



COMP 336 | Natural Language Processing

Lecture 13: Pre-training and large language
models (LLMs)

Spring 2025

Announcements

- Get started on assignment 2 ASAP!
 - Join [#assignment-2](#) Slack channel for discussion

Lecture plan

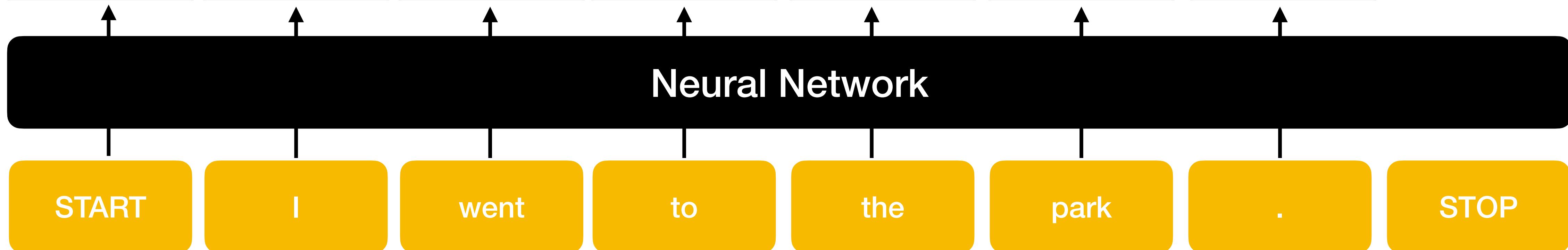
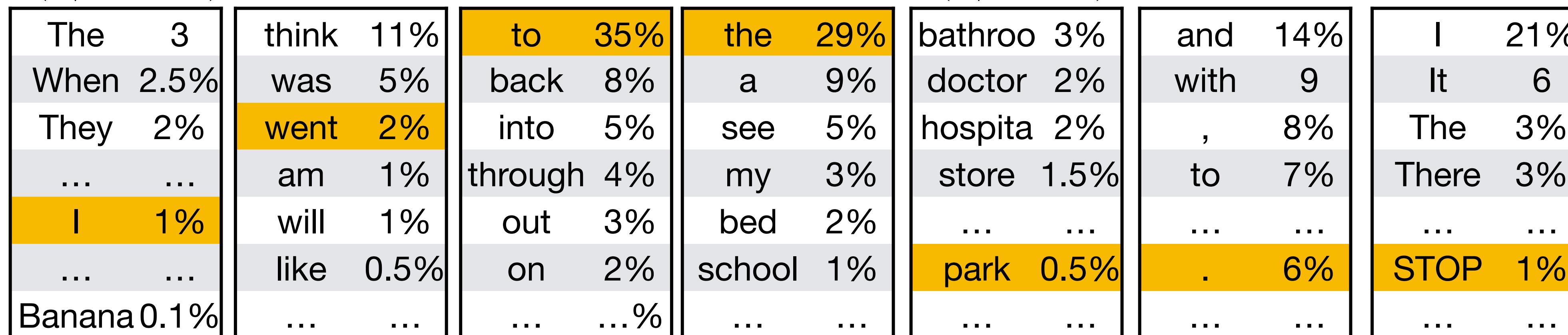
- Neural language models: recap
- Traditional to modern NLP
 - Traditional learning paradigm
 - Supervised training/fine-tuning only, NO pre-training
 - Modern learning paradigm
 - Pretrain + fine-tuning, pretrain + prompting/in-context learning
- Pretraining overview: BERT, T5, GPT

Neural language models: recap

Neural language models: overview

- **Input:** sequences of words (or tokens)
- **Output:** probability distribution over the next word (token)

$$p(x|\text{START}) p(x|\text{START I}) p(x|\dots \text{went}) \quad p(x|\dots \text{to}) \quad p(x|\dots \text{the}) \quad p(x|\dots \text{park}) \quad p(x|\text{START I went to the park.})$$

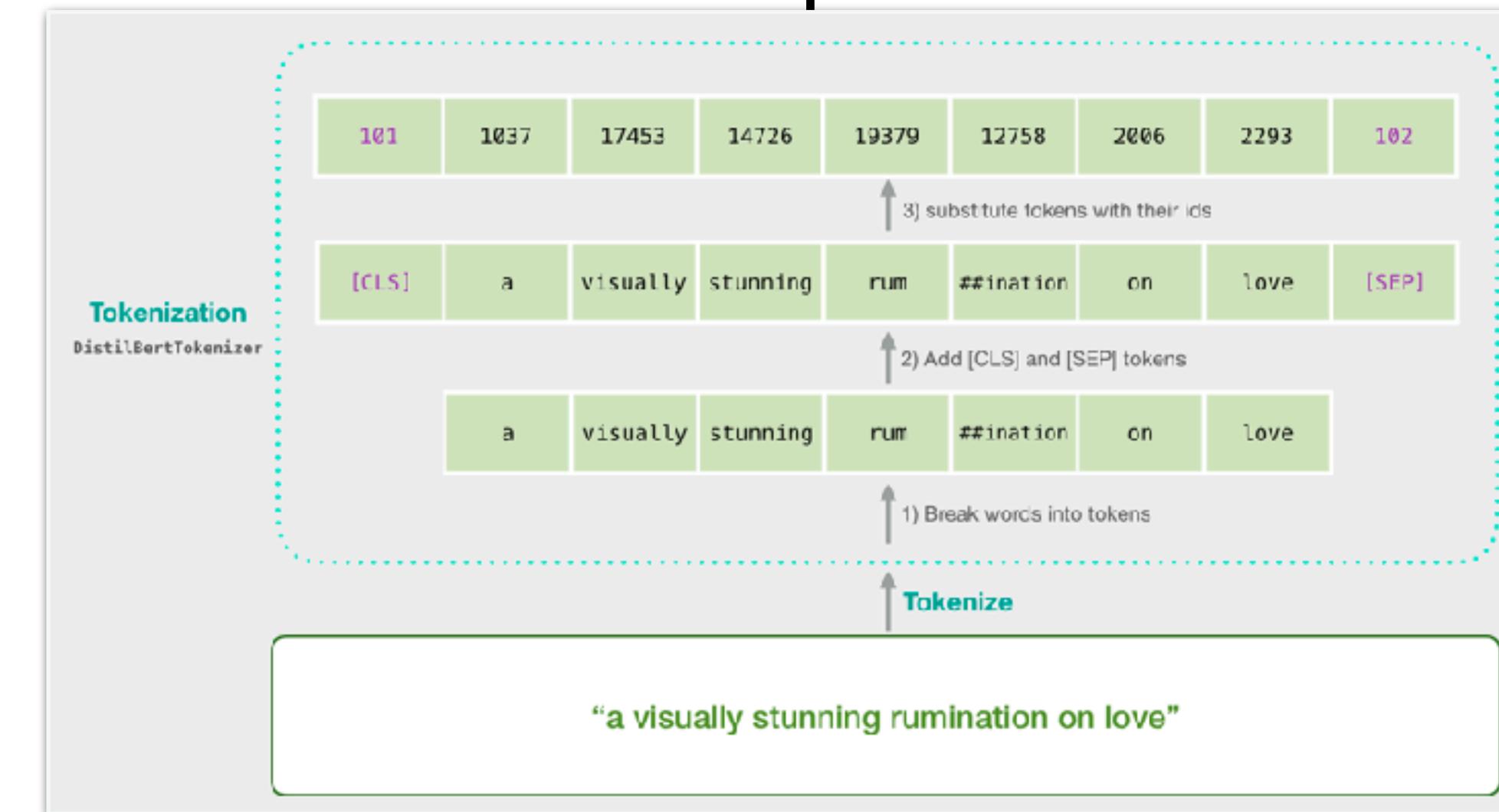


Neural language models: tokenization

$p(x|\text{START})p(x|\text{START I})p(x|\dots\text{went}) \quad p(x|\dots\text{to}) \quad p(x|\dots\text{the}) \quad p(x|\dots\text{park}) \quad p(x|\text{START I went to the park.})$



Tokenization:



Call me Ishmael. Some years ago—never mind how long precisely—having
little or no money in my purse, and nothing particular to interest me on
shore, I thought I would sail about a little and see the watery part of
the world. It is a way I have of driving off the spleen and regulating
the circulation. Whenever I find myself growing grim about the mouth;
whenever it is a damp, drizzling November in my soul; whenever I find
myself involuntarily pausing before coffin warehouses, and bringing up
the rear of every funeral I meet; and especially whenever my hypos get
such an upper hand of me, that it requires a strong moral principle to
prevent me from deliberately stepping into the street, and methodically
knocking people's hats off—then, I account it high time to get to sea
as soon as I can. This is my substitute for pistol and ball. With a
philosophical flourish Cato throws himself upon his sword; I quietly
take to the ship. There is nothing surprising in this. If they but knew
it, almost all men in their degree, some time or other, cherish very
nearly the same feelings towards the ocean with me.

TEXT TOKEN IDS

START

I

went

to

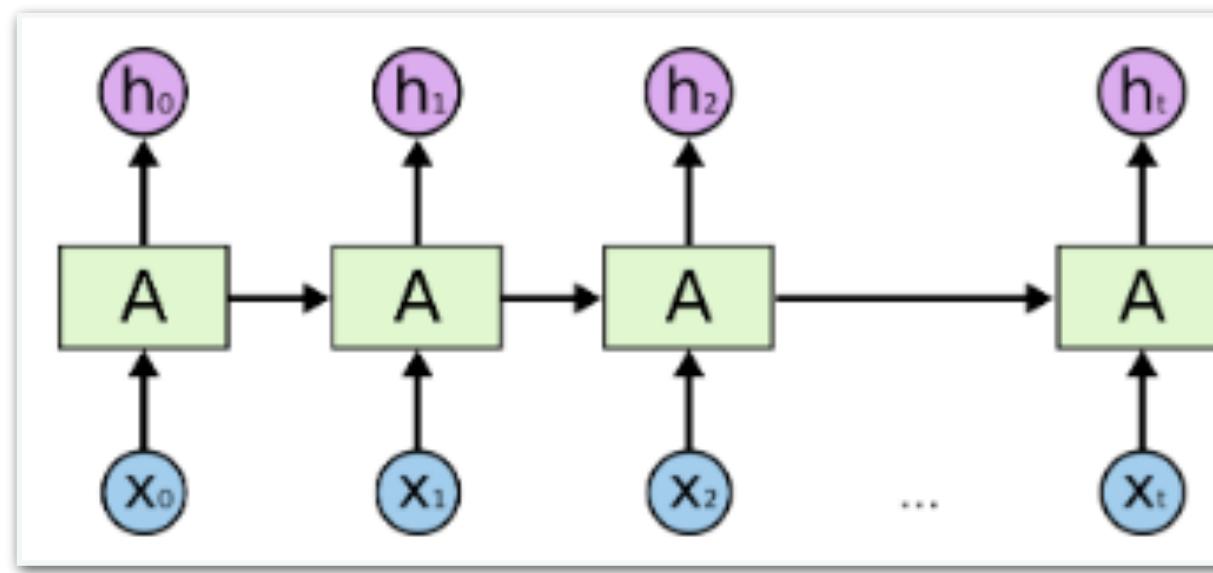
the

park

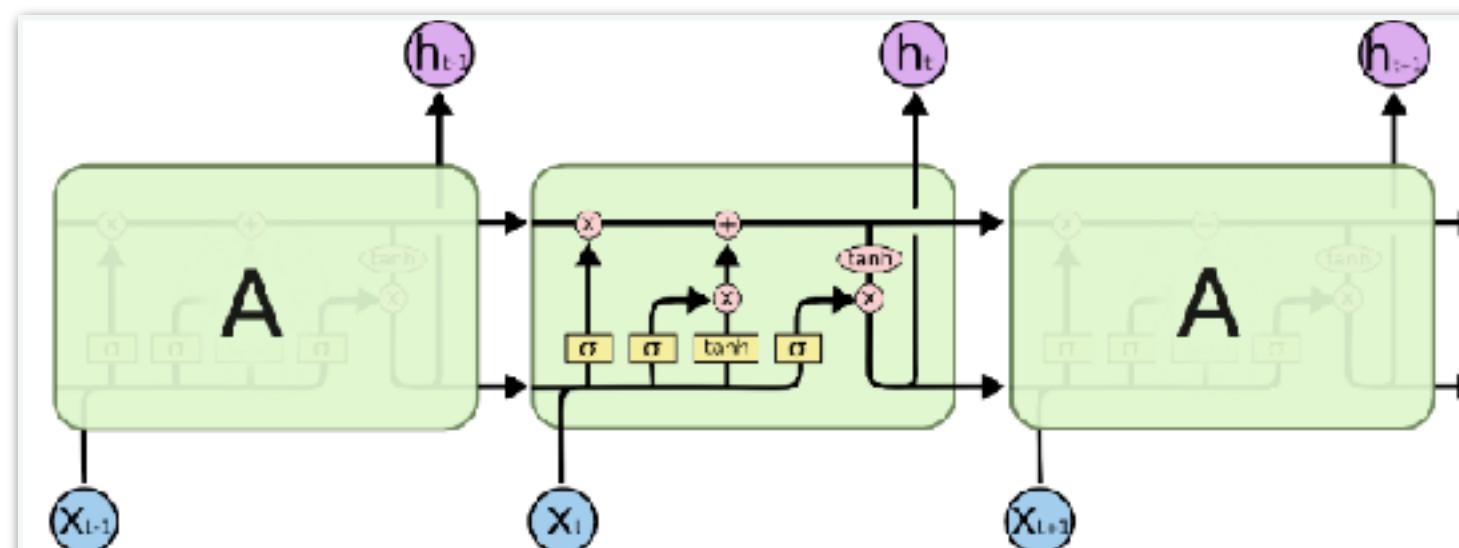
.

STOP

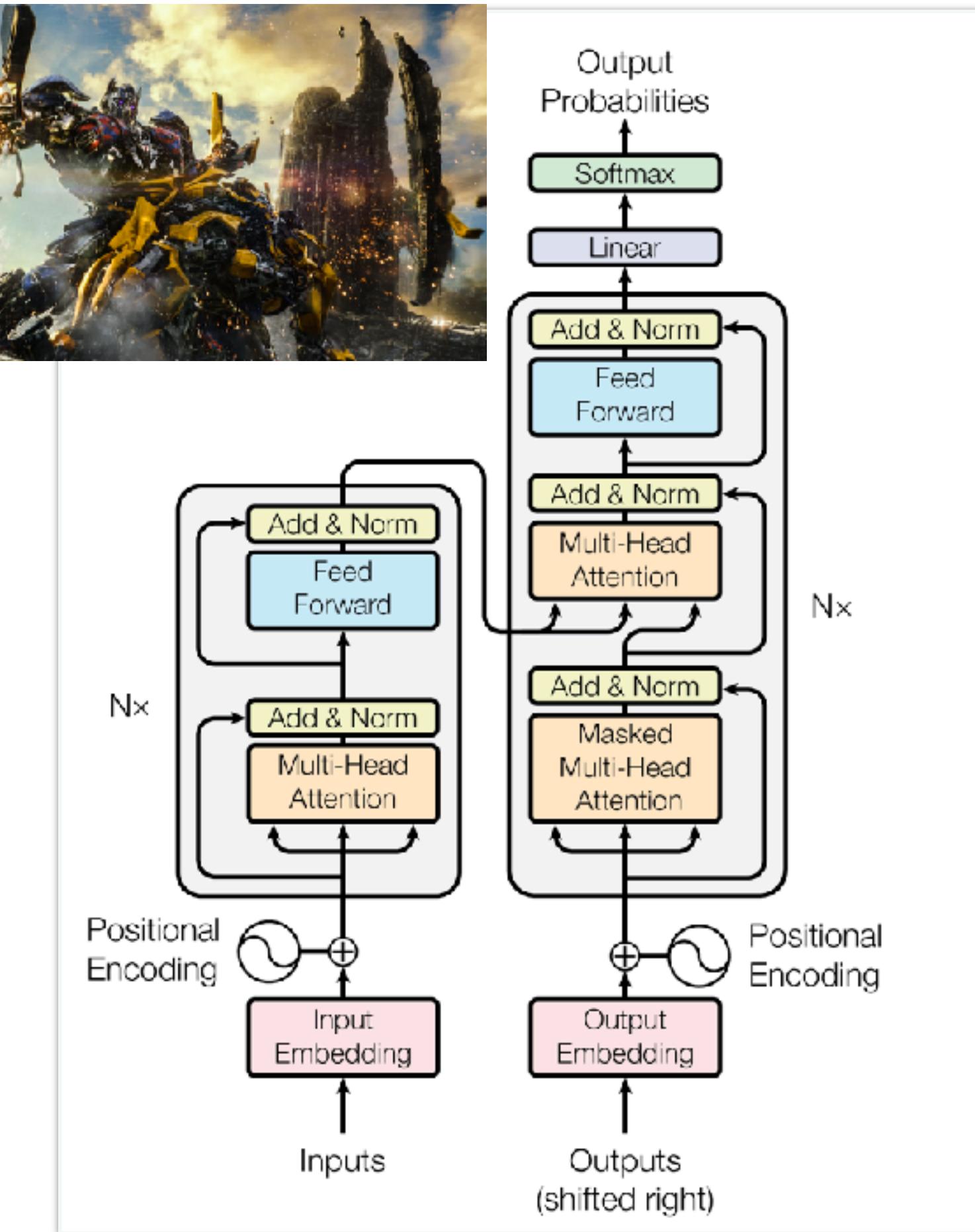
Neural language models: neural networks



RNNs



LSTM



Transformers

Traditional to modern NLP

N-gram language models



Neural language models: BERT, GPT

Traditional models: Naive Bayes

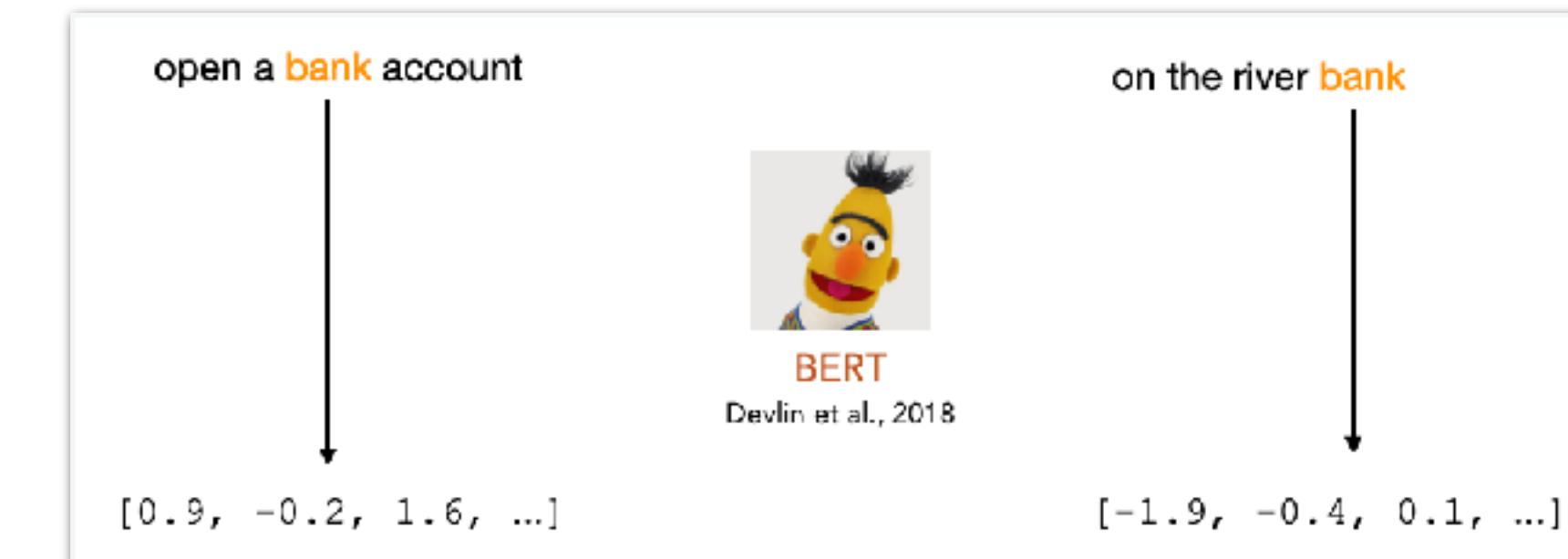
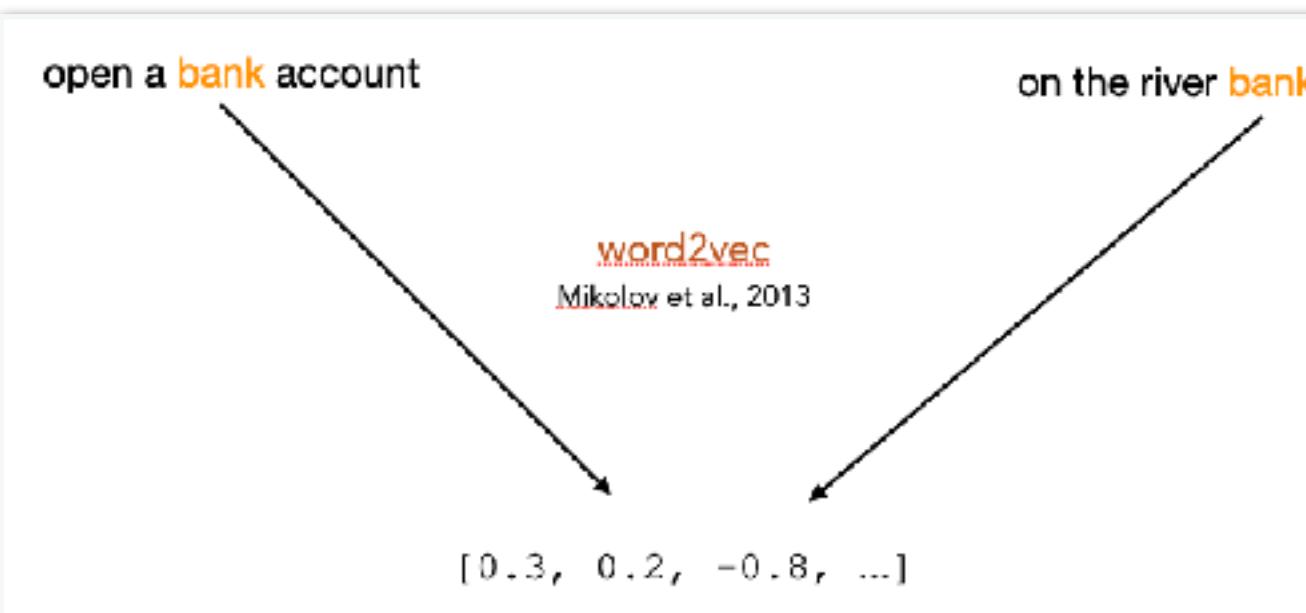


Neural models: Transformers

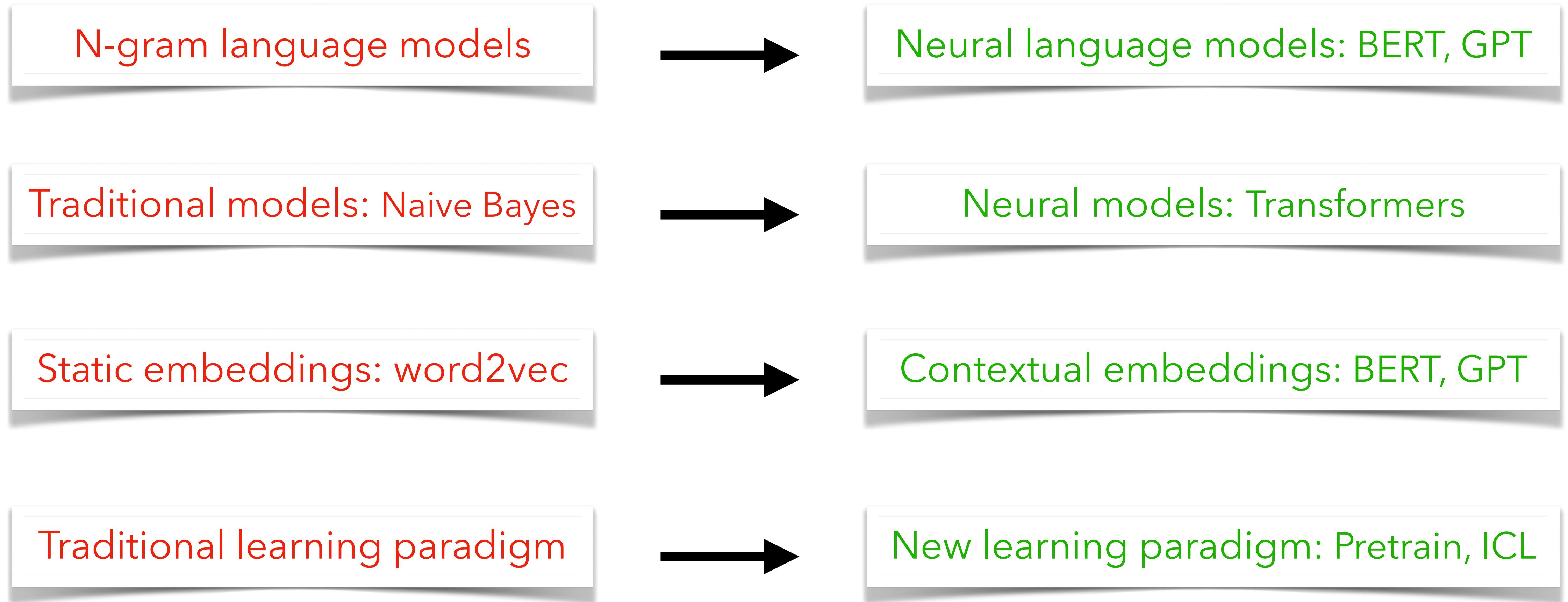
Static embeddings: word2vec



Contextual embeddings: BERT, GPT



Traditional to modern NLP: training paradigm



Question: How to train and use neural language models for different NLP tasks?

Training models for NLP tasks

Foundational Technologies

- Language Modeling
- Part-of-speech Tagging
- Syntactic Parsing
- Dependency Parsing
- Named Entity recognition
- Coreference resolution
- Word Sense Disambiguation
- Semantic Role Labelling
-

High-Level Tasks and Applications

- Sentiment Analysis
- Information Extraction
- Machine Translation
- Question Answering
- Semantic Parsing
- Summarization
- Dialogue systems
- Language and Vision
- Data-to-Text Generation
-

Input X	Output Y	Task
Text	Label	Text Classification (e.g., Sentiment Analysis)
Text	Linguistic Structure	Structured Prediction (e.g., Part-of-Speech Tagging)
Text	Text	Text Generation (e.g., Translation, Summarization)

Example: Training Transformers for sentiment analysis

Task:



Model:



Transformers

Traditional learning paradigm

- **Supervised training/fine-tuning only, NO pre-training**
 - Collect (x, y) task training pairs

Data:

sentence	label
a stirring , funny and finally transporting re imagining of beauty and the beast and 1930s horror films	1
apparently reassembled from the cutting room floor of any given daytime soap	0
they presume their audience won't sit still for a sociology lesson	0
this is a visually stunning rumination on love , memory , history and the war between art and commerce	1
jonathan parker 's bartleby should have been the be all end all of the modern office anomie films	1

Traditional learning paradigm

- **Supervised training/fine-tuning only, NO pre-training**
 - Collect (x, y) task training pairs
 - Randomly initialize your models $f(x)$ (e.g., vanilla Transformers)

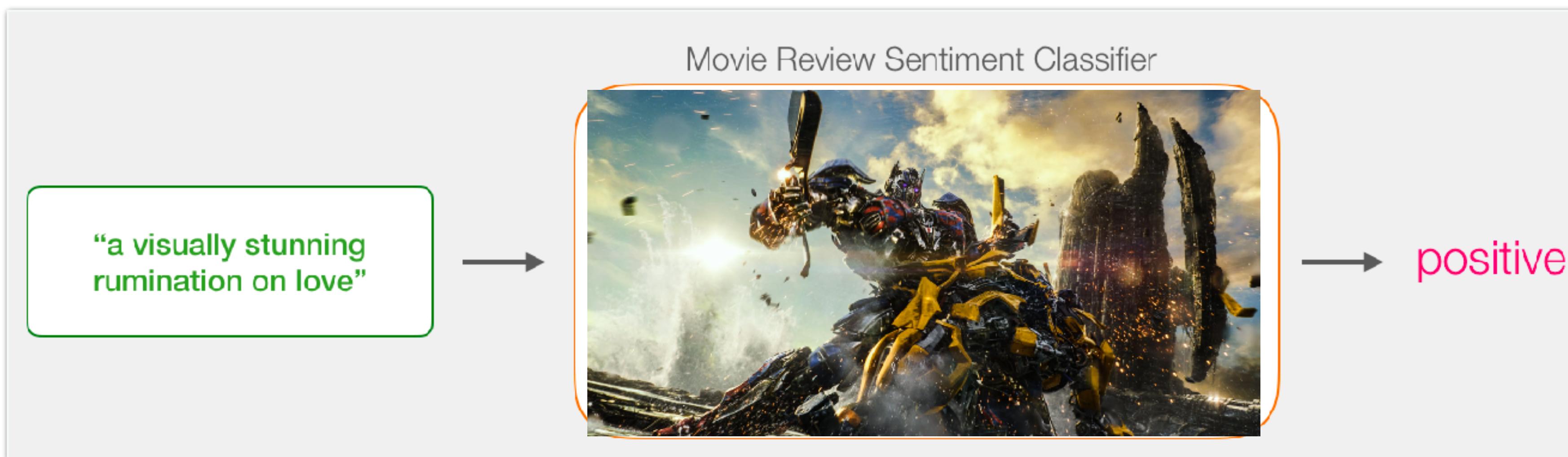


Randomly initialized Transformers
NO pretrained parameters used

Traditional learning paradigm

- **Supervised training/fine-tuning only, NO pre-training**

- Collect (x, y) task training pairs
- Randomly initialize your models $f(x)$ (e.g., vanilla Transformers)
- Train $f(x)$ on (x, y) pairs



Traditional learning paradigm

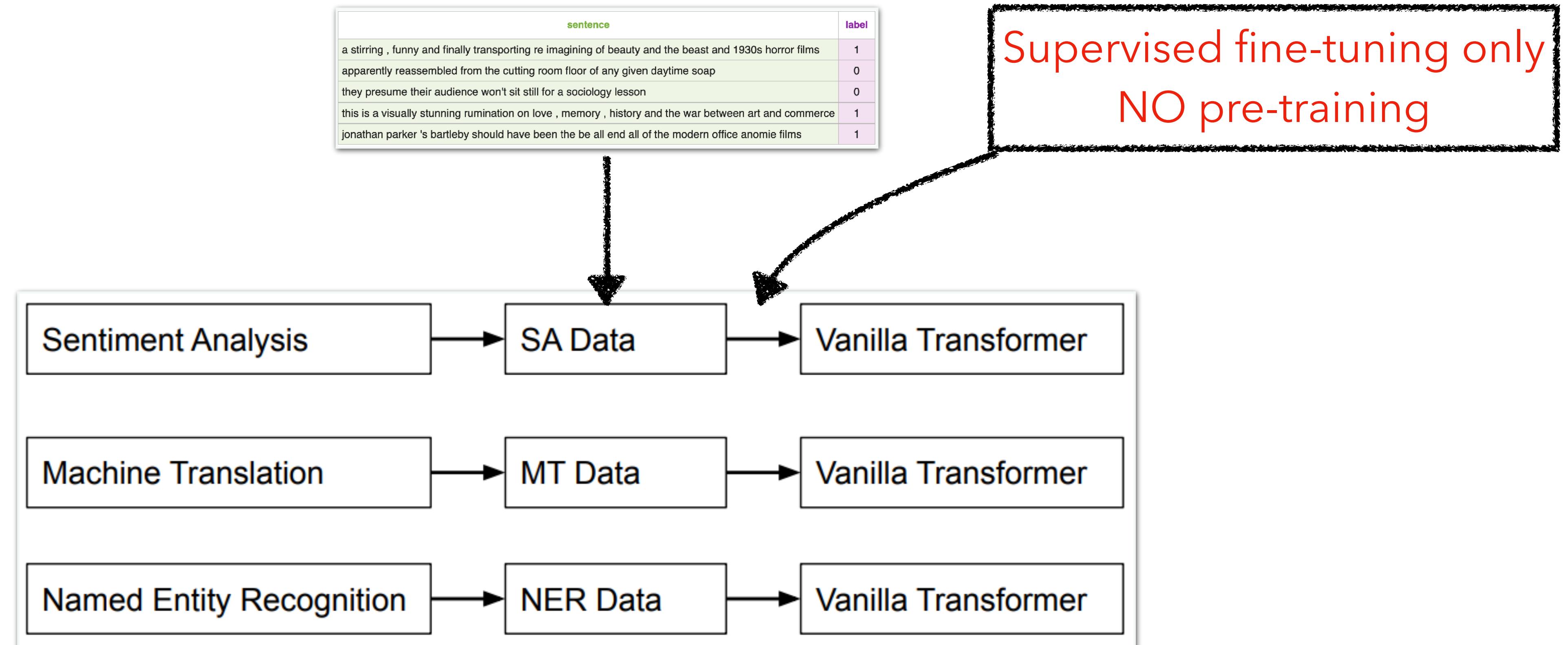
- **Supervised training/fine-tuning only, NO pre-training**
 - Collect (x, y) task training pairs
 - Randomly initialize your models $f(x)$ (e.g., vanilla Transformers)
 - Train $f(x)$ on (x, y) pairs



Then you get a trained Transformers **ONLY** for sentiment analysis
The model can be: NB, LR, RNNs, LSTM too

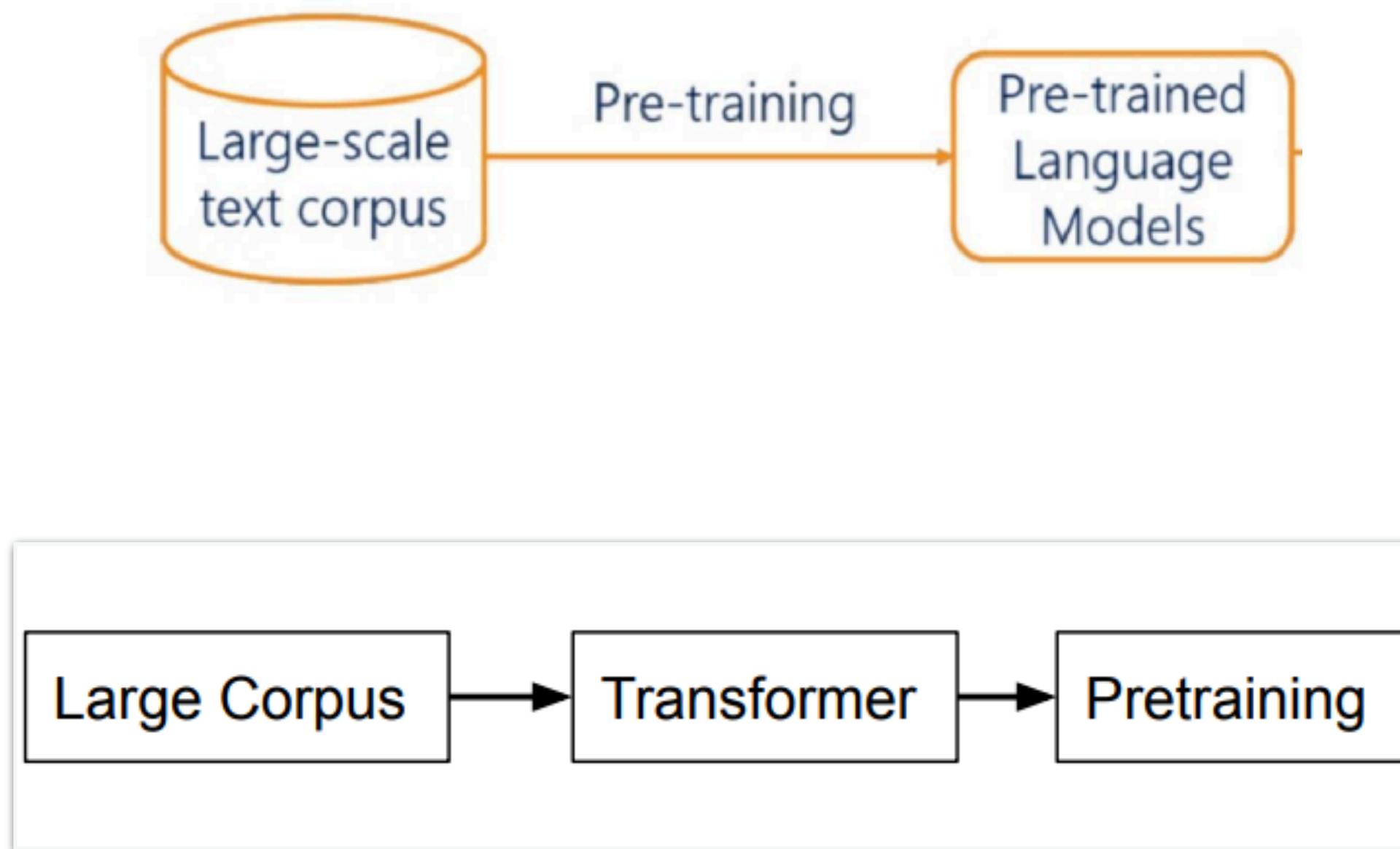
Traditional learning paradigm

- **Supervised training/fine-tuning only, NO pre-training**
 - Train Transformer or other models separately for each task



Modern learning paradigm

- **Pre-training + supervised training/fine-tuning**
 - First train Transformer using a lot of general text using unsupervised learning. This is called **pretraining**.



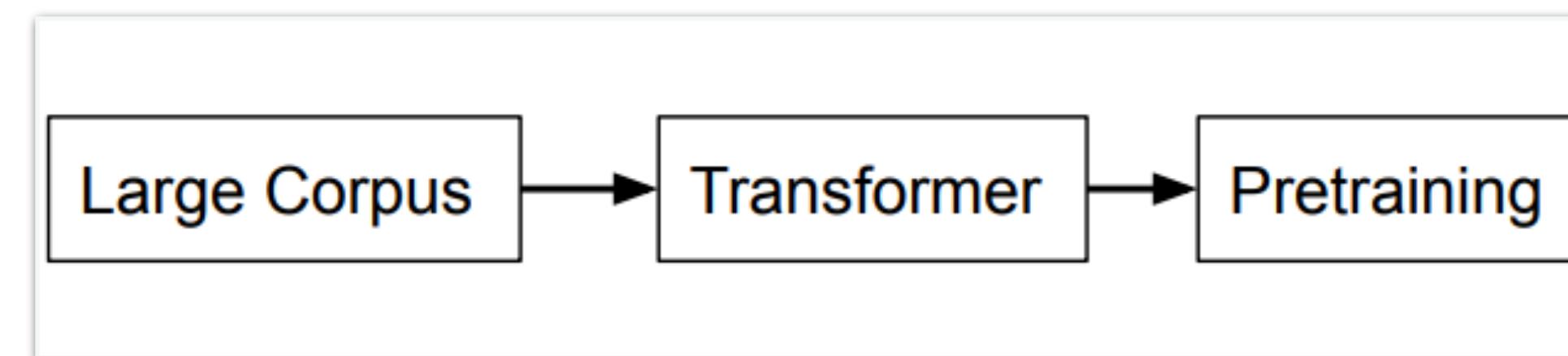
Modern learning paradigm

- **Pre-training + supervised training/fine-tuning**

- First train Transformer using a lot of general text using unsupervised learning. This is called **pretraining**.



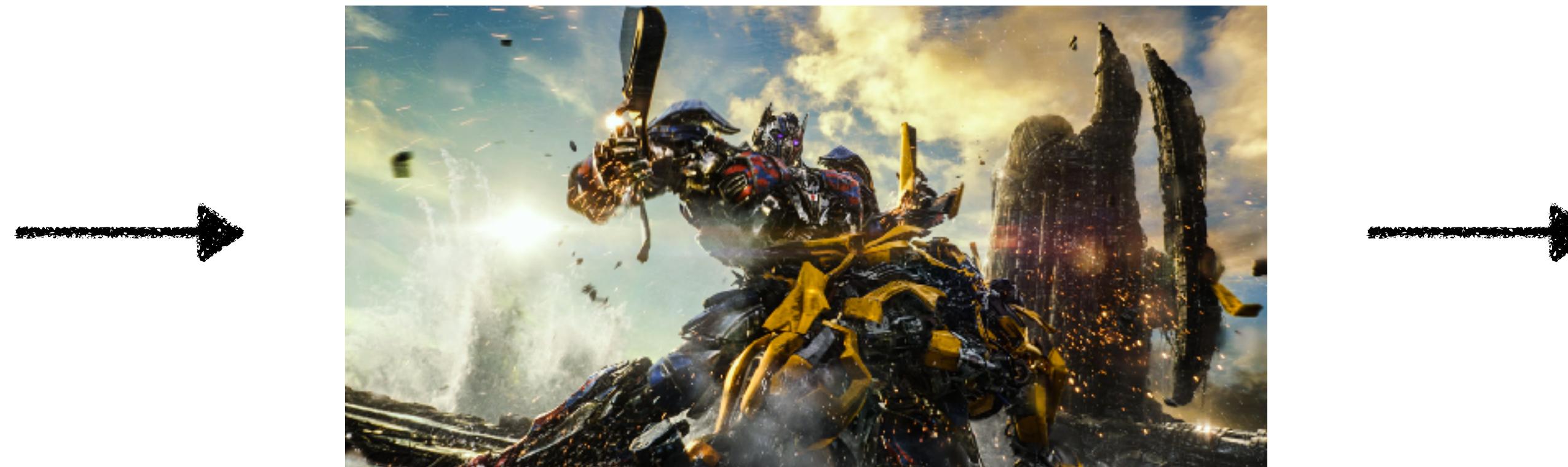
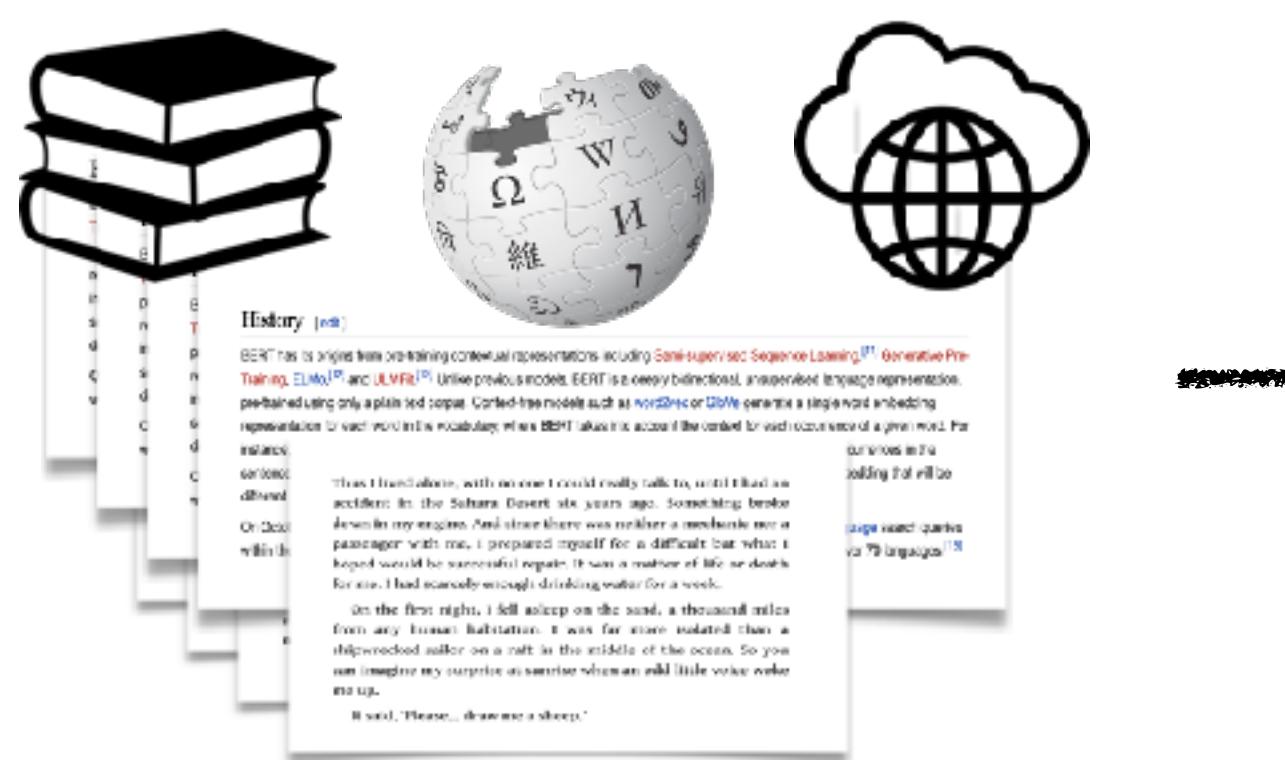
1. Randomly initialized Transformers



Modern learning paradigm

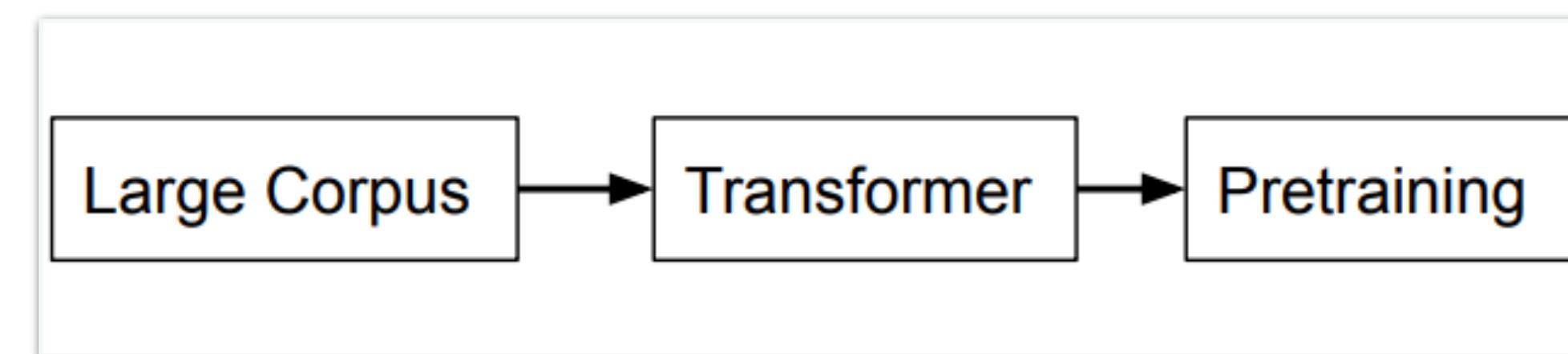
- **Pre-training + supervised training/fine-tuning**

- First train Transformer using a lot of general text using unsupervised learning. This is called **pretraining**.



**general
training objectives?**

2. Pre-train Transformers on large text data with **general objectives**

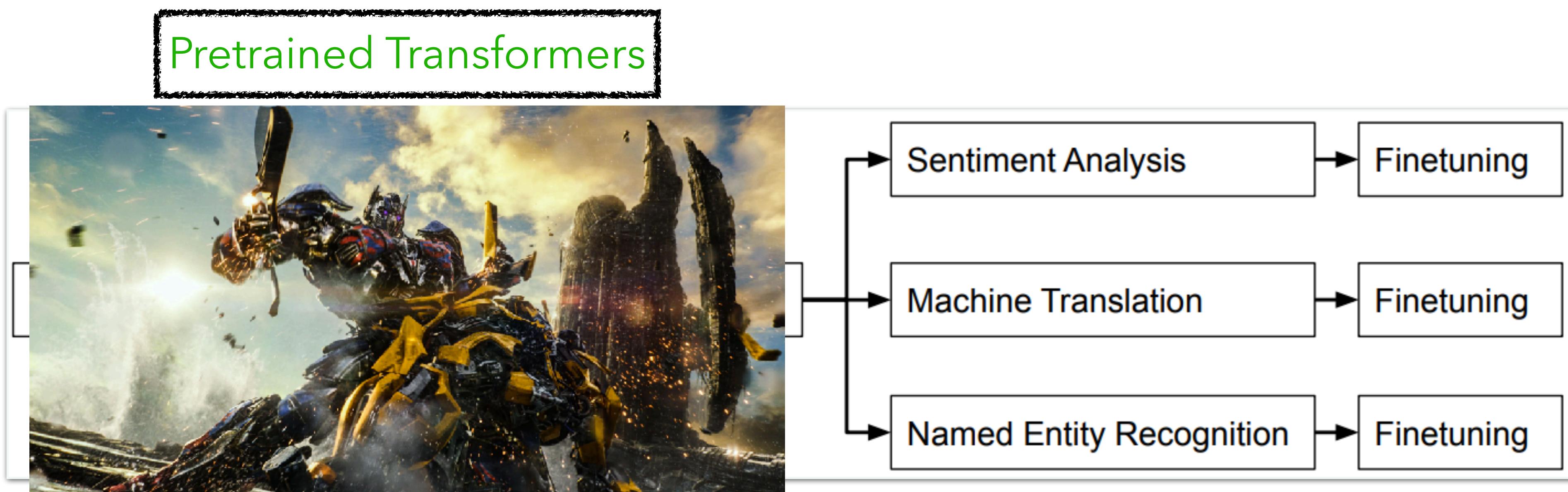


>2M GPU hours

Modern learning paradigm

- **Pre-training + supervised training/fine-tuning**

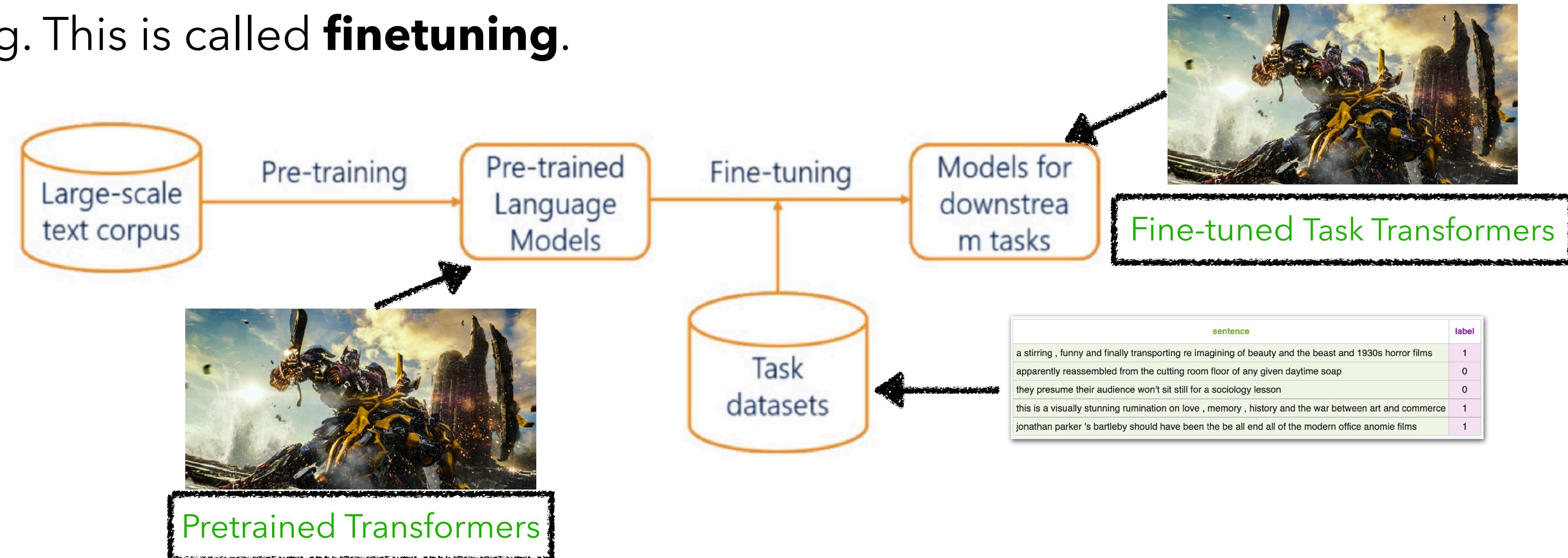
- First train Transformer using a lot of general text using unsupervised learning. This is called **pretraining**.
- Then train the pretrained Transformer for a specific task using supervised learning. This is called **finetuning**.



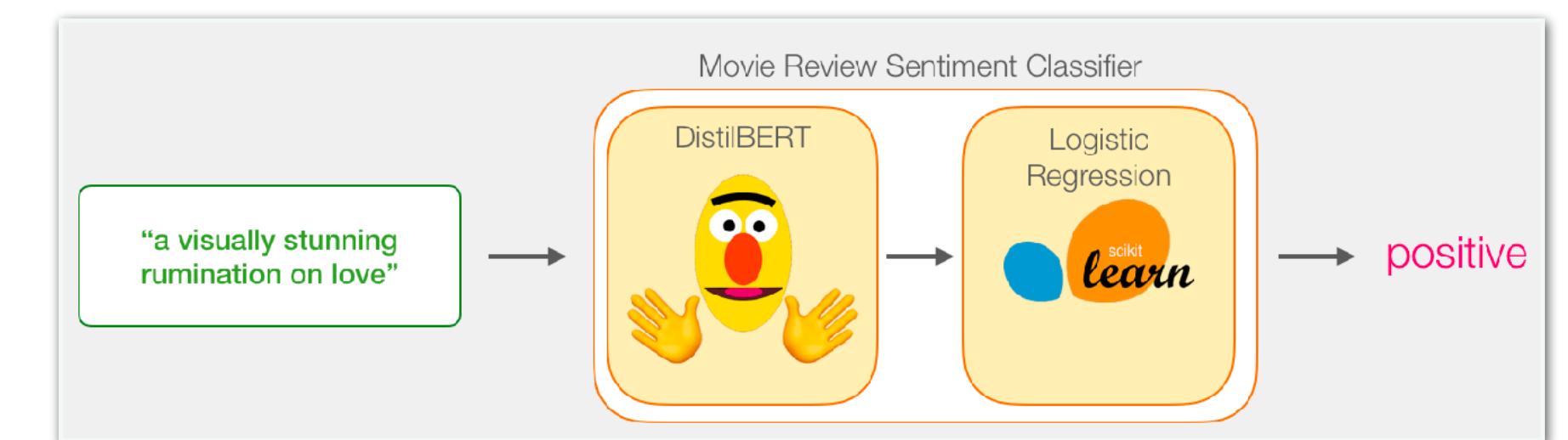
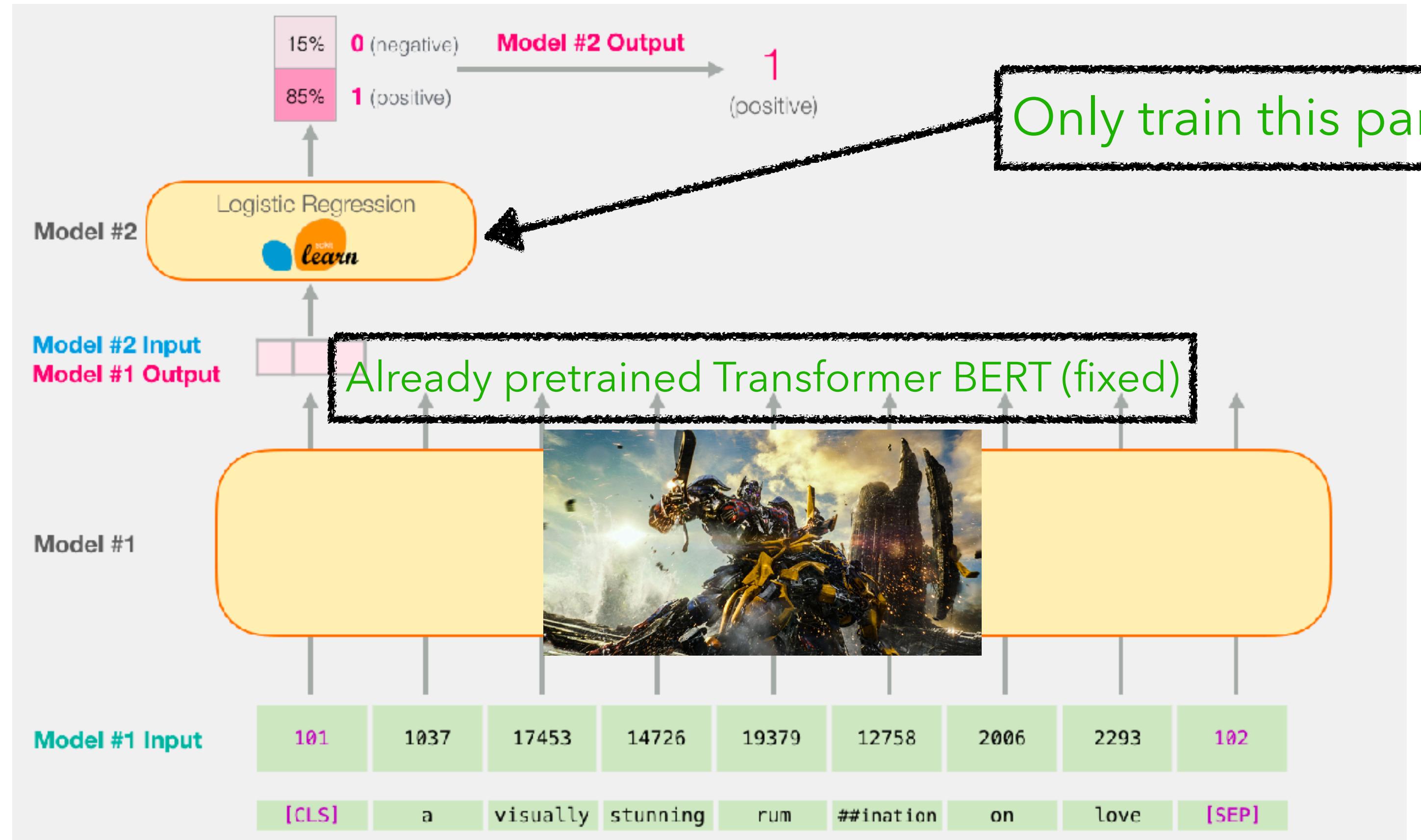
Modern learning paradigm

- **Pre-training + supervised training/fine-tuning**

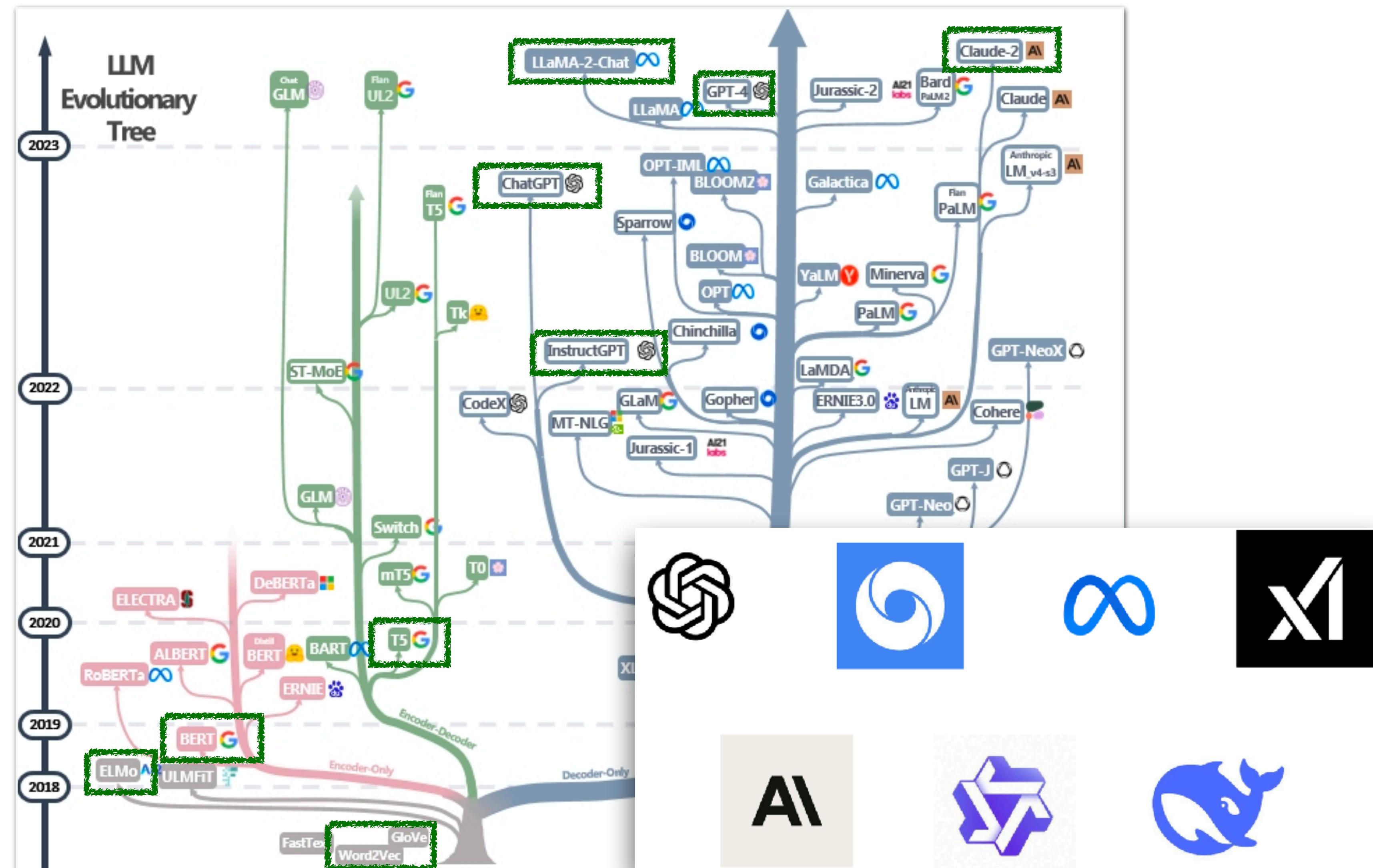
- First train Transformer using a lot of general text using unsupervised learning. This is called **pretraining**.
- Then train the pretrained Transformer for a specific task using supervised learning. This is called **finetuning**.



Example: BERT for sentiment classification



Evolution tree of pretrained LMs



Latest learning paradigm with LLMs

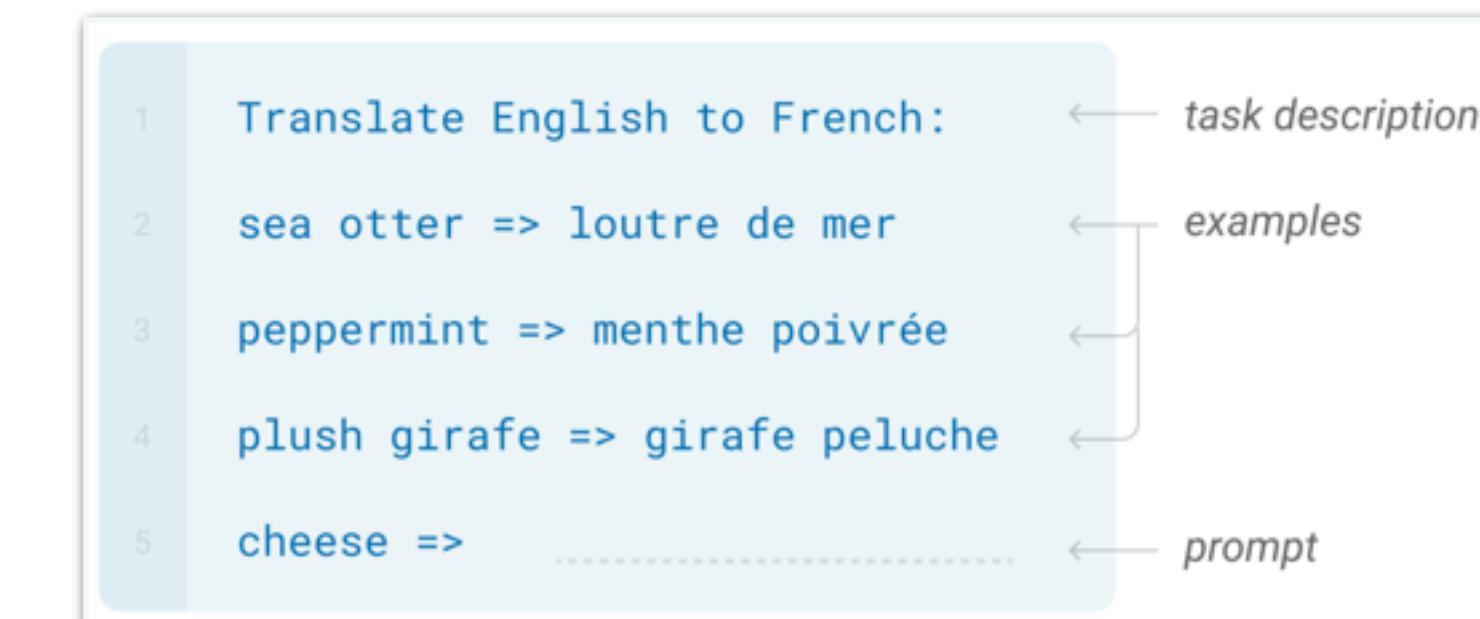
- **Pre-training + prompting/in-context learning (no training this step)**
 - First train a **large (>7~175B)** Transformer using a lot of general text using unsupervised learning. This is called **large language model pretraining**.

Latest learning paradigm with LLMs

- Pre-training + prompting/in-context learning (no training this step)
 - First train a **large (>7~175B)** Transformer using a lot of general text using unsupervised learning. This is called **large language model pretraining**.
 - Then **directly use** the pretrained large Transformer (**no further finetuning/training**) for any different task given only a natural language description of the task or a few task (x, y) examples. This is called **prompting/in-context learning**.



Zero-shot prompting



Few-shot prompting/in-context learning

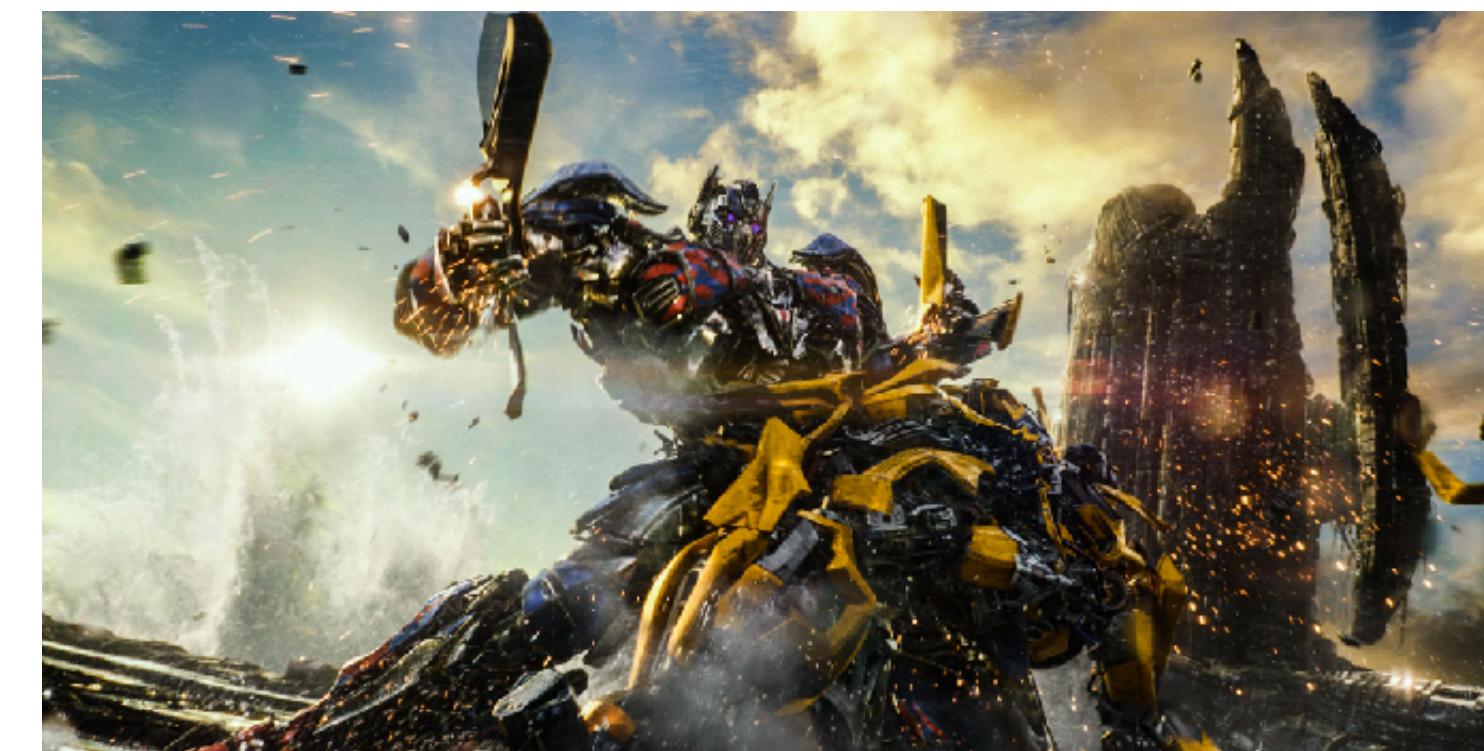
Example: Prompting ChatGPT for sentiment analysis

- Pre-training + prompting/in-context learning (no training this step)

 You
what is the sentiment of "predictable with no fun"? just tell me: positive, negative, or neutral.

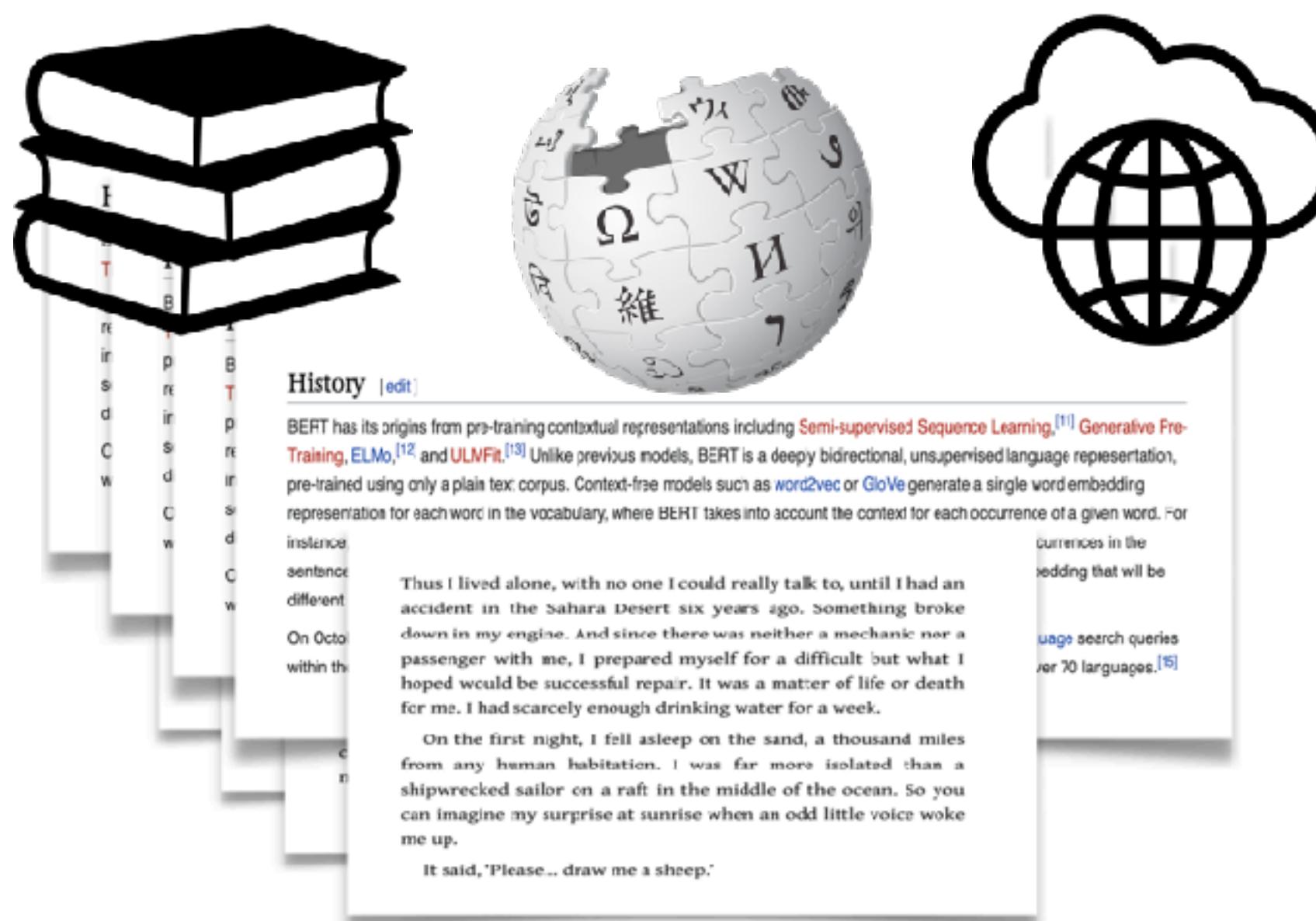
 ChatGPT
Negative.

Already pretrained ChatGPT
No further training for sentiment analysis
Just prompting to conduct the task!



Pretraining: training objectives?

- During pretraining, we have a large text corpus (**no task labels**)
 - **Key question: what labels or objectives used to train the vanilla Transformers?**



Pretraining: training objectives?

- During pretraining, we have a large text corpus (**no task labels**)
 - **Key question: what labels or objectives used to train the vanilla Transformers?**



**Training
labels/objectives?**

Pretraining: training objectives?



BERT

Devlin et al., 2018

The cabs ___ the same rates as those
___ by horse-drawn cabs and were ___
quite popular, ___ the Prince of
Wales (the ___ King Edward VII)
travelled in ___. The cabs quickly
___ known as "hummingbirds" for ___
noise made by their motors and their
distinctive black and ___ livery.
Passengers ___ ___ the interior
fittings were ___ when compared to
___ cabs but there ___ some
complaints ___ the ___ lighting made
them too ___ to those outside ___.

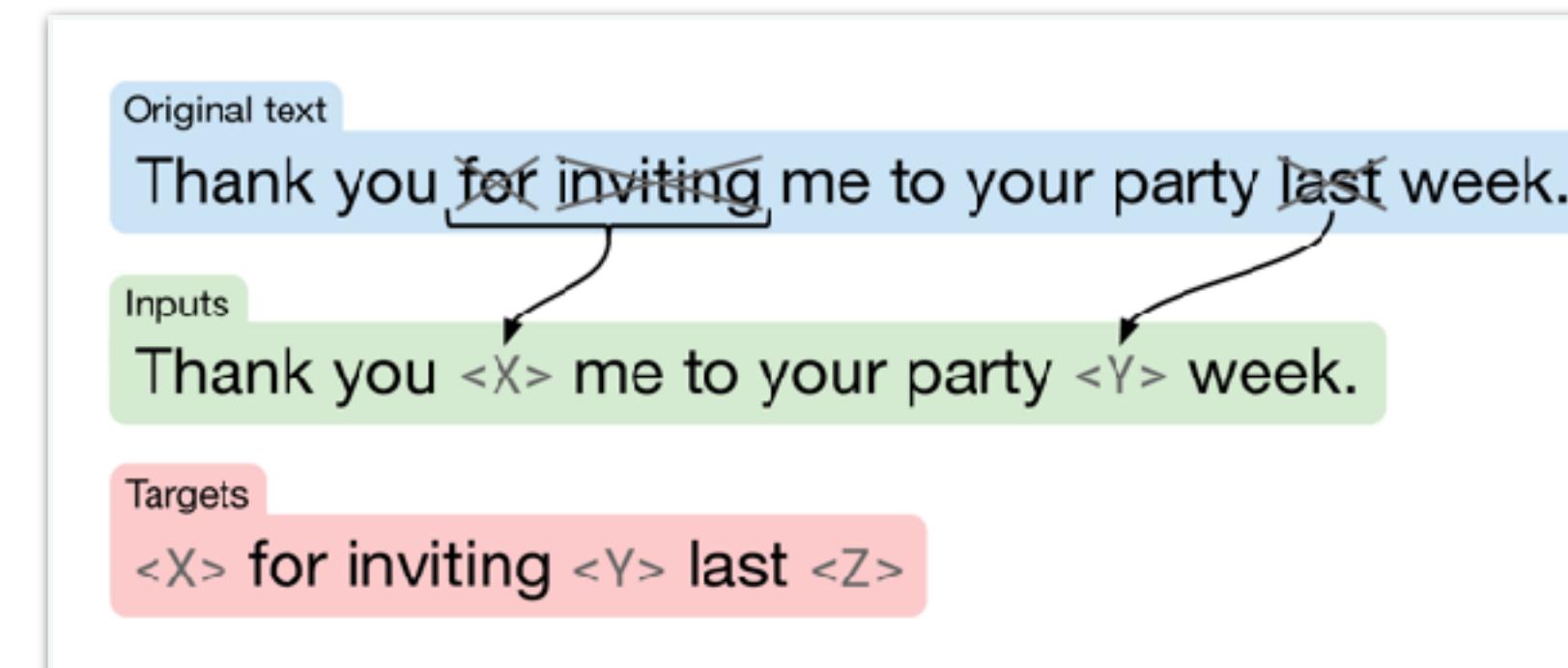
charged, used, initially, even,
future, became, the, yellow,
reported, that, luxurious,
horse-drawn, were that,
internal, conspicuous, cab

Masked token prediction



T5

Raffel et al., 2019



Denoising span-mask prediction



GPT - 4

Text: Second Law of Robotics: A robot must obey the orders given it by human beings

Example #	Input (features)	Correct output (labels)
1	Second law of robotics :	a
2	Second law of robotics : a	robot
3	Second law of robotics : a robot	must
...		

Generated training examples

Advantages of pre-training

- **Leveraging rich underlying information** from abundant raw texts.
- **Reducing the reliance of task-specific labeled data** that is difficult or costly to obtain.
- **Initializing model parameters** for more **generalizable** NLP applications.
- **Saving training cost** by providing a reusable model checkpoints.
- **Providing robust representation** of language contexts.

Pre-training architectures

Encoder

- E.g., BERT, RoBERTa, DeBERTa, ...
- **Autoencoder** model
- **Masked** language modeling

Encoder-Decoder

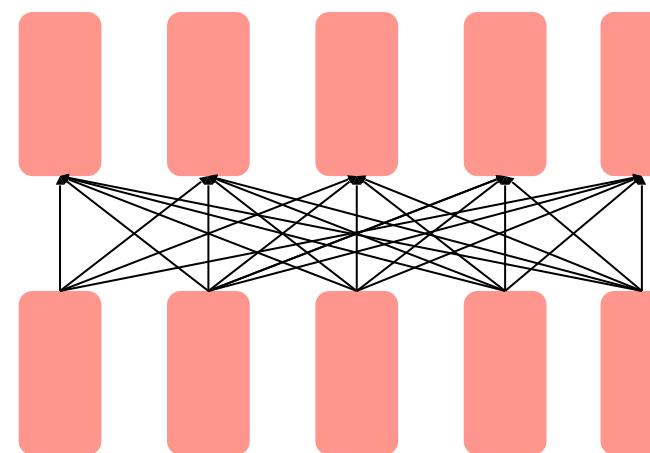
- E.g., T5, BART, ...
- **seq2seq** model

Decoder

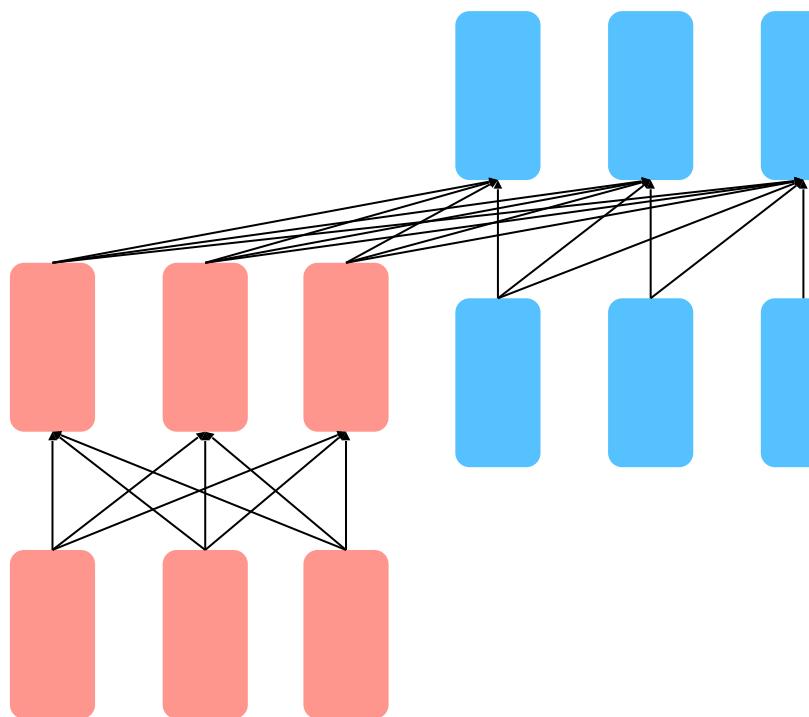
- E.g., GPT, GPT2, GPT3, ...
- **Autoregressive** model
- **Left-to-right** language modeling

Pre-training architectures

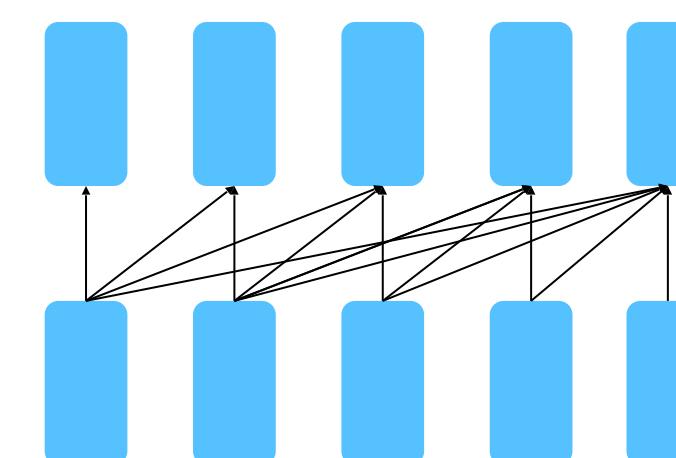
Encoder



Encoder-Decoder



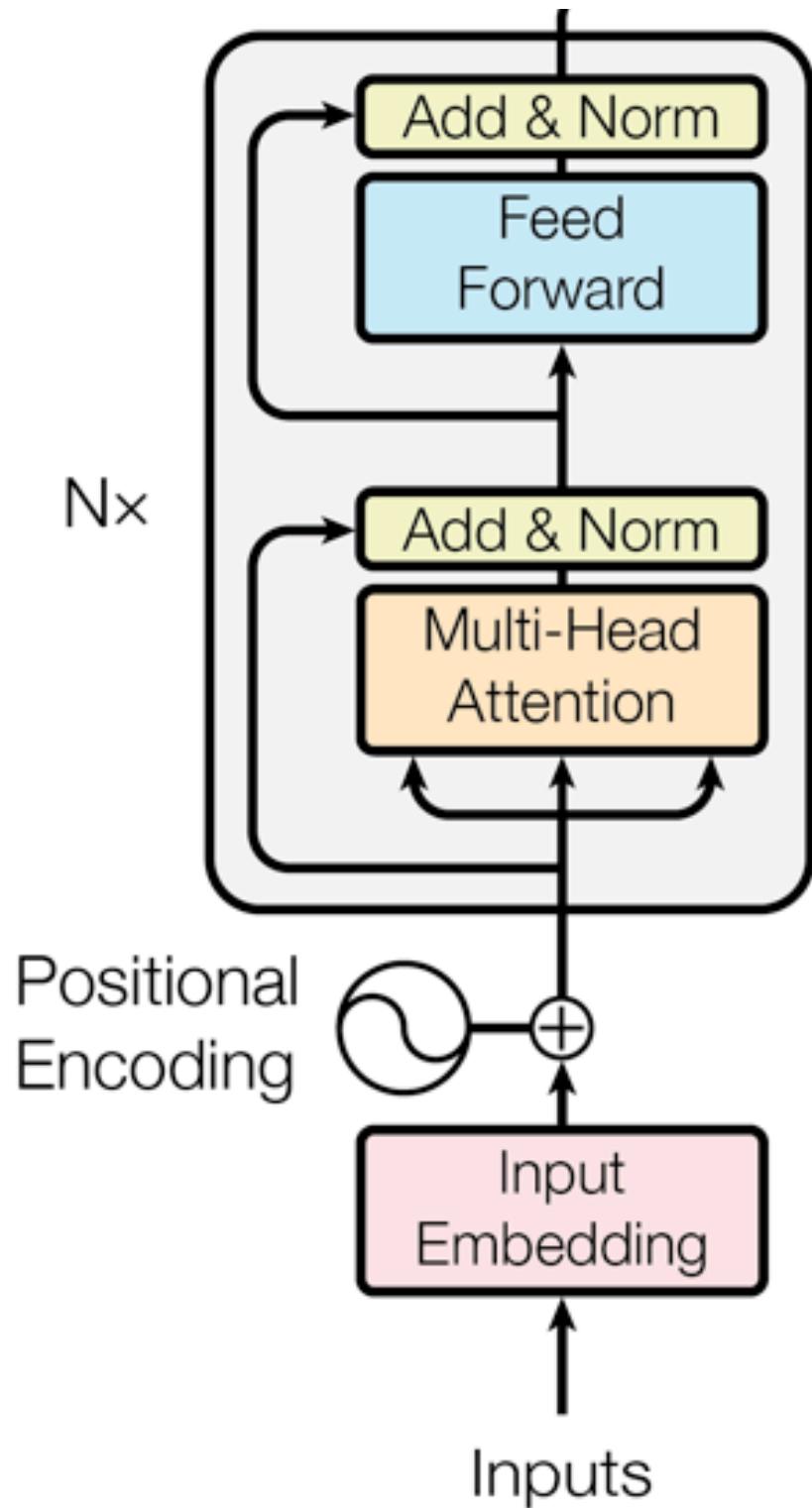
Decoder



- Bidirectional; can condition on the future context
- Map two sequences of different length together
- Language modeling; can only condition on the past context

BERT: Bidirectional Encoder Representations from Transformers

(Released in 2018/10)



- It is a fine-tuning approach based on a deep **bidirectional Transformer encoder** instead of a Transformer decoder
- The key: learn representations based on **bidirectional contexts**

Example #1: we went to the river bank.

Example #2: I need to go to bank to make a deposit.

- Two new pre-training objectives:
 - **Masked language modeling (MLM)**
 - Next sentence prediction (NSP) - Later work shows that NSP hurts performance though..



Masked Language Modeling (MLM)

- Q: Why we can't do language modeling with bidirectional models?



- Solution: Mask out k% of the input words, and then predict the masked words

the man went to [MASK] to buy a [MASK] of milk

store
↑
gallon
↑

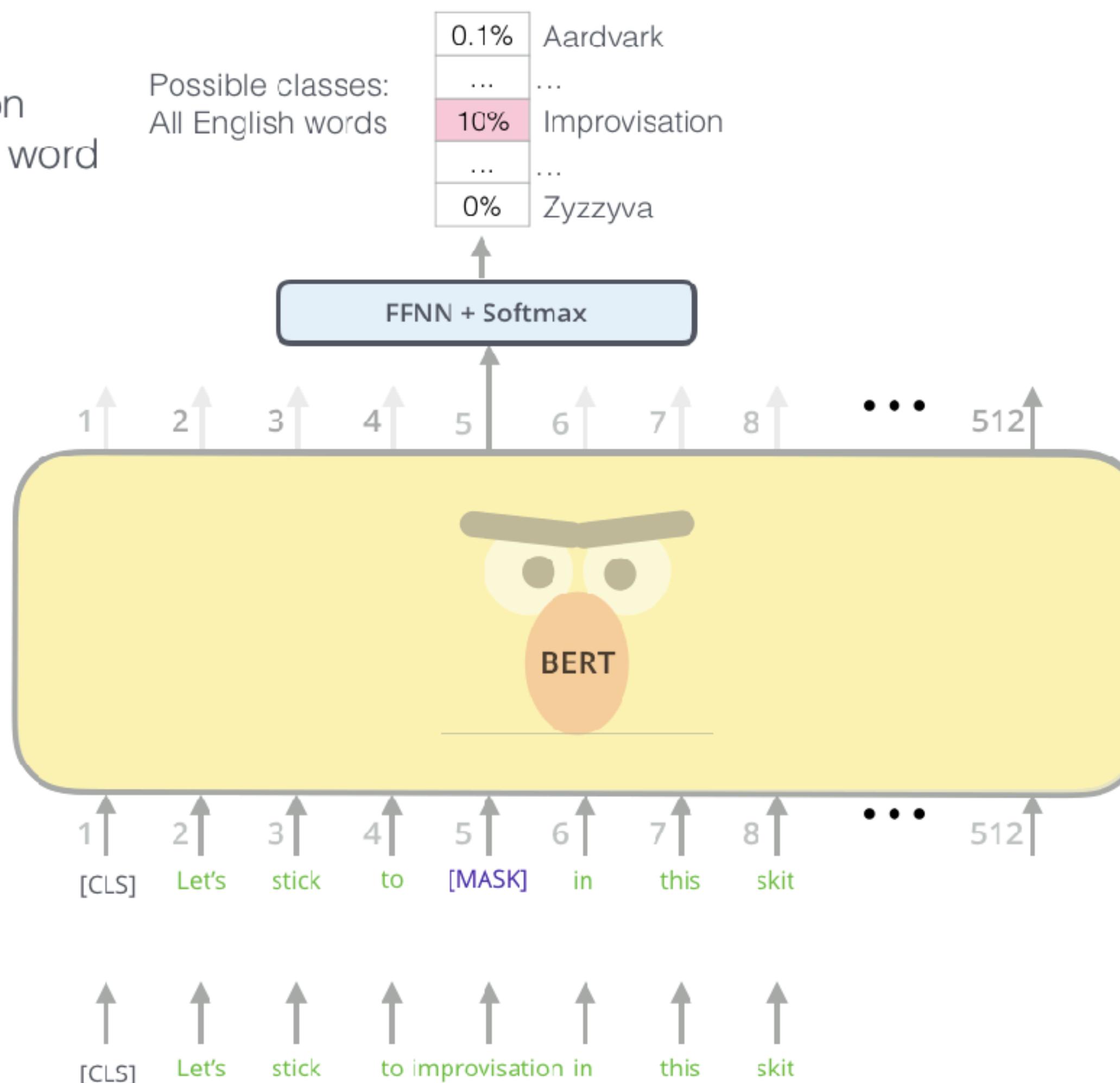
$k = 15\%$ in practice

Masked Language Modeling (MLM)

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

Randomly mask 15% of tokens

Input



MLM: 80-10-10 corruption

For the 15% predicted words,

- 80% of the time, they replace it with [MASK] token

went to the store → went to the [MASK]

- 10% of the time, they replace it with a random word in the vocabulary

went to the store → went to the running

- 10% of the time, they keep it unchanged

went to the store → went to the store

Why? Because [MASK] tokens are never seen during fine-tuning

(See Table 8 of the paper for an ablation study)

Next Sentence Prediction (NSP)

- Motivation: many NLP downstream tasks require understanding the relationship between two sentences (natural language inference, paraphrase detection, QA)
- NSP is designed to reduce the gap between pre-training and fine-tuning

[CLS]: a special token
always at the beginning

Input = [CLS] the man went to [MASK] store [SEP]

he bought a gallon [MASK] milk [SEP]

Label = IsNext

[SEP]: a special token used
to separate two segments



They sample two contiguous segments for 50% of the time and another random segment from the corpus for 50% of the time

Input = [CLS] the man [MASK] to the store [SEP]

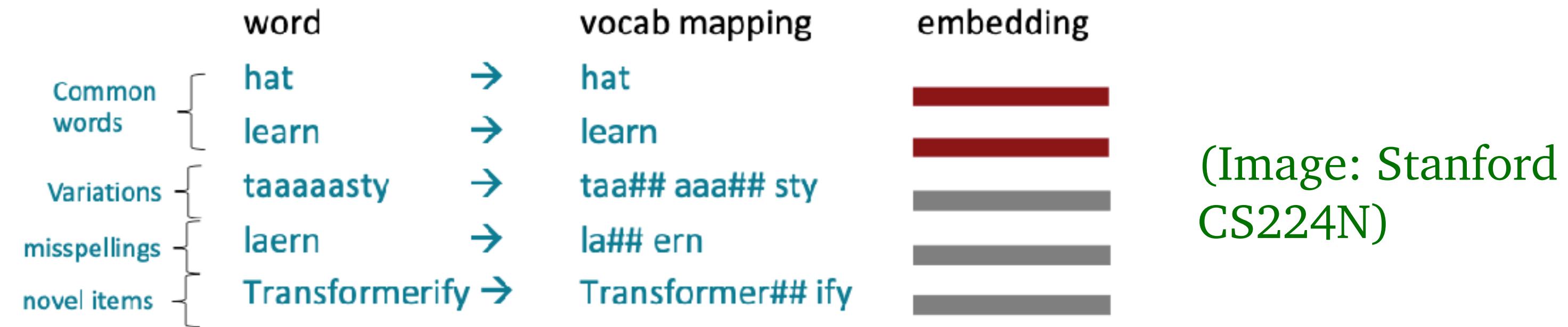
penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

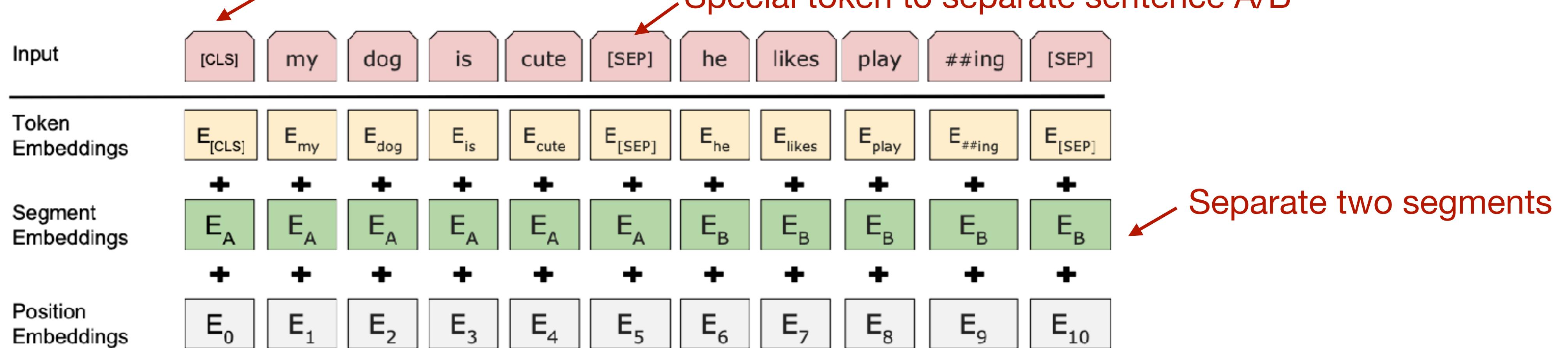
This actually hurts model learning based on later work!

BERT pre-training

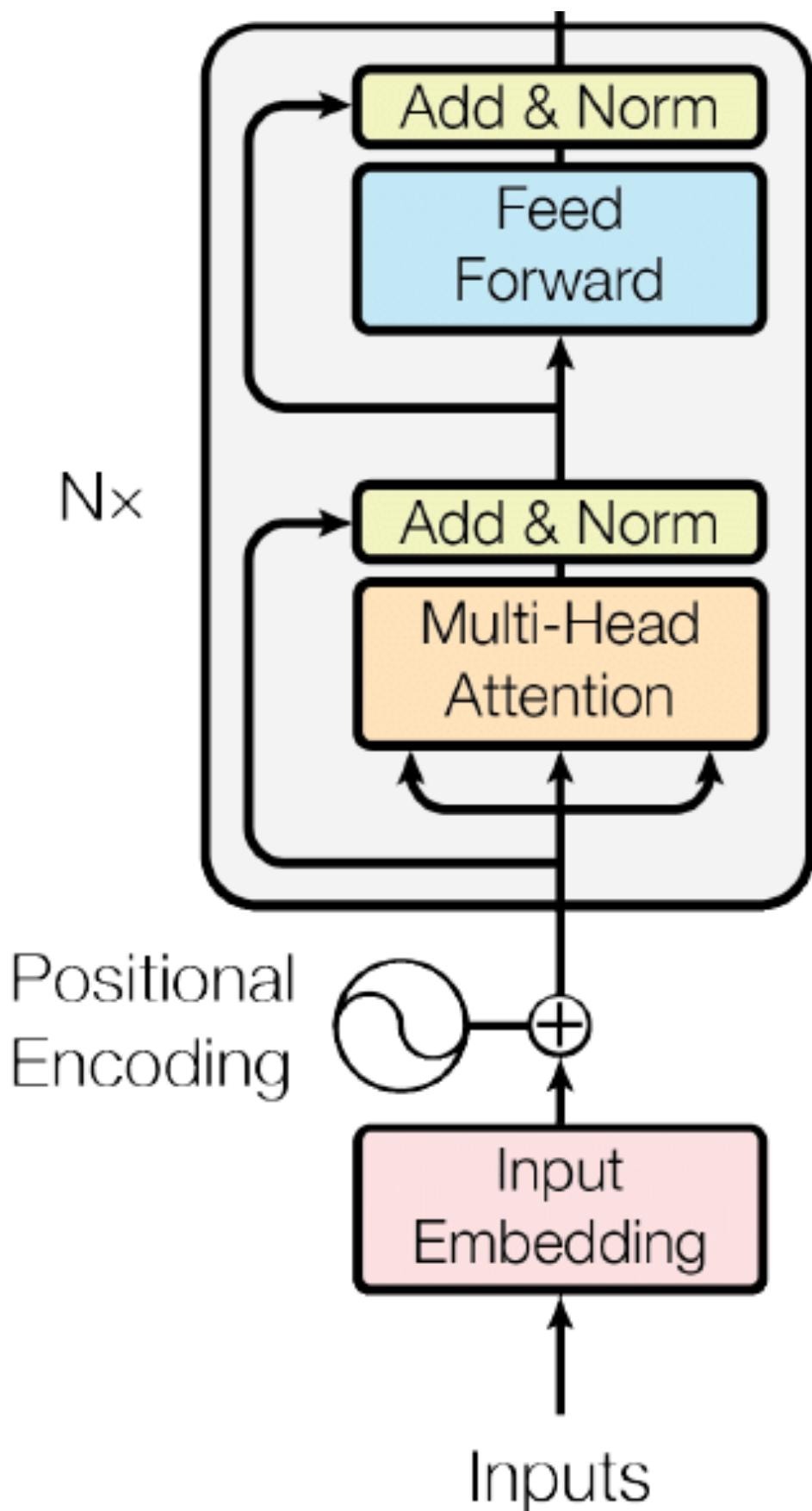
- Vocabulary size: 30,000 wordpieces (common sub-word units) (Wu et al., 2016)



- Input embeddings: Special token added to the beginning of each input sequence

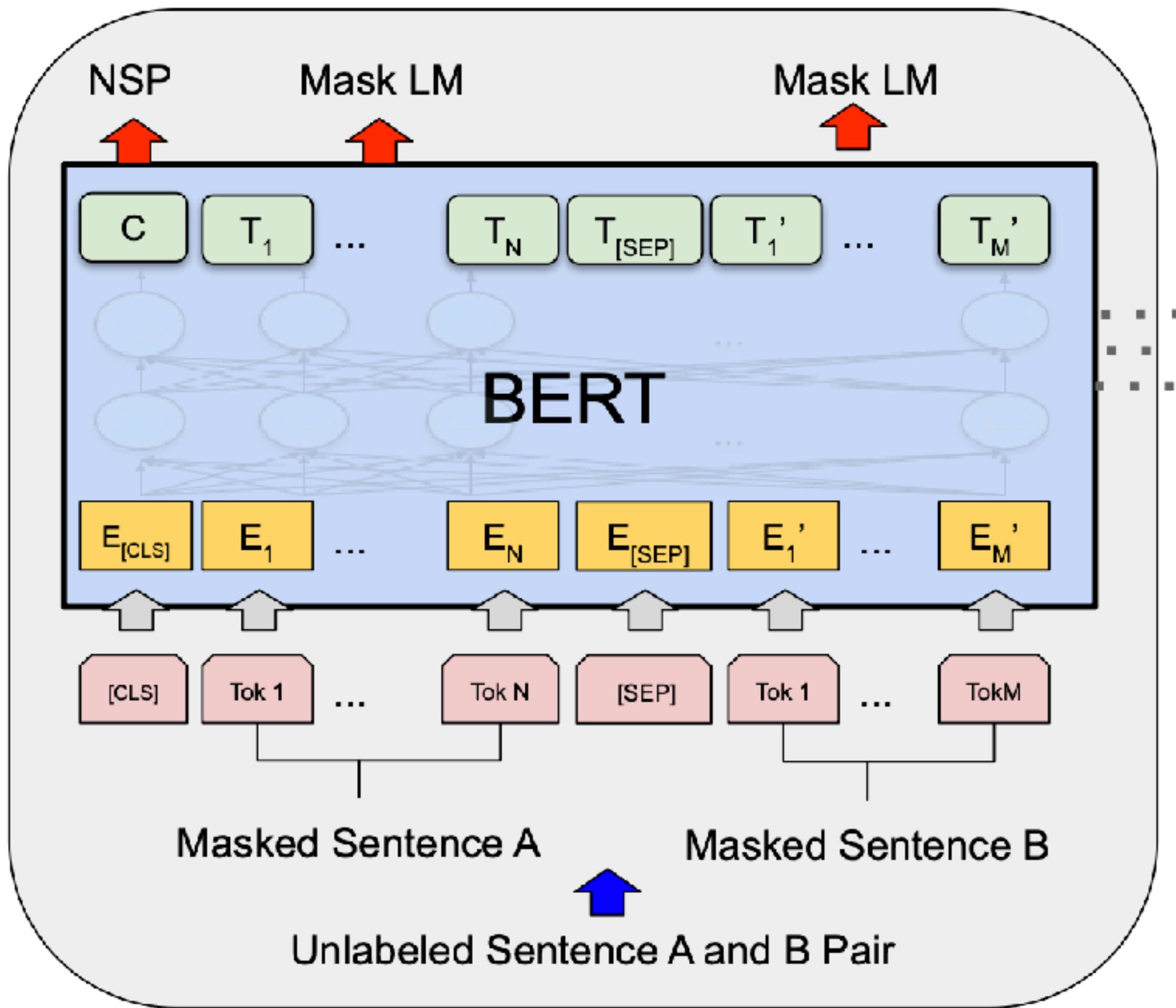


BERT pre-training



- BERT-base: 12 layers, 768 hidden size, 12 attention heads, 110M parameters
- BERT-large: 24 layers, 1024 hidden size, 16 attention heads, 340M parameters
- Training corpus: Wikipedia (2.5B) + BooksCorpus (0.8B)
- Max sequence size: 512 wordpiece tokens (roughly 256 and 256 for two non-contiguous sequences)
- Trained for 1M steps, batch size 128k

BERT pre-training

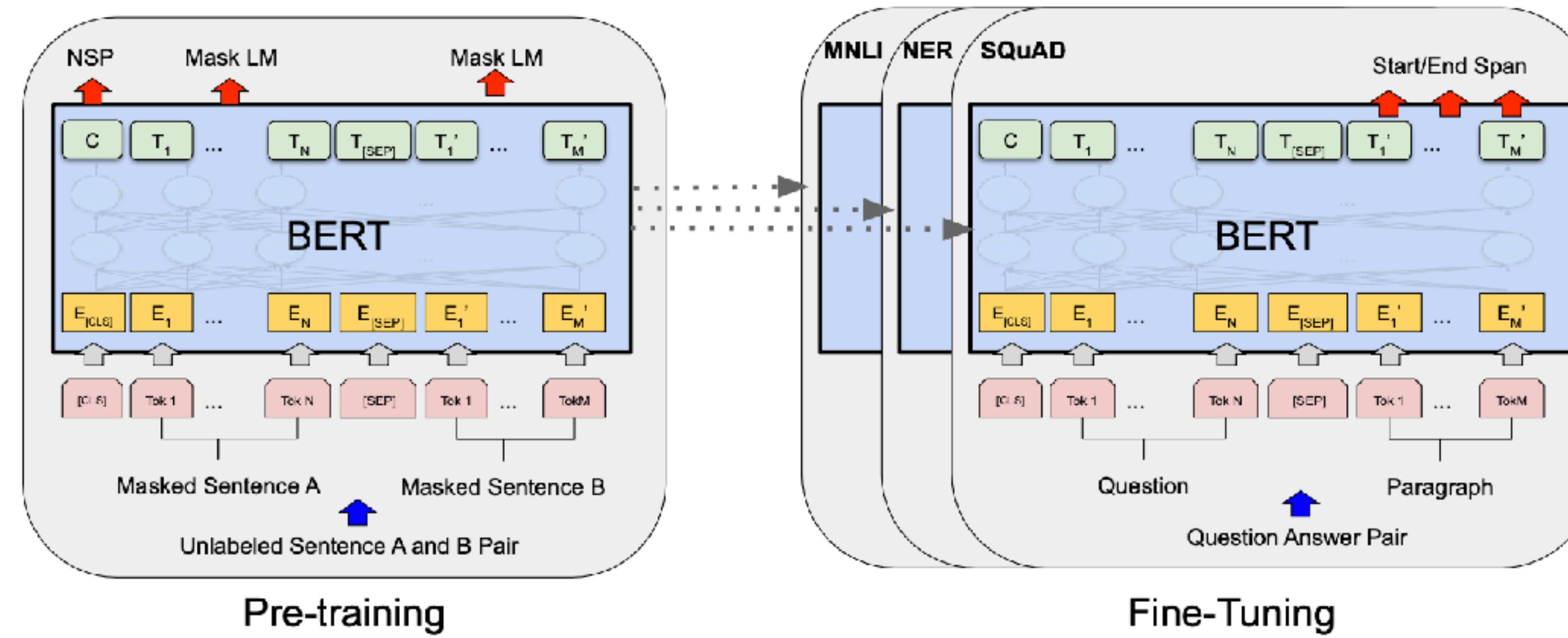


Pre-training

- MLM and NSP are trained together
- [CLS] is pre-trained for NSP
- Other token representations are trained for MLM

Pretraining / fine-tuning

“Pre-train” a model on a large dataset for task X, then “fine-tune” it on a dataset for task Y



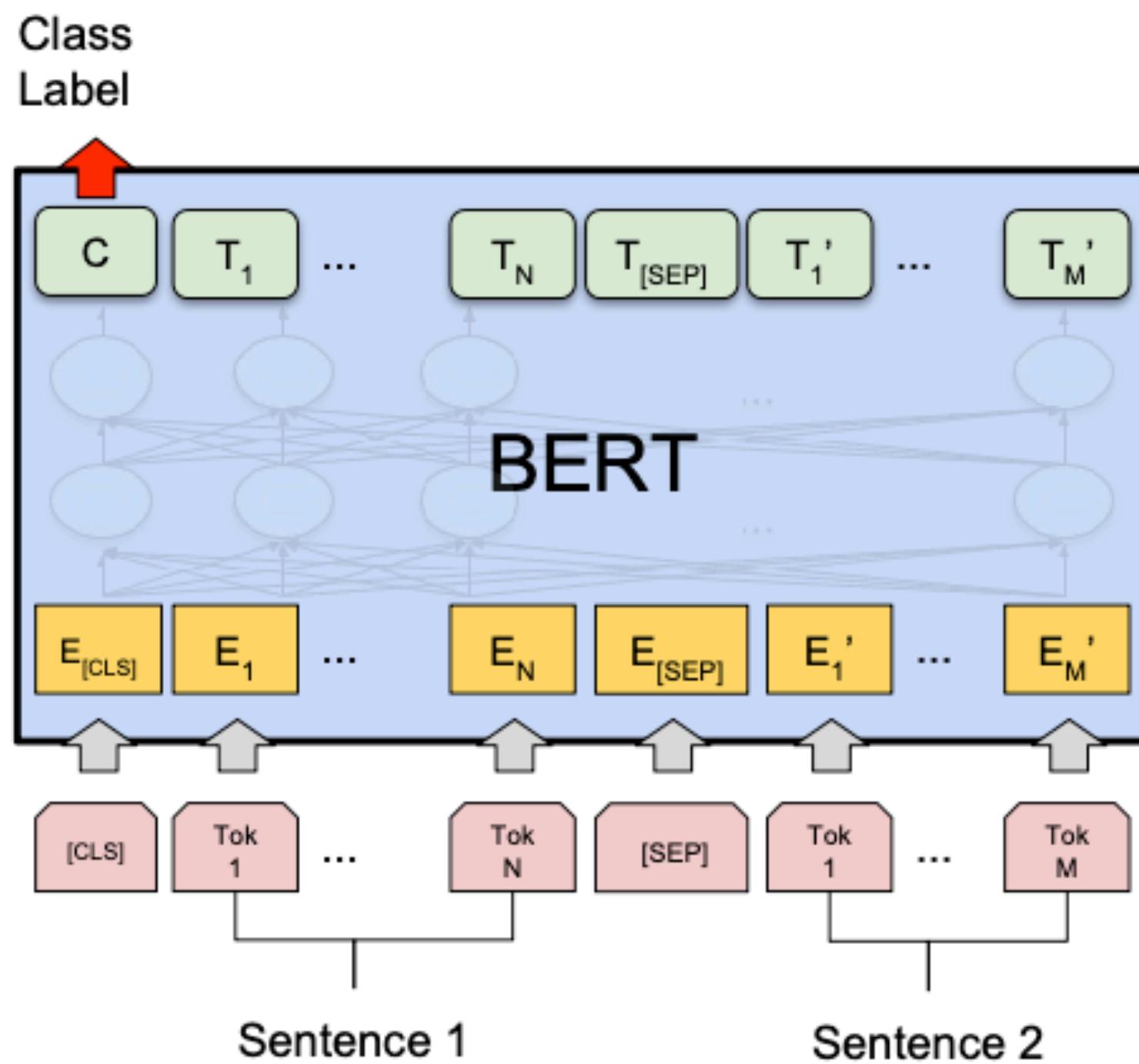
“**Fine-tuning** is the process of **taking the network learned by these pre-trained models**, and **further training the model**, often via an added neural net classifier that takes the top layer of the network as input, to perform some downstream task.”

Fine-tuning is a training process and takes **gradient descent steps!**

BERT fine-tuning

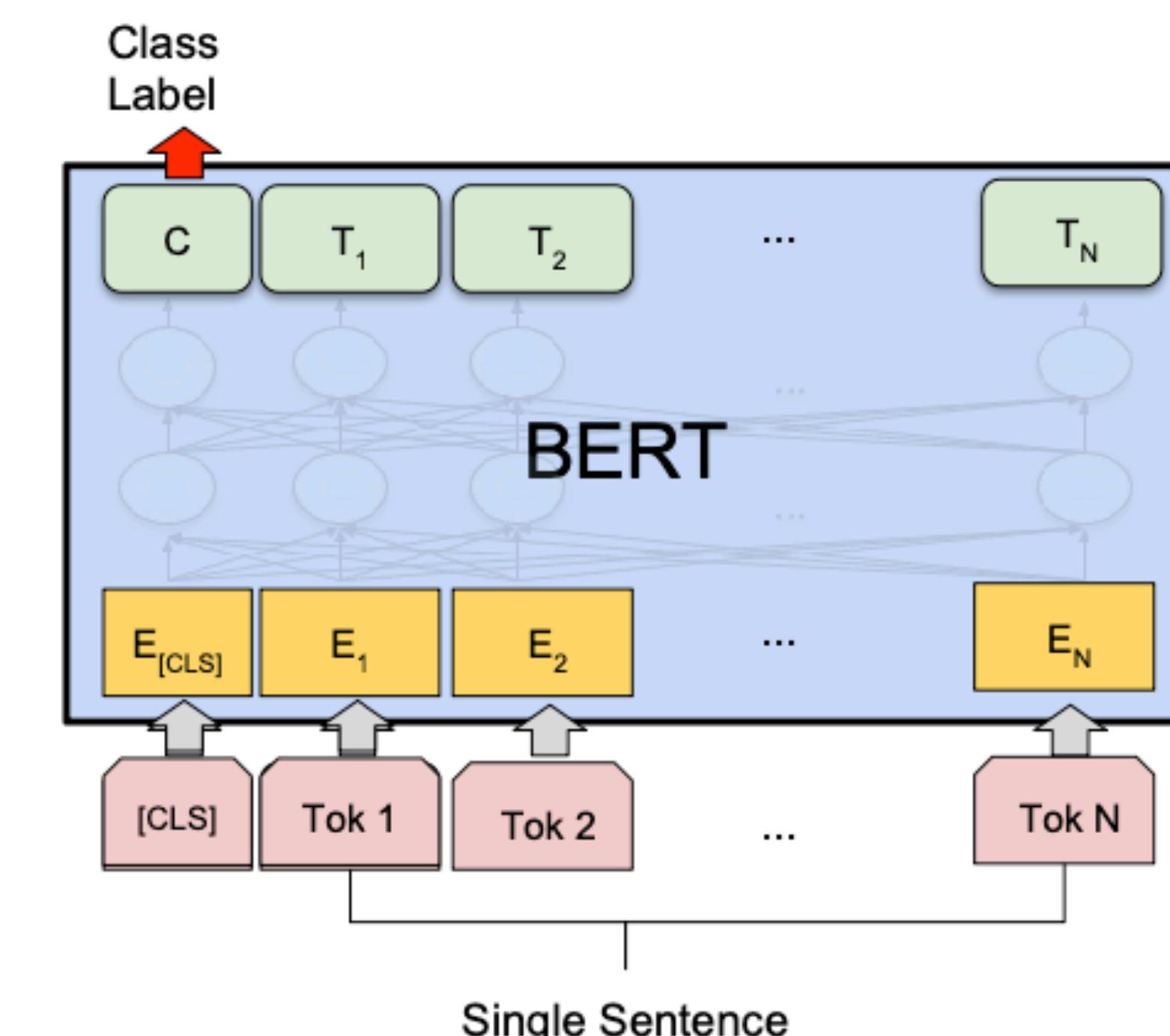
“Pretrain once, finetune many times.”

sentence-level tasks



(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI:** natural language inference over question answering data
- **SST-2:** sentiment analysis

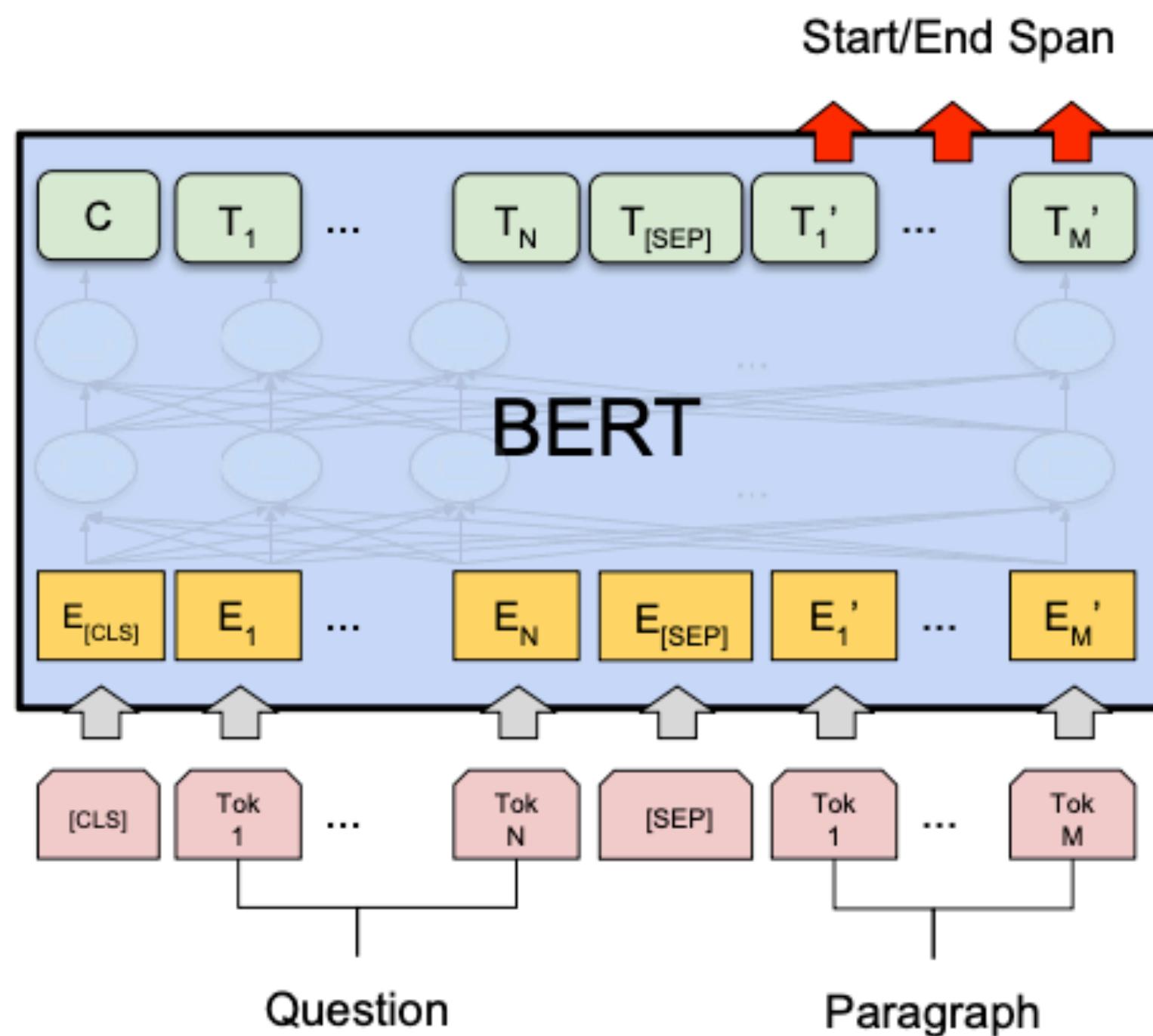


(b) Single Sentence Classification Tasks:
SST-2, CoLA

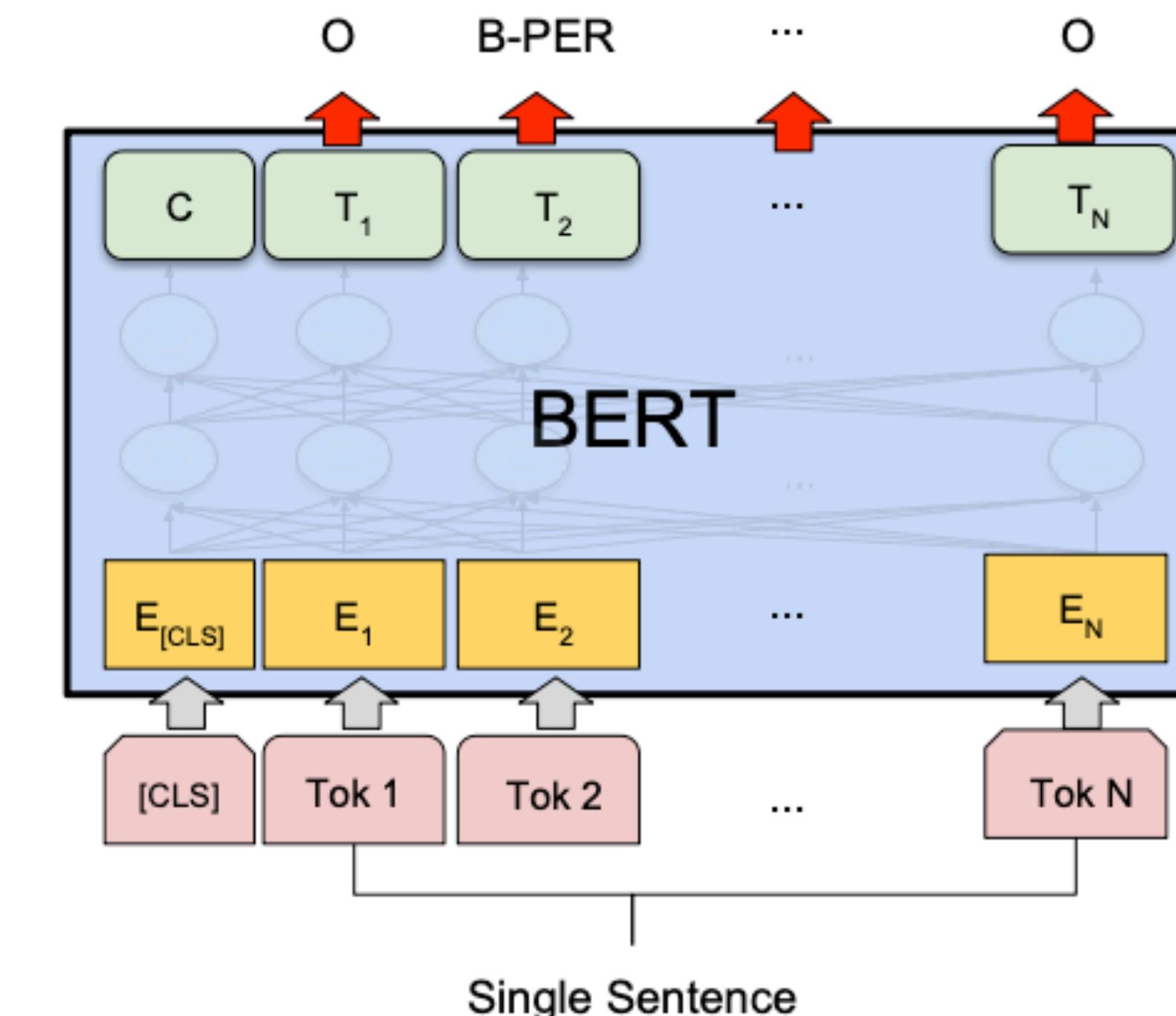
BERT fine-tuning

“Pretrain once, finetune many times.”

token-level tasks

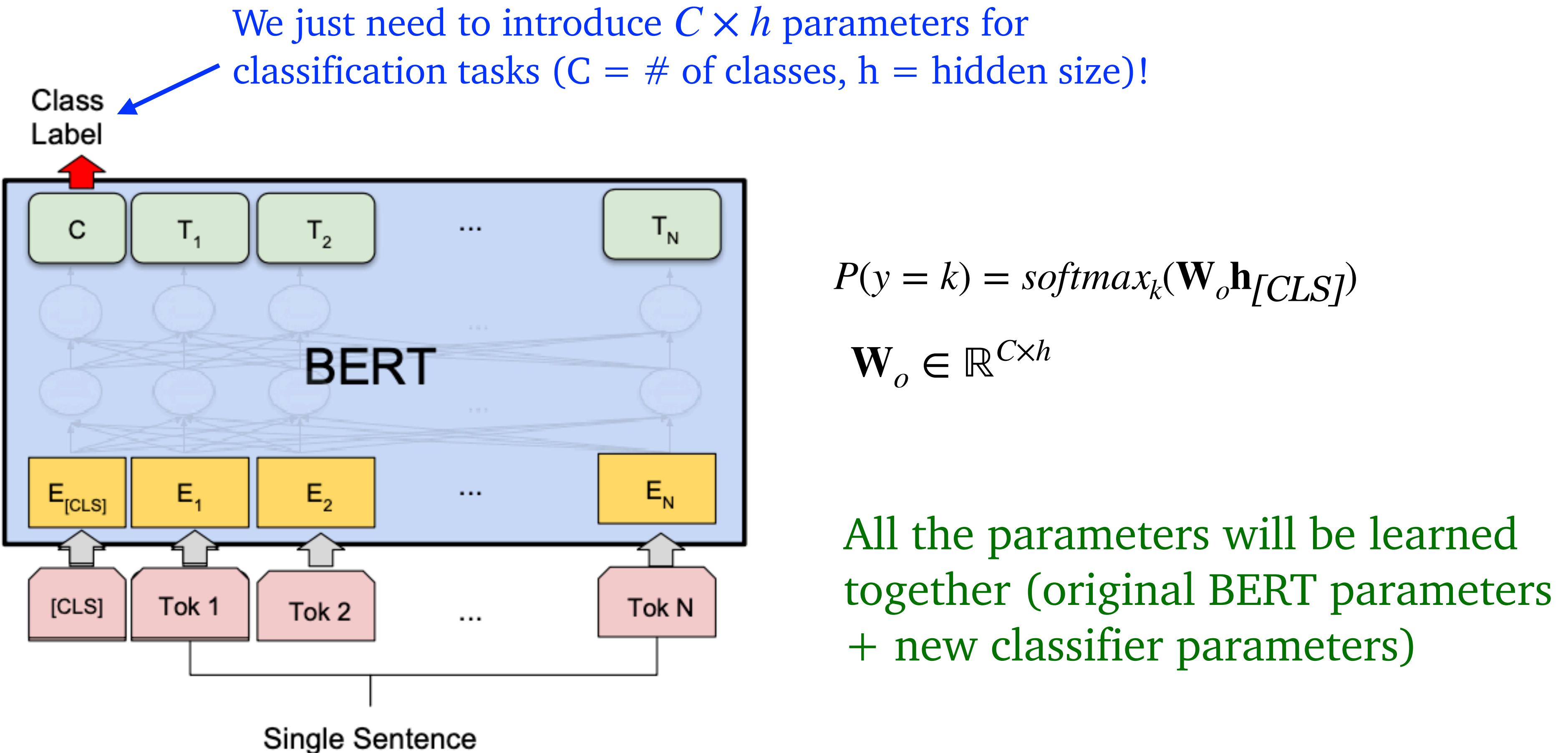


(c) Question Answering Tasks:
SQuAD v1.1

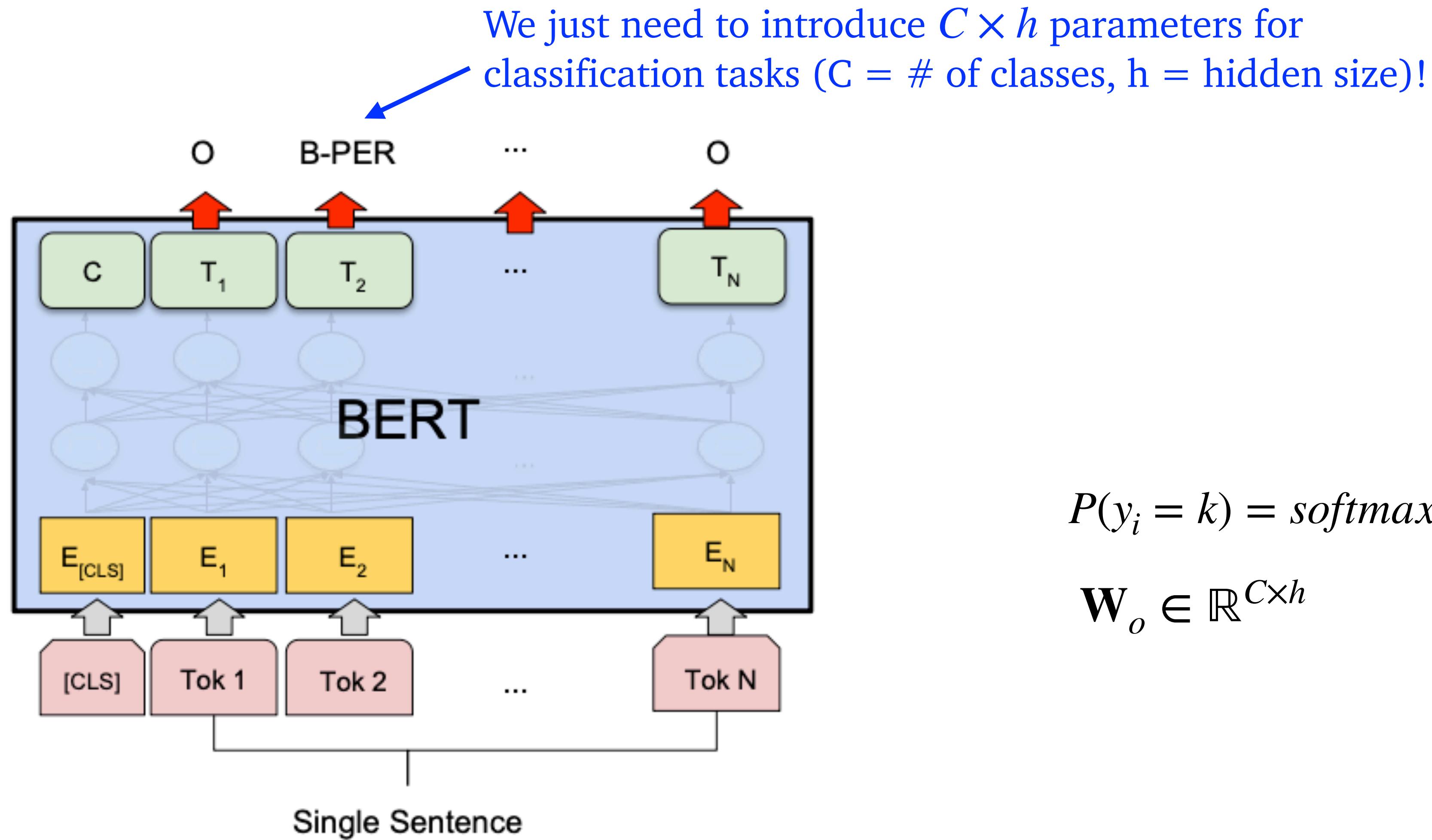


(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Example: sentiment classification



Example: named entity recognition (NER)

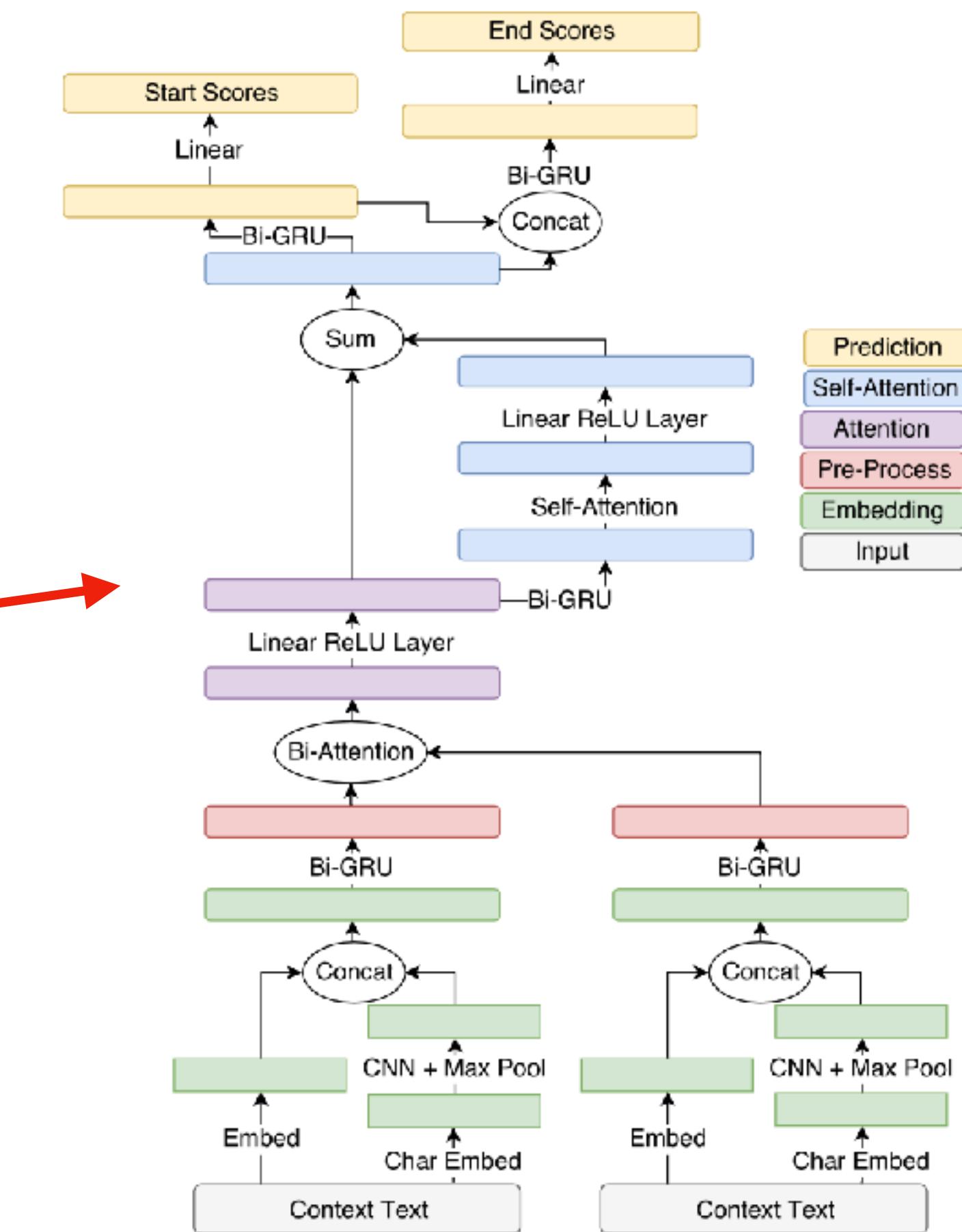


Experimental results: GLUE

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

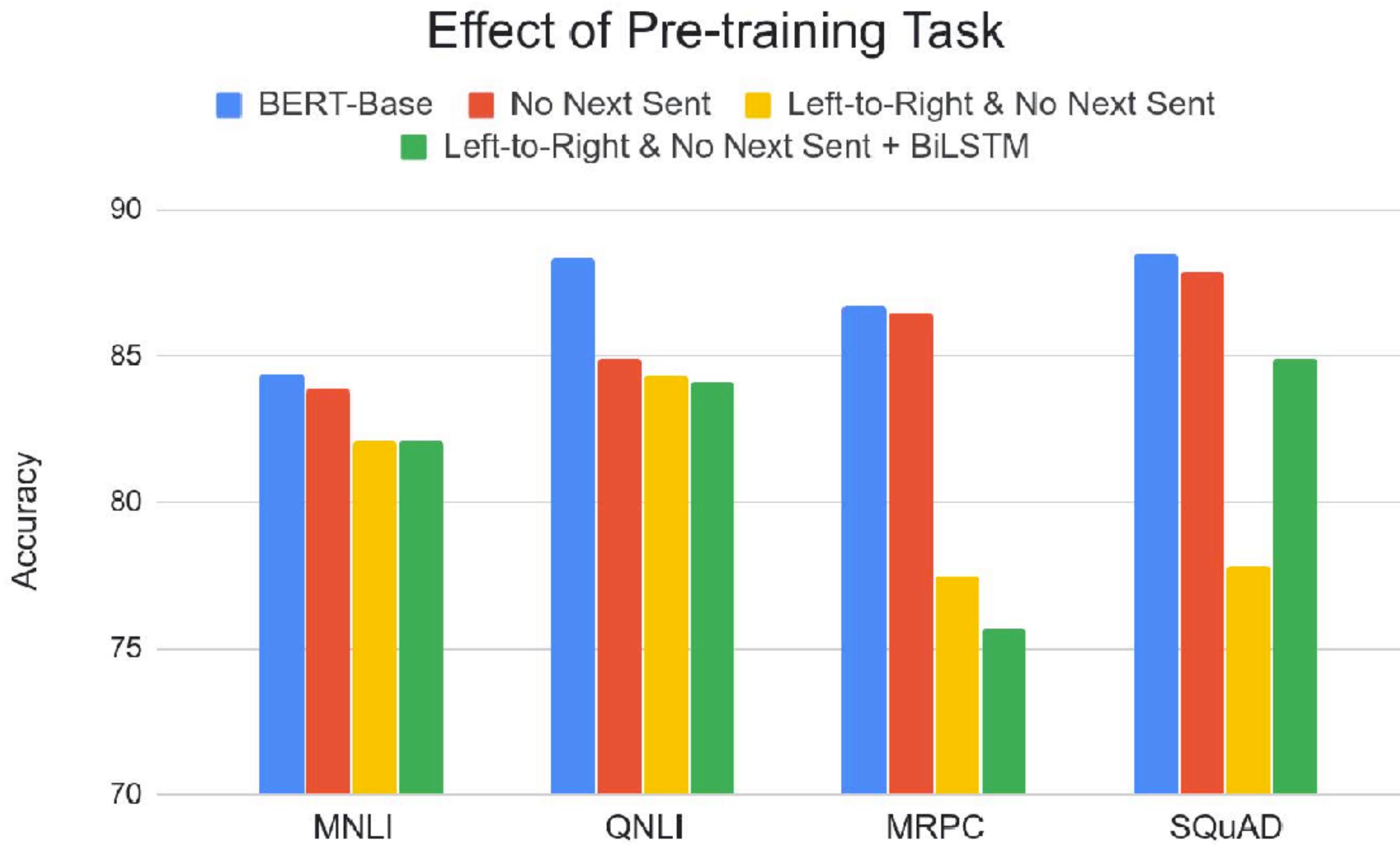
Experimental results: SQuAD

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT _{BASE} (Single)	80.8	88.5	-	-
BERT _{LARGE} (Single)	84.1	90.9	-	-
BERT _{LARGE} (Ensemble)	85.8	91.8	-	-
BERT _{LARGE} (Sgl.+TriviaQA)	84.2	91.1	85.1	91.8
BERT _{LARGE} (Ens.+TriviaQA)	86.2	92.2	87.4	93.2



SQuAD = Stanford Question Answering dataset

Ablation study: pre-training tasks



- MLM >> left-to-right LMs
- NSP improves on some tasks
- Note: later work ([Joshi et al., 2020](#); [Liu et al., 2019](#)) argued that NSP is not useful

Ablation study: model sizes

# layers	hidden size	# of heads	Dev Set Accuracy			
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

The bigger, the better!

Encoder: other variations of BERT

- **ALBERT [Lan et al., 2020]**: incorporates two parameter reduction techniques that lift the major obstacles in scaling pre-trained models
- **DeBERTa [He et al., 2021]**: decoding-enhanced BERT with disentangled attention
- **SpanBERT [Joshi et al., 2019]**: masking contiguous spans of words makes a harder, more useful pre-training task
- **ELECTRA [Clark et al., 2020]**: corrupts texts by replacing some tokens with plausible alternatives sampled from a small generator network, then train a discriminative model that predicts whether each token in the corrupted input was replaced by a generator sample or not.
- **DistilBERT [Sanh et al., 2019]**: distilled version of BERT that's 40% smaller
- **TinyBERT [Jiao et al., 2019]**: distill BERT for both pre-training & fine-tuning
- ...

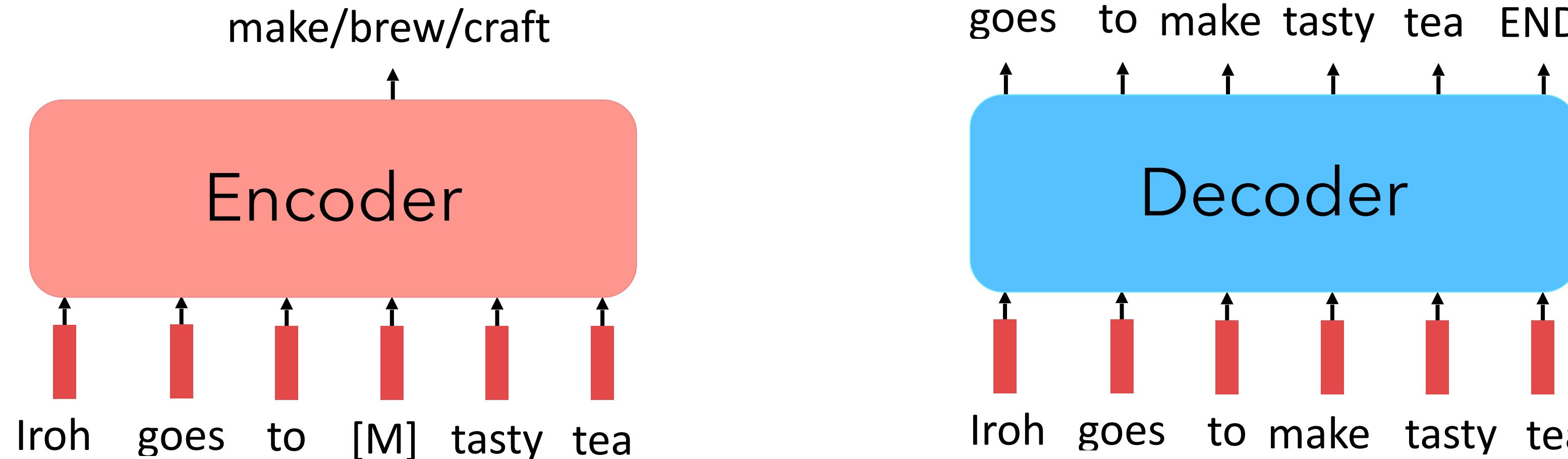
Encoder: pros & cons



- Consider both left and right context
- Capture intricate contextual relationships

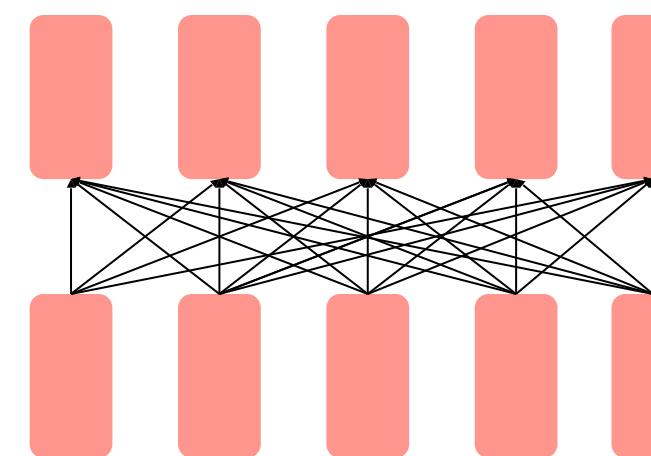


- Not good at generating open-text from left-to-right, one token at a time

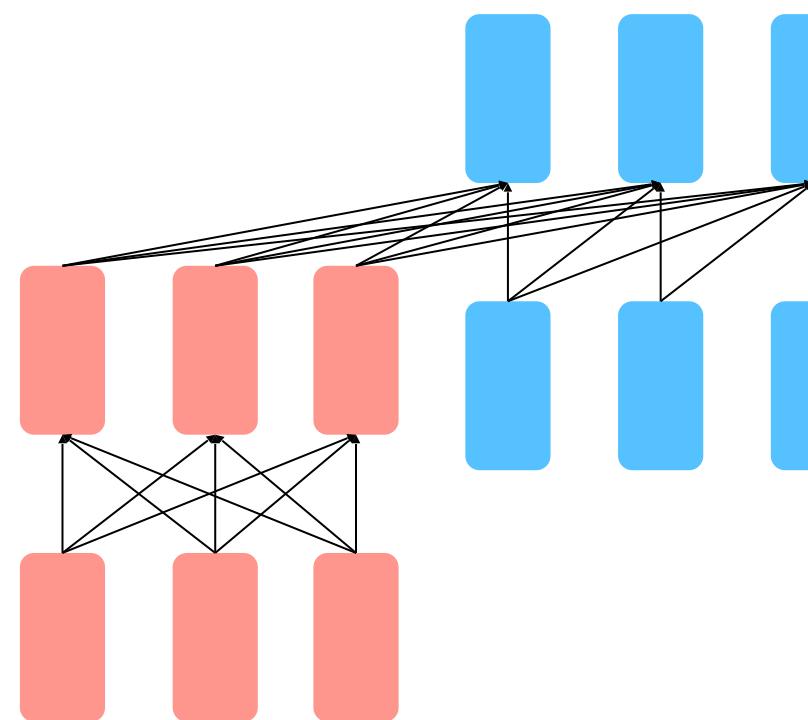


Pre-training architectures

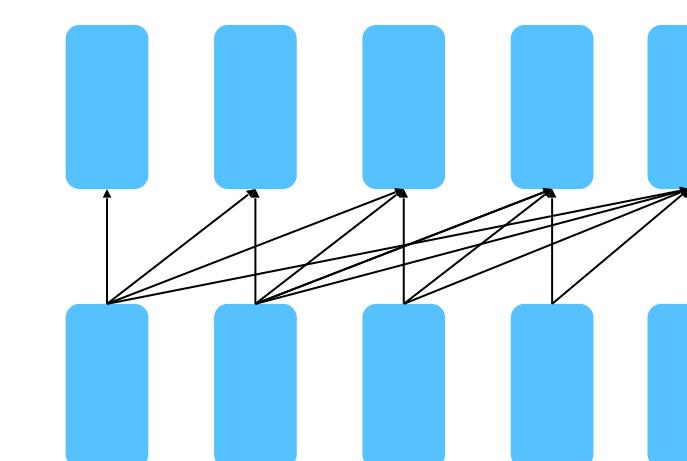
Encoder



Encoder-Decoder



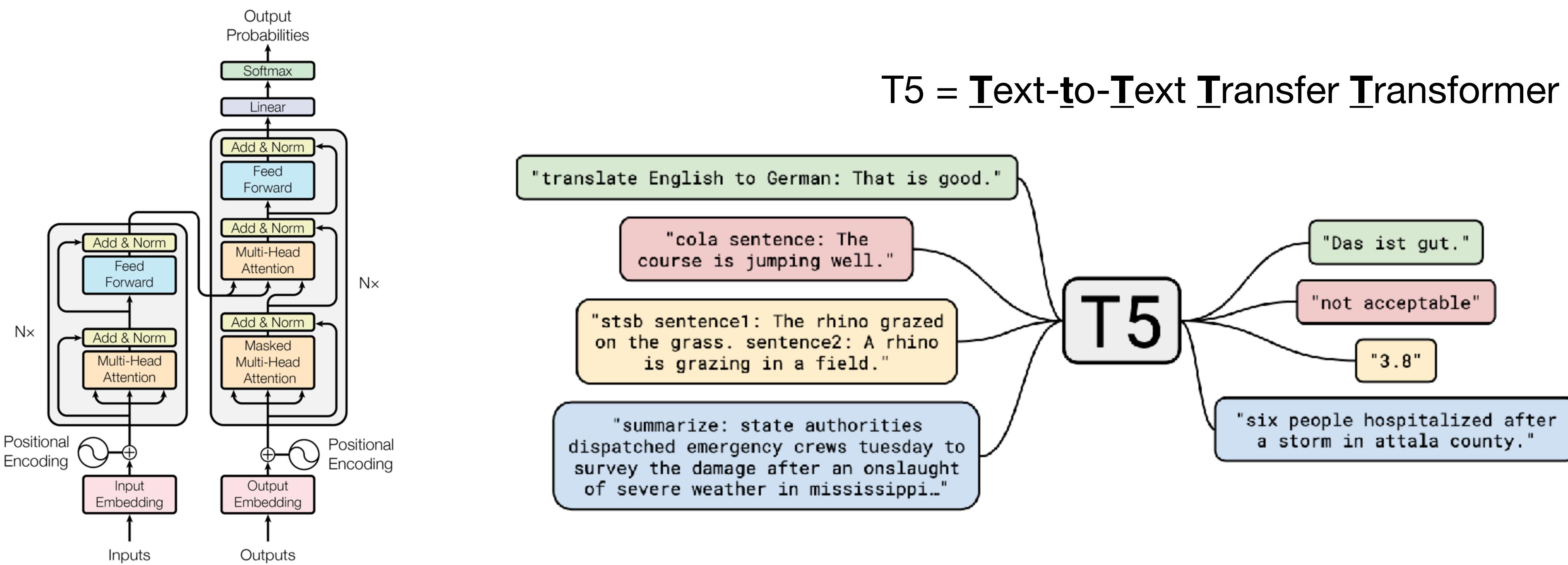
Decoder



- Bidirectional; can condition on the future context
- Map two sequences of different length together
- Language modeling; can only condition on the past context

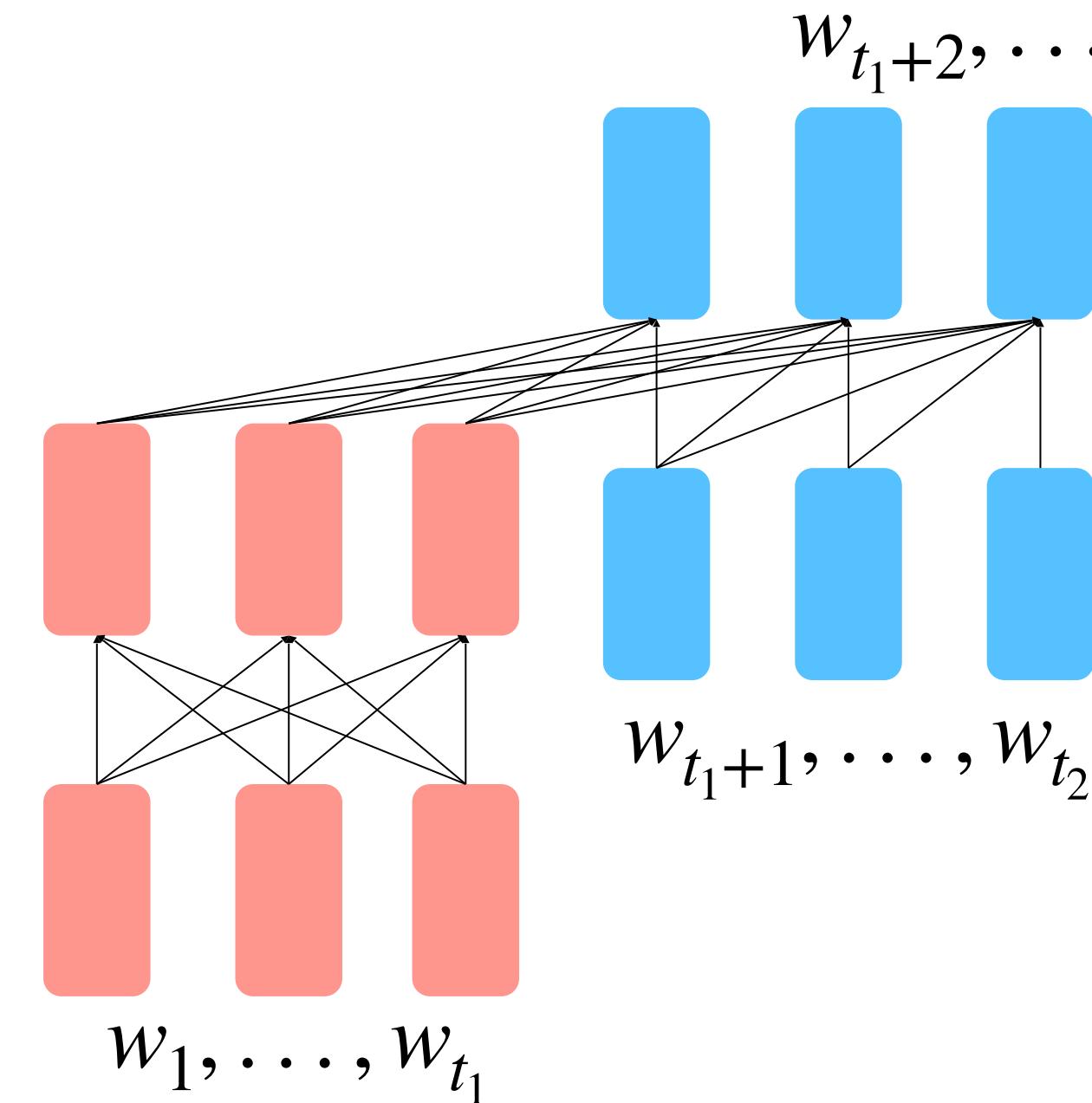
Text-to-text models: the best of both worlds

- So far, **encoder-only models (e.g., BERT)** enjoy the benefits of **bidirectionality** but they can't be used to generate text
- **Decoder-only models (e.g., GPT)** can do generation but they are left-to-right LMs..
- **Text-to-text models combine the best of both worlds!**



Encoder-decoder: architecture

- Moving towards **open-text generation**...
- **Encoder** builds a representation of the source and gives it to the **decoder**
- **Decoder** uses the source representation to generate the target sentence
- The **encoder** portion benefits from **bidirectional** context; the **decoder** portion is used to train the whole model through **language modeling**



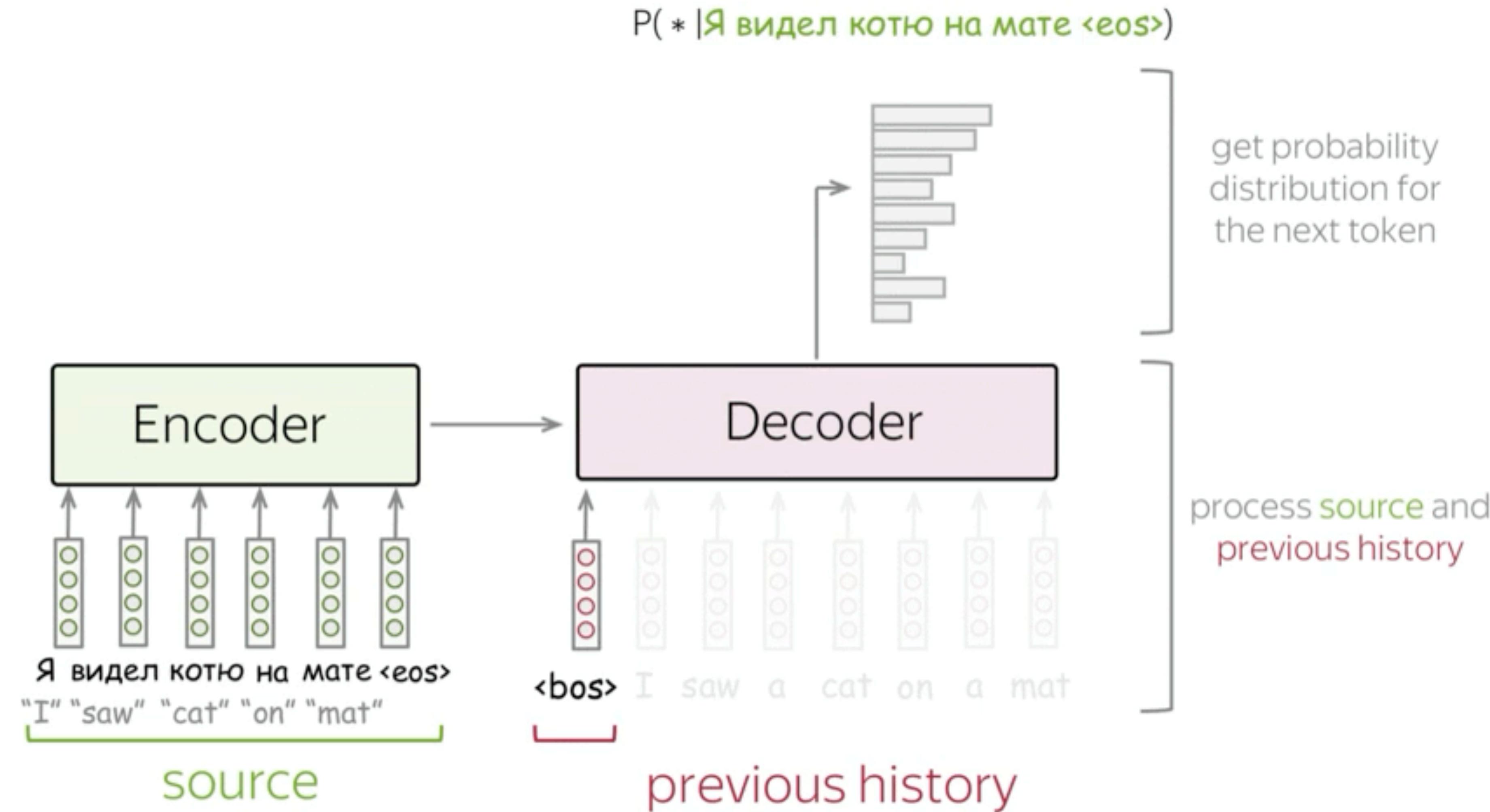
$$h_1, \dots, h_{t_1} = \text{Encoder}(w_1, \dots, w_{t_1})$$

$$h_{t_1+1}, \dots, h_{t_2} = \text{Decoder}(w_{t_1+1}, \dots, w_{t_2}, h_1, \dots, h_{t_1})$$

$$y_i \sim Ah_i + b, i > t$$

[Raffel et al., 2018]

Encoder-decoder: machine translation example

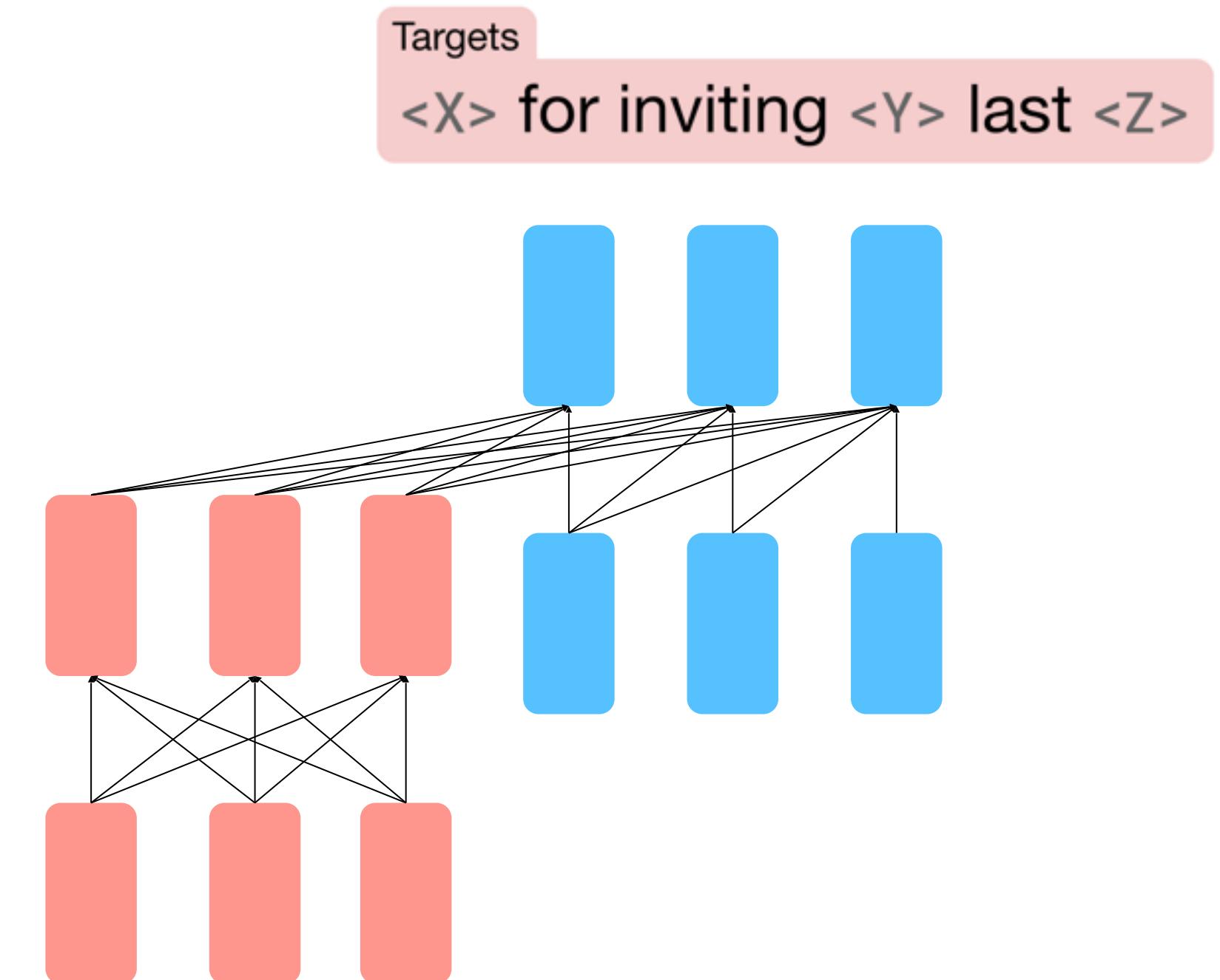


Encoder-decoder: training objective

- T5 [Raffel et al., 2018]
- **Text span corruption (denoising):** Replace different-length spans from the input with unique placeholders (e.g., <extra_id_0>); decode out the masked spans.
- Done during **text preprocessing:** training uses **language modeling** objective at the decoder side

Original text
Thank you ~~for inviting~~ me to your party ~~last~~ week.

Inputs
Thank you <X> me to your party <Y> week.



Encoder-decoder:T5

[Raffel et al., 2018]

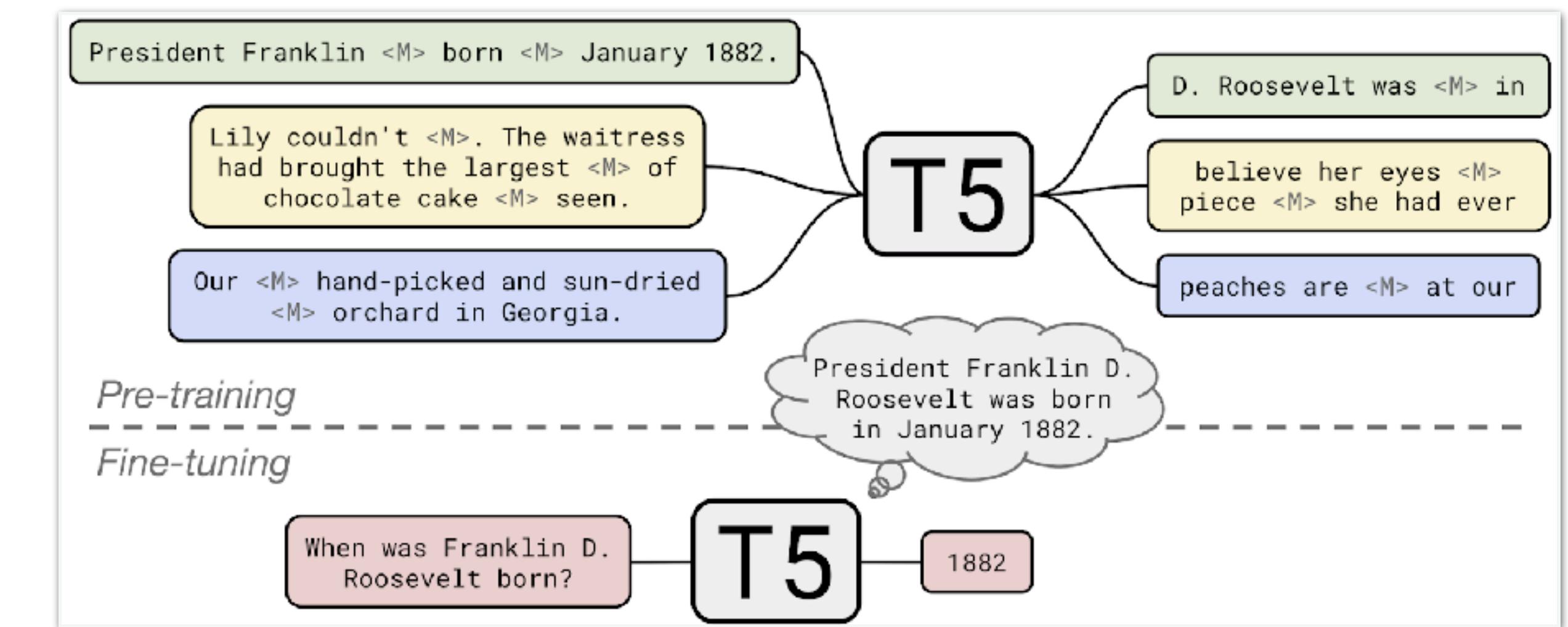
- **Encoder-decoders** works better than decoders
- **Span corruption (denoising)** objective works better than language modeling

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Enc-dec, shared	Denoising	P	M	82.81	18.78	80.63	70.73	26.72	39.03	27.46
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

Encoder-decoder:T5

[Raffel et al., 2018]

- **Text-to-Text:** convert NLP tasks into input/output text sequences
- **Dataset:** Colossal Clean Crawled Corpus (C4), 750G text data!
- **Various Sized Models:**
 - Base (222M)
 - Small (60M)
 - Large (770M)
 - 3B
 - 11B
- **Achieved SOTA with scaling & purity of data**



Encoder-decoder: pros & cons



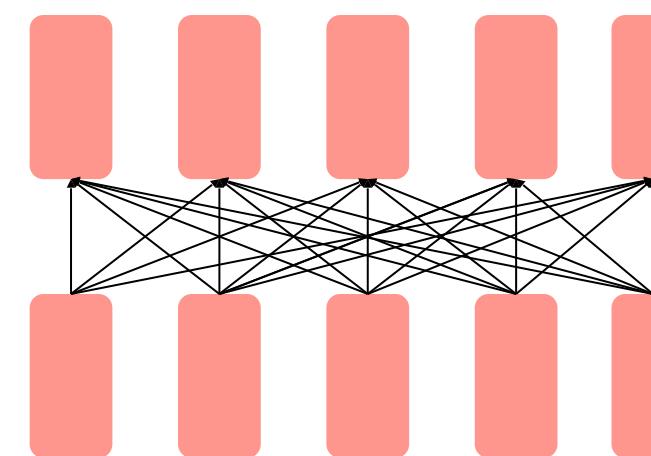
- A nice middle ground between leveraging **bidirectional** contexts and **open-text** generation
- Good for **multi-task** fine-tuning



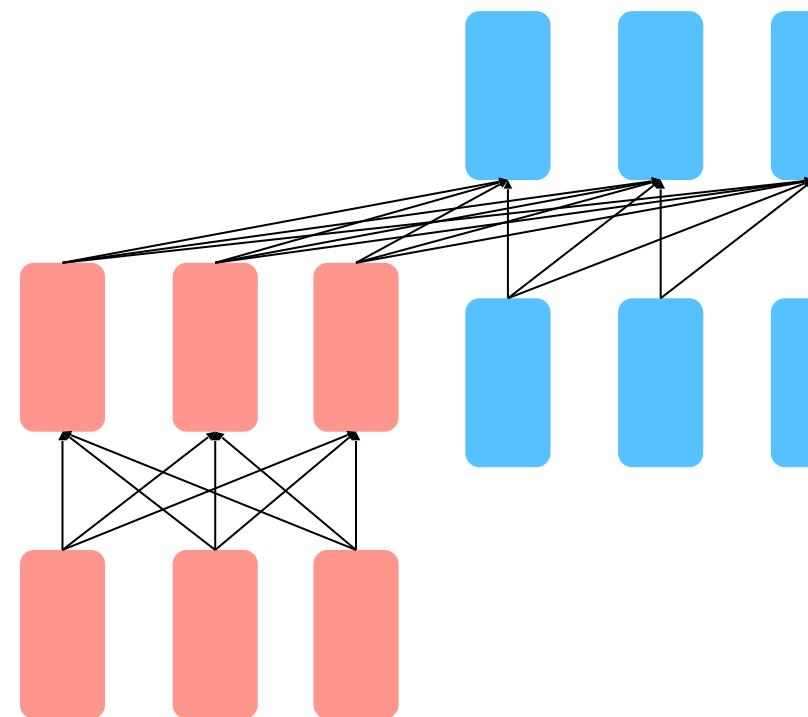
- Require more **text wrangling**
- **Harder to train**
- **Less flexible** for natural language generation

Pre-training architectures

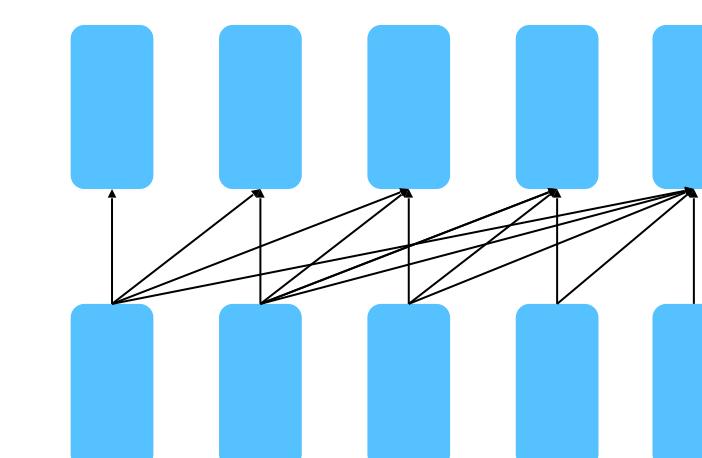
Encoder



Encoder-Decoder



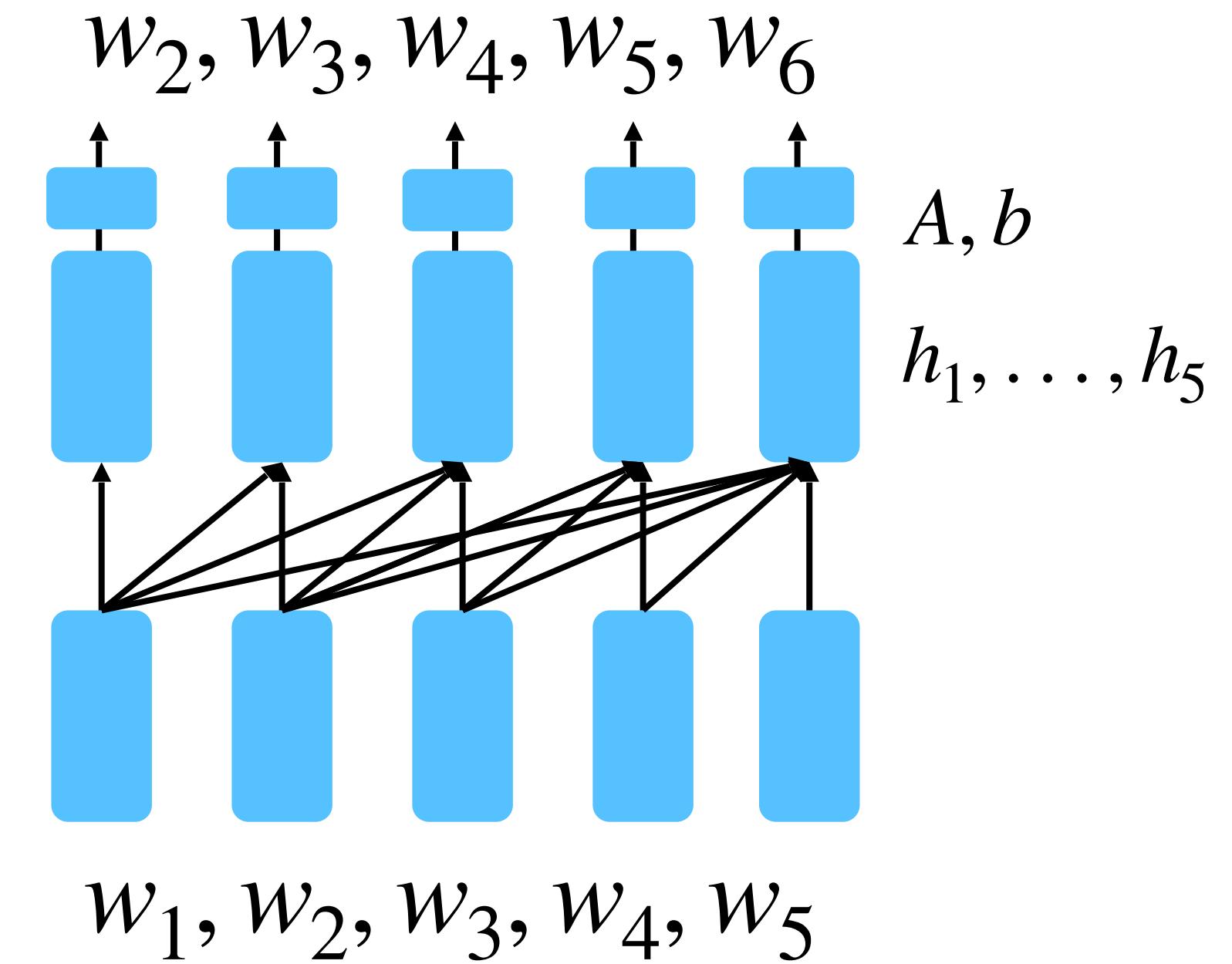
Decoder



- Bidirectional; can condition on the future context
- Map two sequences of different length together
- Language modeling; can only condition on the past context

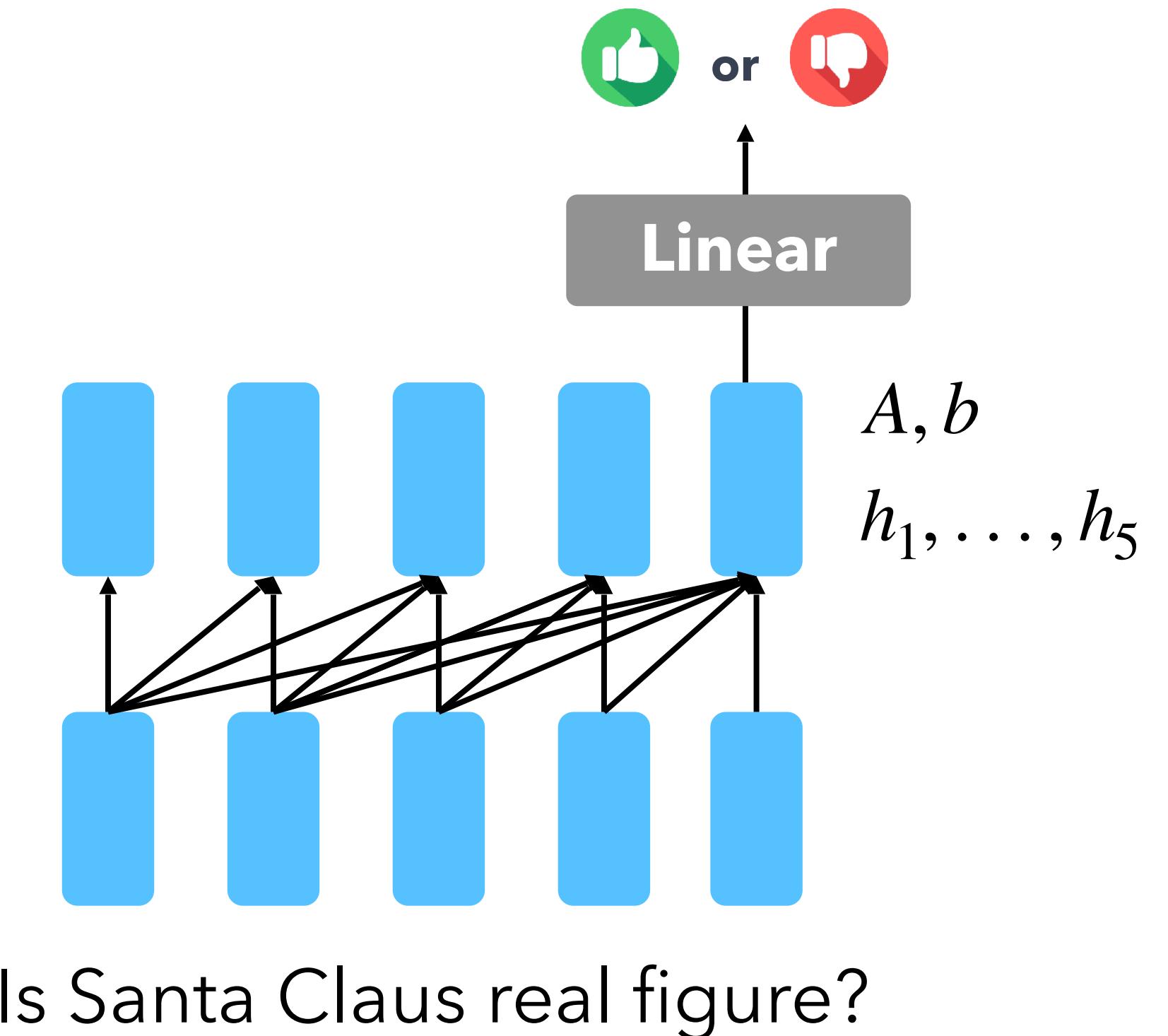
Decoder: training objective

- Many most famous generative LLMs are **decoder-only**
 - e.g., GPT1/2/3/4, Llama1/2
- **Language modeling!** Natural to be used for **open-text generation**
- **Conditional LM:** $p(w_t | w_1, \dots, w_{t-1}, x)$
 - Conditioned on a source context x to generate from left-to-right
- Can be fine-tuned for **natural language generation (NLG)** tasks, e.g., dialogue, summarization.



Decoder: training objective

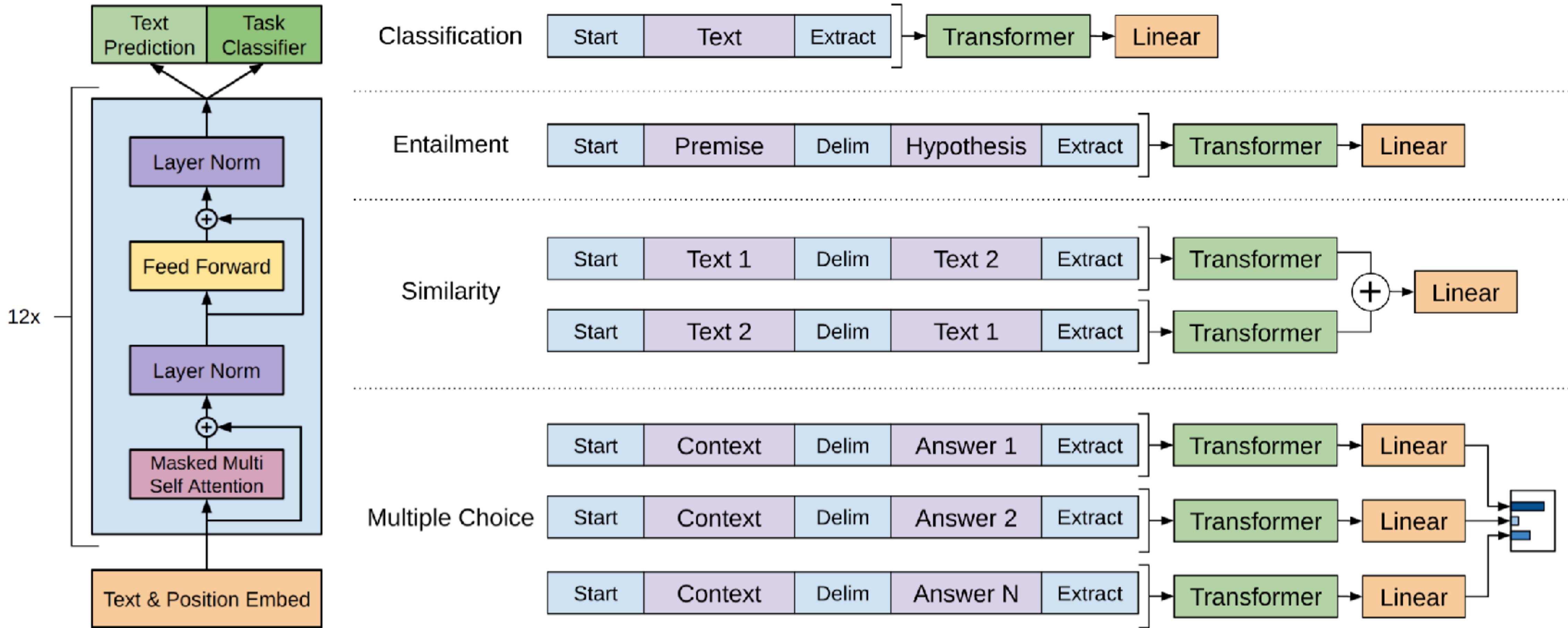
- Customizing the pre-trained model for downstream tasks:
 - Add a **linear layer** on top of the last hidden layer to make it a classifier!
 - During fine-tuning, trained the randomly **initialized linear layer**, along with **all parameters** in the neural net.



Decoder: GPT

Generative Pre-trained Transformer

[Radford et al., 2018]



How to use these pre-trained models?



Transformers

• **Transformers** ▾

Search documentation 8K

v4.27.2 EN ☀️ 92,354

- CANINE
- CodeGen
- ConvBERT
- CPM
- CTRL
- DeBERTa
- DeBERTa-v2
- DialoGPT
- DistilBERT**
- DPR
- ELECTRA

DistilBERT

All model pages distilbert Hugging Face Spaces

Overview

The DistilBERT model was proposed in the blog post [Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT](#), and the paper [DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter](#). DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than `bert-base-uncased`, runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark.

```
>>> from transformers import AutoTokenizer\n\n>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")\n\n>>> def tokenize_function(examples):\n...     return tokenizer(examples["text"], padding="max_length", truncation=True)\n\n>>> tokenized_datasets = dataset.map(tokenize_function, batched=True)\n\n>>> from transformers import AutoModelForSequenceClassification\n\n>>> model = AutoModelForSequenceClassification.from_pretrained("bert-base-cased", num_labels=5)
```

How to pick a proper architecture for a given task?

- Right now **decoder-only** models seem to dominate the field at the moment
 - e.g., GPT1/2/3/4, Mistral, Llama1/2
- T5 (seq2seq) works well with multi-tasking
- **Picking the best model architecture remains an open research question!**