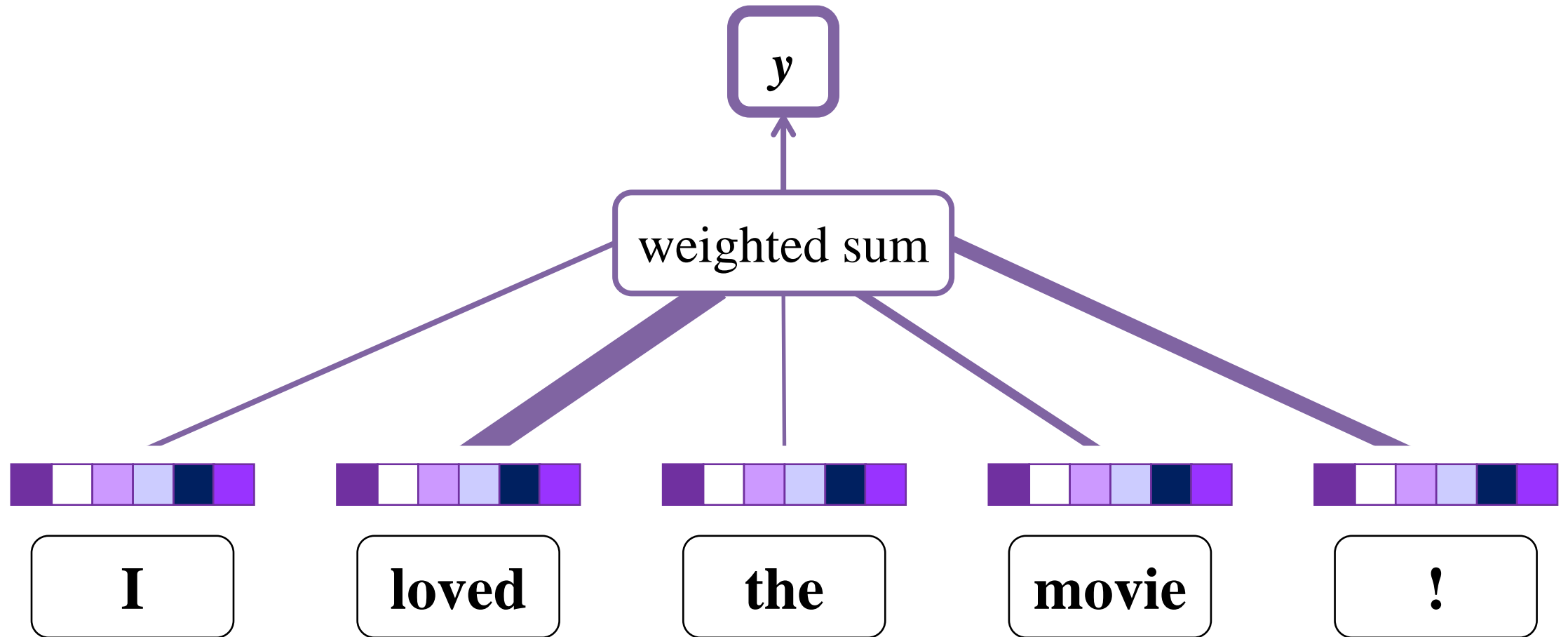
$$r_5 = v \cdot x_5$$



x_5 : !

Adapted from the slides by Bamman (2021)

...this seems familiar



Adapted from the slides by Bamman (2021)

Transformers rely on a
specific kind of attention

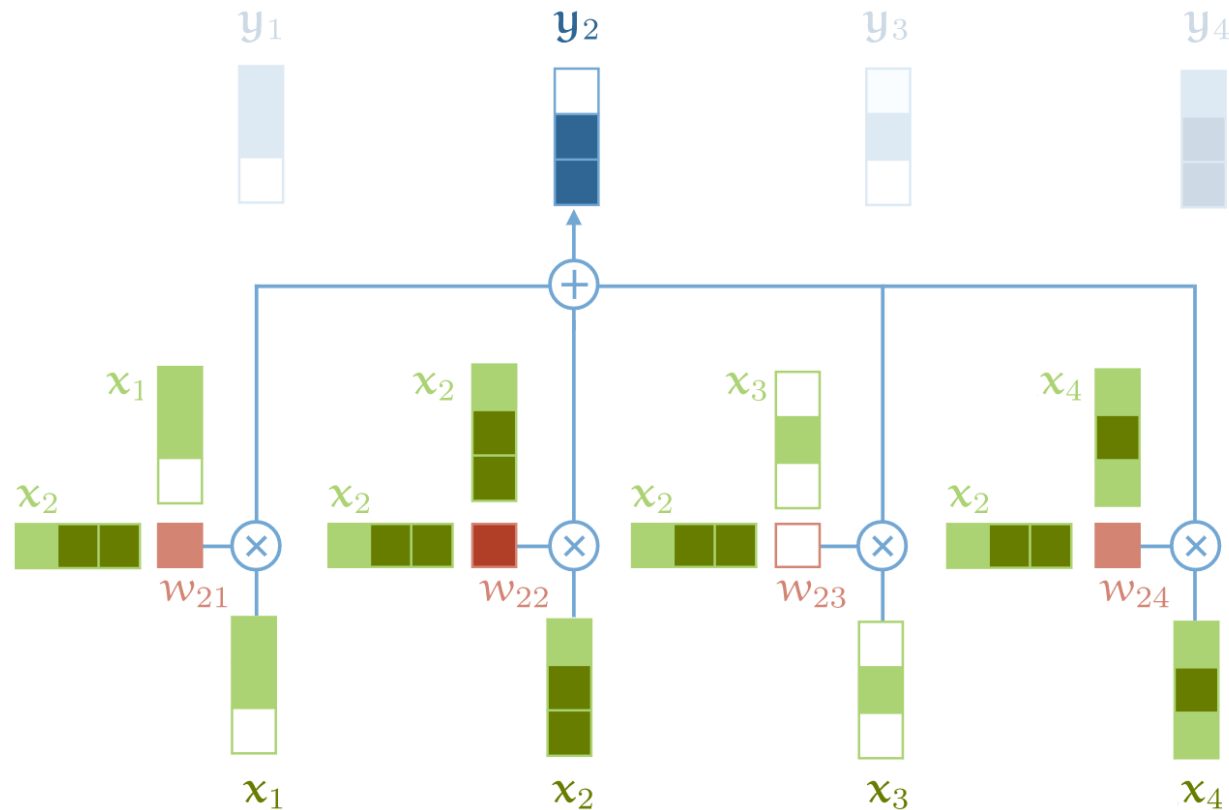
Self-Attention Operation

Sequence-to-sequence operation

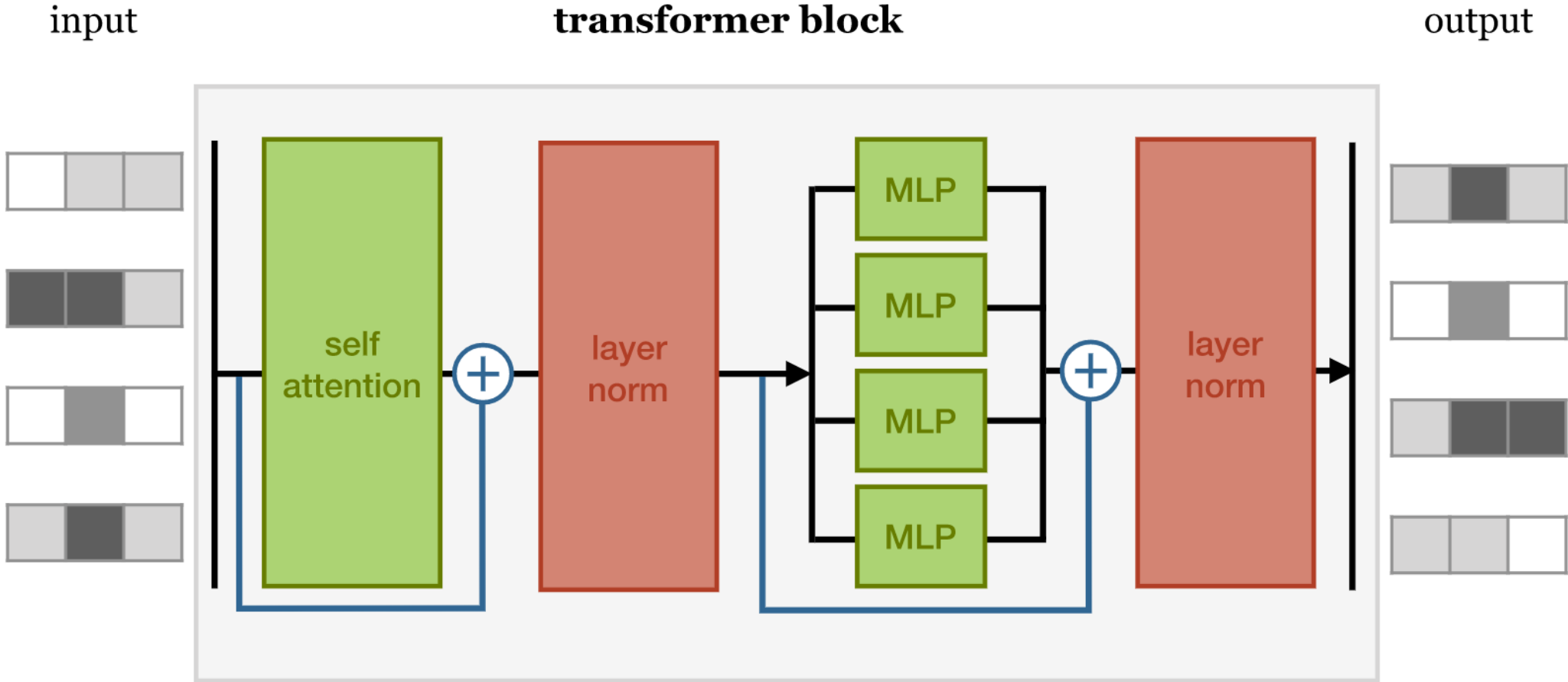


Self-Attention Operation

Outputs are **weighted average** of all input vectors



Transformer Block



Bloem (2019)

Transformer Model

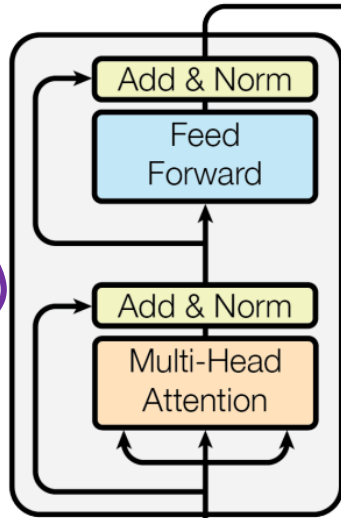
Transformer Encoder Block

Nx

Positional Encoding

Input Embedding

Inputs



Output Probabilities

Softmax

Linear

Add & Norm

Feed Forward

Add & Norm

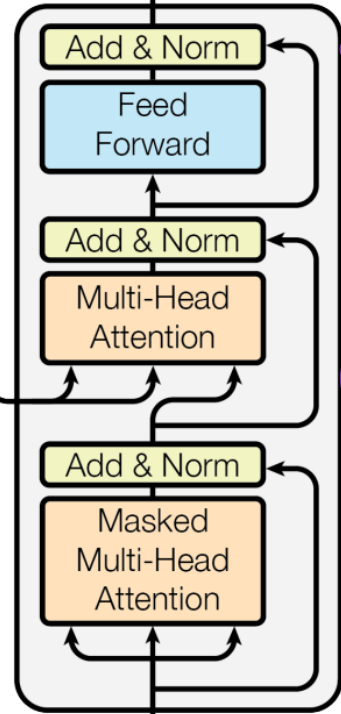
Multi-Head Attention

Add & Norm

Masked Multi-Head Attention

Output Embedding

Outputs (shifted right)



Nx

Positional Encoding

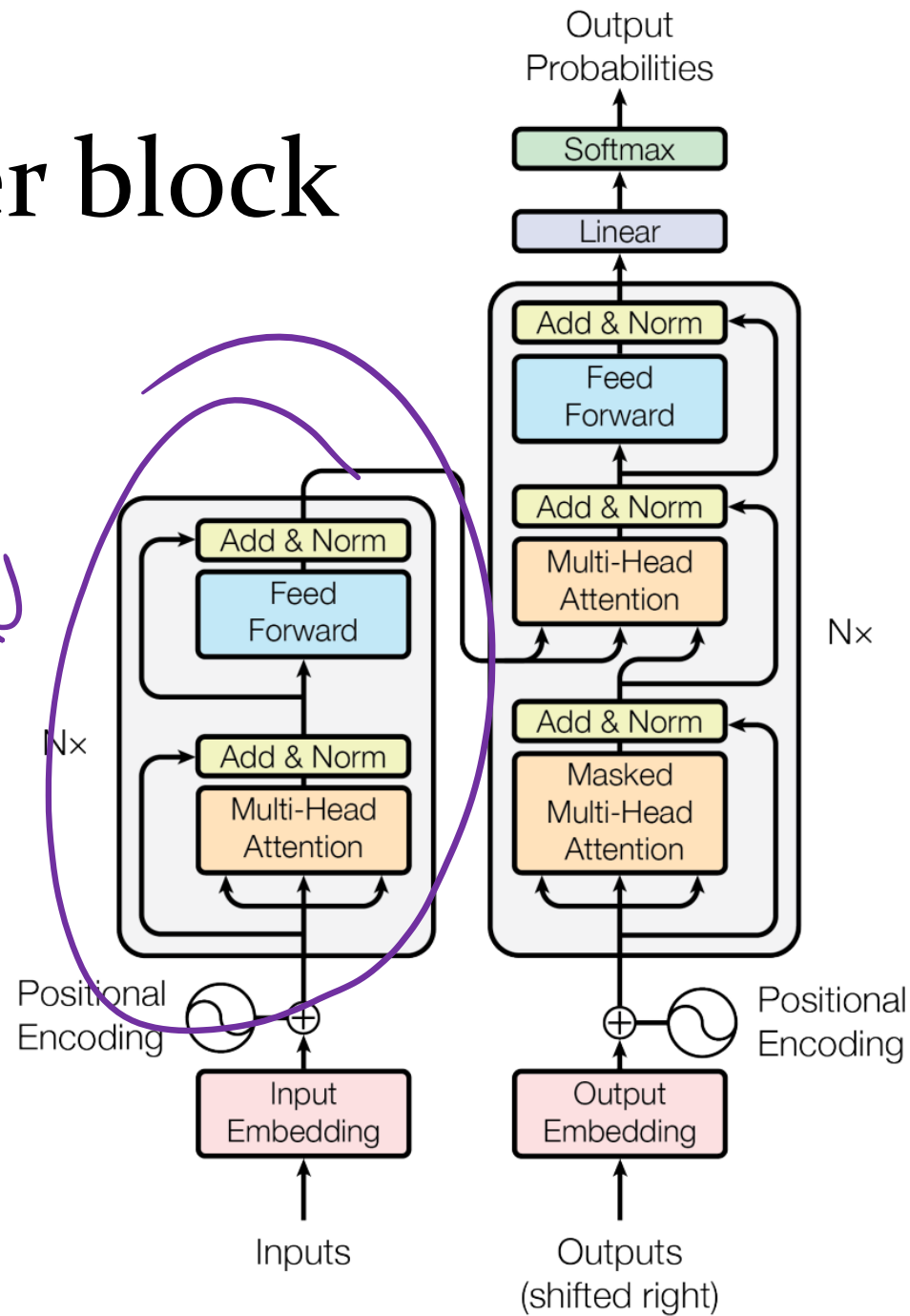
Decoder Block



...and back to BERT

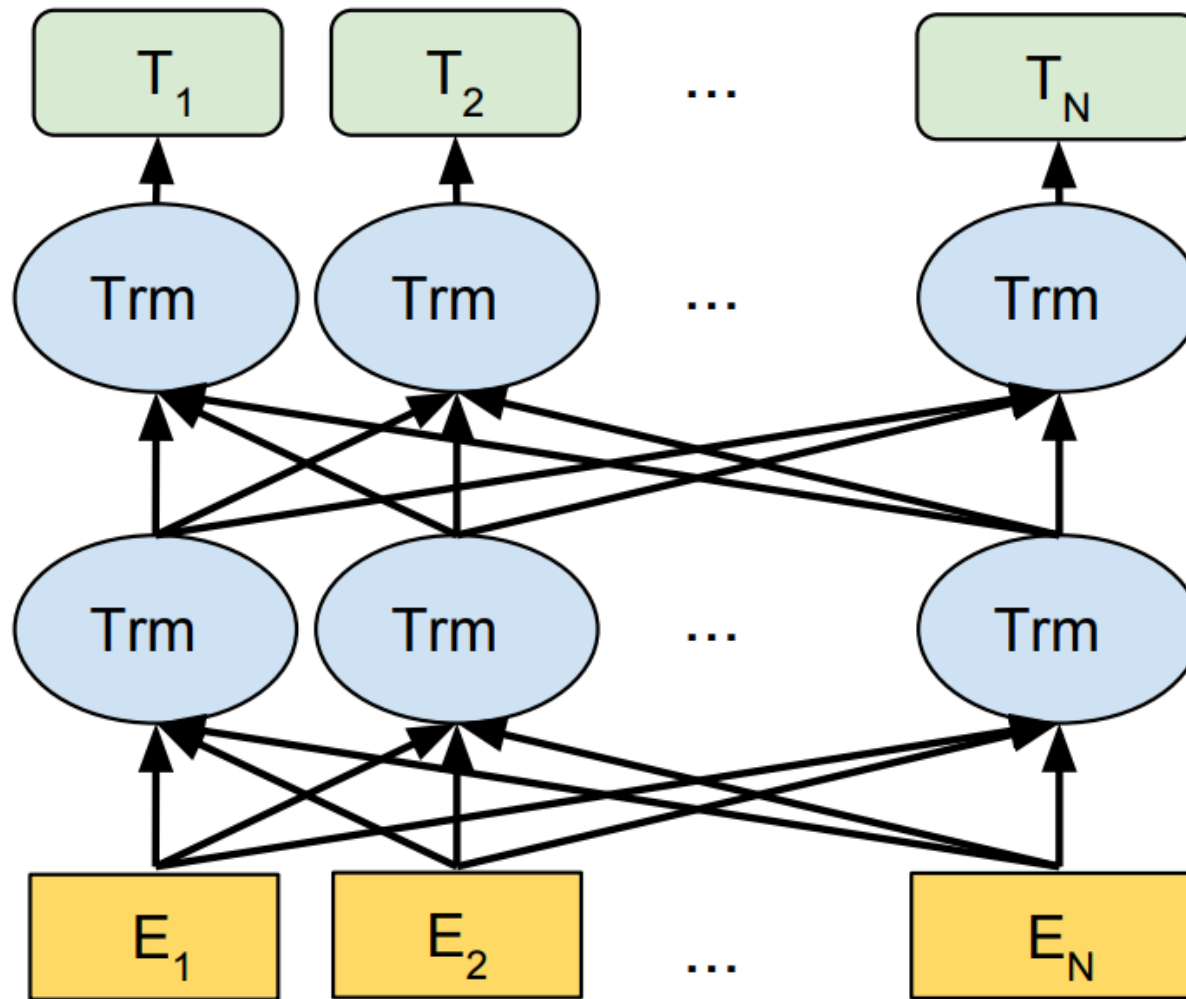
Uses encoder block

blocks

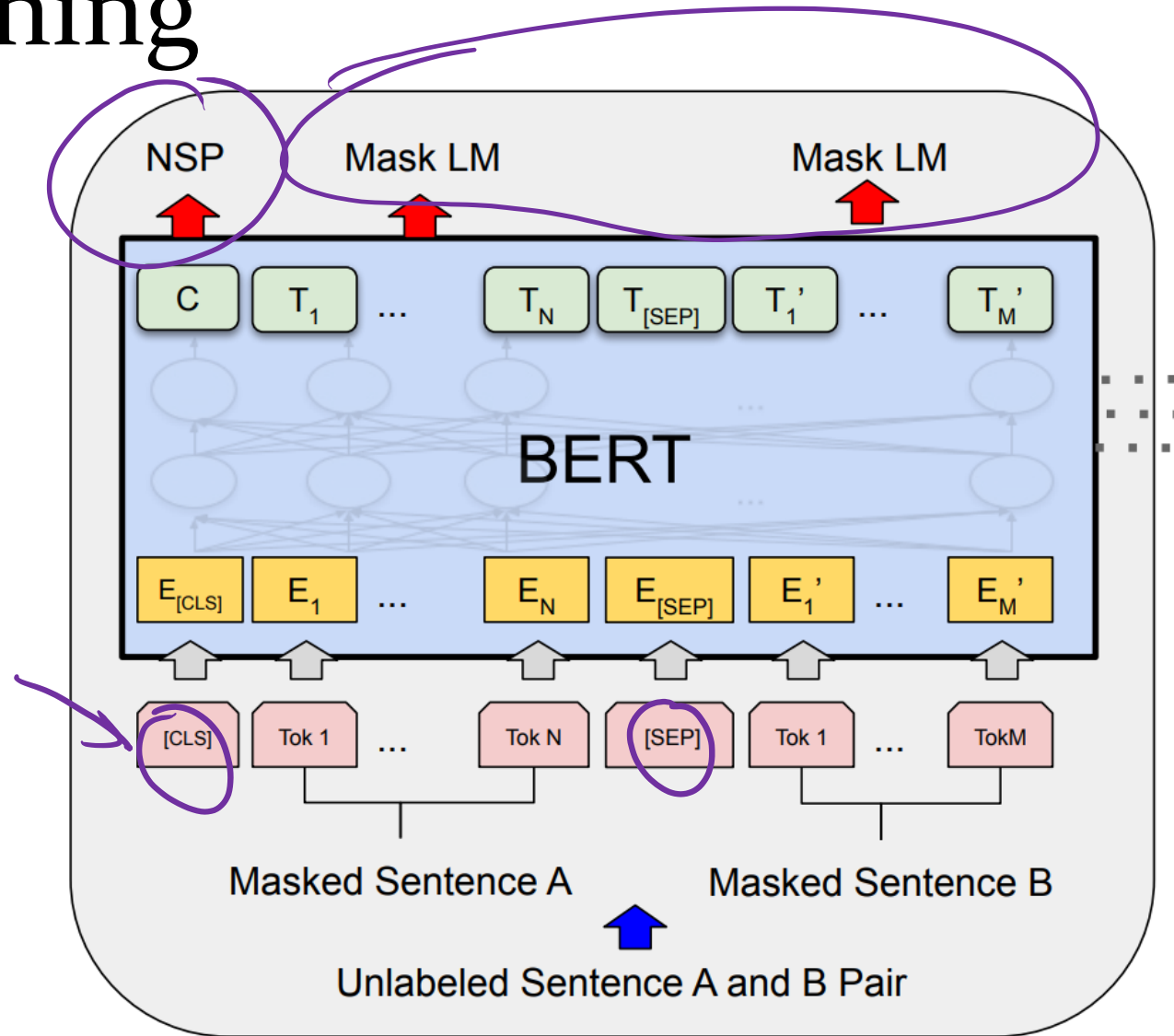


BERT

12 layers



Pre-Training



Pre-Training Tasks

1. Masked Language Model (MLM)
2. Next Sentence Prediction (NSP)

Masked Language Model

Setting: Randomly mask some tokens of the input

Objective: Predict the original word types of each masked token based solely on its context

Masked Language Model Procedure

Apply procedure to 15% of tokens

- 80% of the time: Replace the word with the [MASK] token
- 10% of the time: Replace the word with a random word
- 10% of the time: Keep the word unchanged

Masked Language Model Procedure

Example: my dog is hairy

- 80% of the time: Replace the word with the [MASK] token
my dog is [MASK]
- 10% of the time: Replace the word with a random word
my dog is apple
- 10% of the time: Keep the word unchanged
my dog is hairy

Next Sentence Prediction

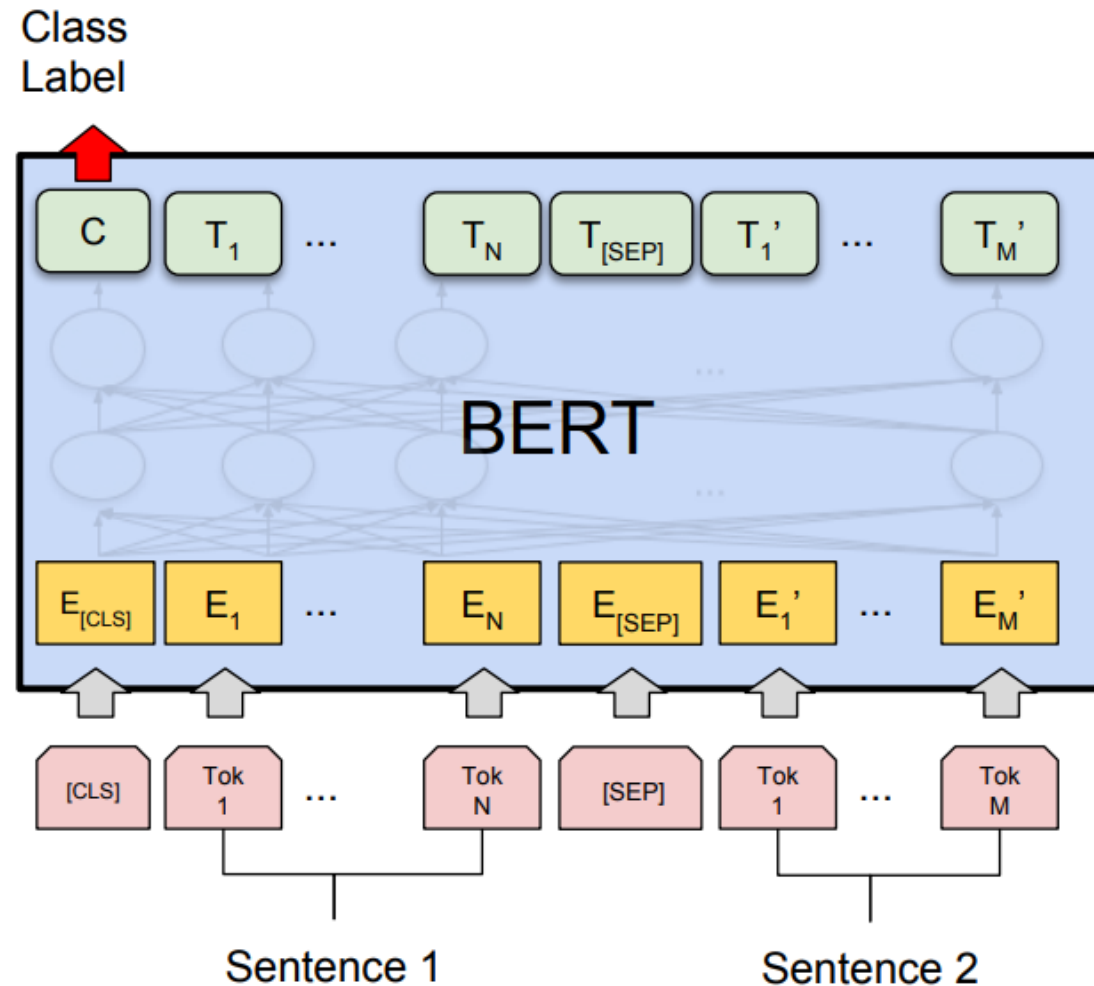
Input = [CLS] the man went to [MASK] store [SEP]
 he bought a gallon [MASK] milk [SEP]

Label = IsNext

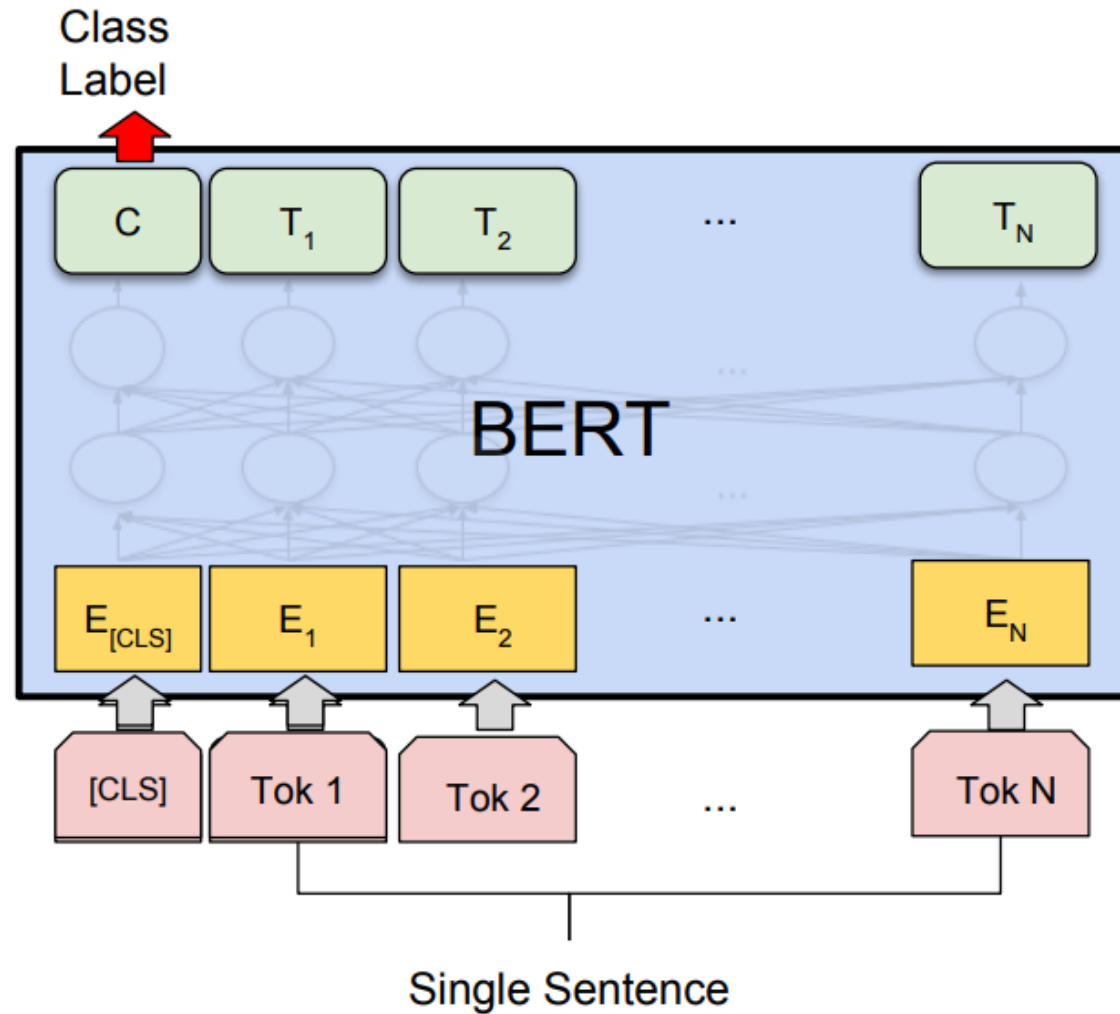
Input = [CLS] the man [MASK] to the store [SEP]
 penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

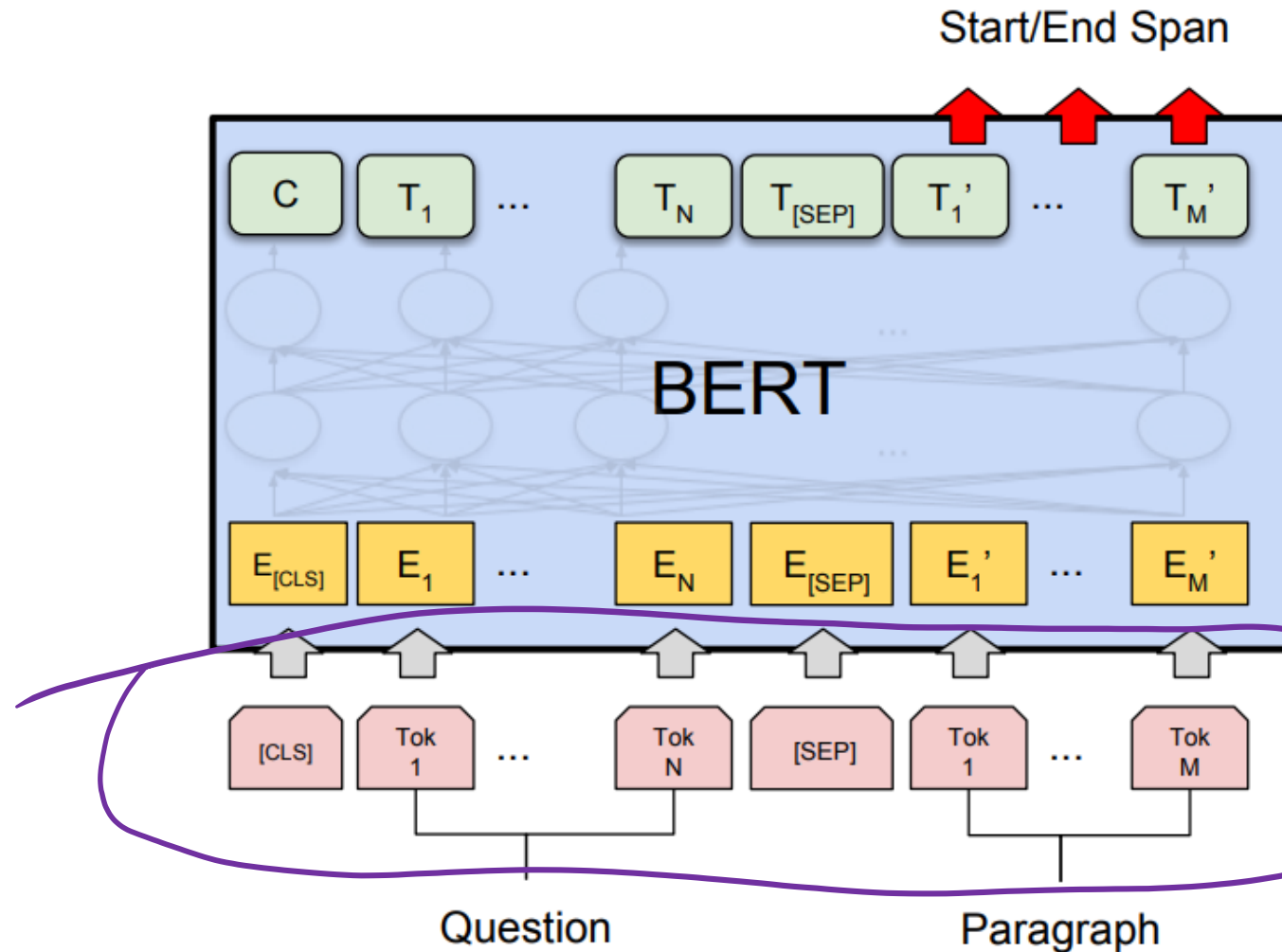
Fine-Tuning: Sentence Pair Classification



Fine-Tuning: Single Sentence Classification

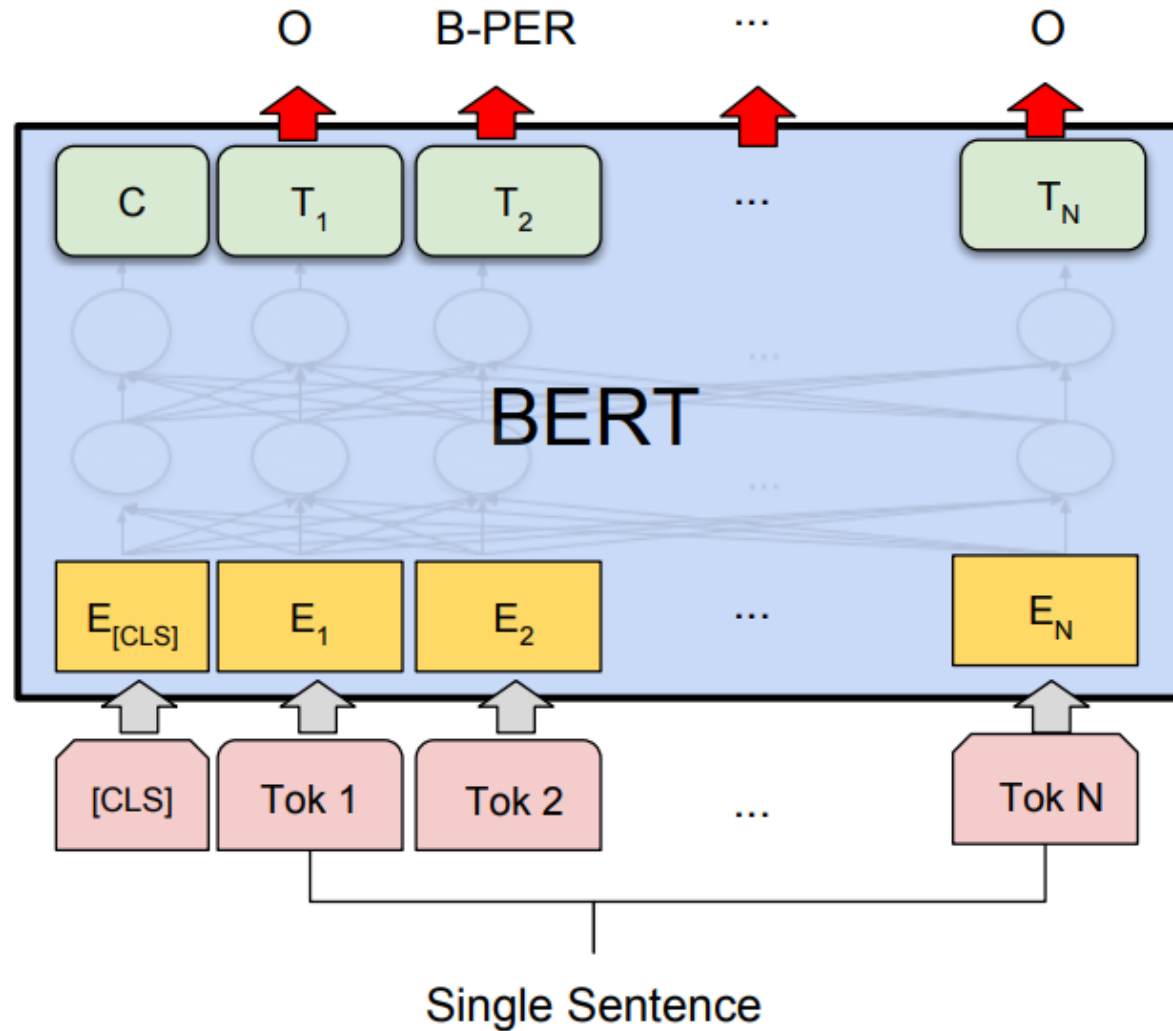


Fine-Tuning: Question Answering



512

Fine-Tuning: Single Sentence Tagging



Huge gains for many tasks

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Still visible in leaderboards today

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-
1	ERNIE Team - Baidu	ERNIE	↗	91.1	75.5	97.8	93.9/91.8	93.0/92.6	75.2/90.9	92.3	9
2	AliceMind & DURL	StructBERT + CLEVER	↗	91.0	75.3	97.7	93.9/91.9	93.5/93.1	75.6/90.8	91.7	9
3	liangzhu ge	DeBERTa + CLEVER		90.9	73.9	97.5	92.8/90.4	93.2/92.9	76.4/90.9	92.1	9
4	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4	↗	90.8	71.5	97.5	94.0/92.0	92.9/92.6	76.2/90.8	91.9	9
5	HFL iFLYTEK	MacALBERT - DKM		90.7	74.8	97.0	94.5/92.6	92.8/92.6	74.7/90.6	91.3	9
+	PING-AN Omni-Sinitic	ALBERT - DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6	9
7	T5 Team - Google	T5	↗	90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	9
8	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART	↗	89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	9
+	Huawei Noah's Ark Lab	NEZHA-Large		89.8	71.7	97.3	93.3/91.0	92.4/91.9	75.2/90.7	91.5	9
+	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)	↗	89.7	70.5	97.5	93.4/91.2	92.6/92.3	75.4/90.7	91.4	9

GLUE leaderboard

