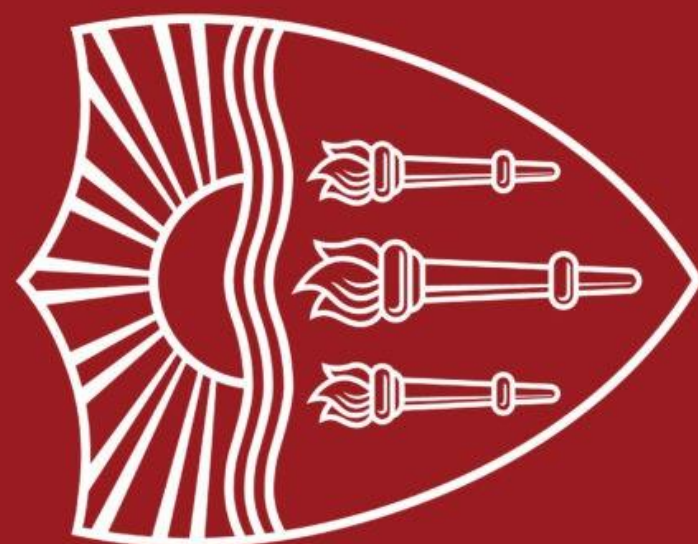


Lecture 14:

Language Generation and Tokenization

Instructor: Xiang Ren
USC CSCI 544 Applied NLP
Fall 2026



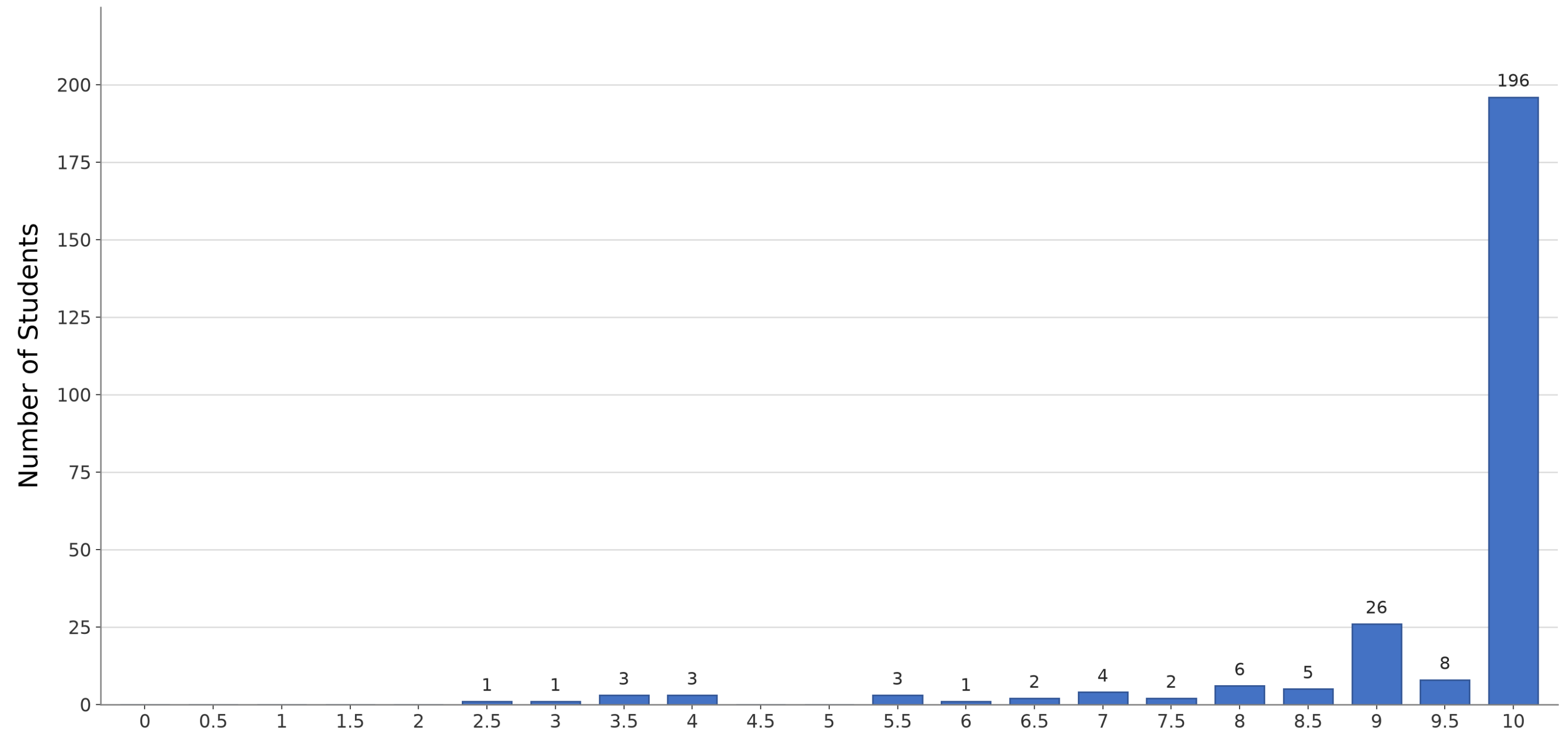
Announcements

- HW2 will be released on Friday.
- Tue 10/20: Project midterm presentation
- 10/27: Project midterm report due!

Quiz 3 Stats

219 Attended, 28
absent

Quiz 3 Score Distribution



Mean = 9.46; Std Dev = 1.33

Mid-term Presentation Rubric

The project midterm presentation (2min presentation with 2 slides + 1min FAQ) will be given by your team in class and graded on clarity, novelty, and presentation quality (by instructor and TA).

Mid-term Presentation Rubric

The grading rubric that you will follow for peer evaluations (via Google Form on the day of the class) and the instructors will follow as well will be:

(1 point) Clarity: How clearly did the team articulate their central problem statement?

1/1: The problem is explicit and well-motivated. The team explains inputs/outputs of the central task.

0.5/1: The topic is clear but the problem statement itself is too vague or high-level.

0/1: There is no identifiable problem statement:

(1 point) Novelty: How clearly did the team situate their project idea in comparison to prior work?

1/1: The team names specific prior work and articulates how their project goes beyond.

0.5/1: Prior work is referenced vaguely and not named/cited.

0/1: No prior work is mentioned.

Mid-term Presentation Rubric

The grading rubric that you will follow for peer evaluations (via Google Form on the day of the class) and the instructors will follow as well will be:

(2 point) Progress so far: How clearly did the team articulate the work they have done so far and what work remains?

2/2: The team shows concrete completed work (however small) and can clearly articulate what exactly needs to be done before the final report.

1/2: Some vague progress is described and/or the remaining work plan is unclear and not well-scoped.

0/2: Little or no progress has been shown and/or there is no plan for future work.

(1 point) Q&A: How well did the team answer audience questions?

1/1: Answers are direct, concise, and clear.

0.5/ 1: Answers are provided but not really relevant to the questions asked.

0/1: Team cannot answer any questions.

Lecture Outline

- Announcements
- Recap: The pre-training and fine-tuning paradigm
- Tokenization
- Natural Language Generation

Recap: Pre-training and Fine-tuning

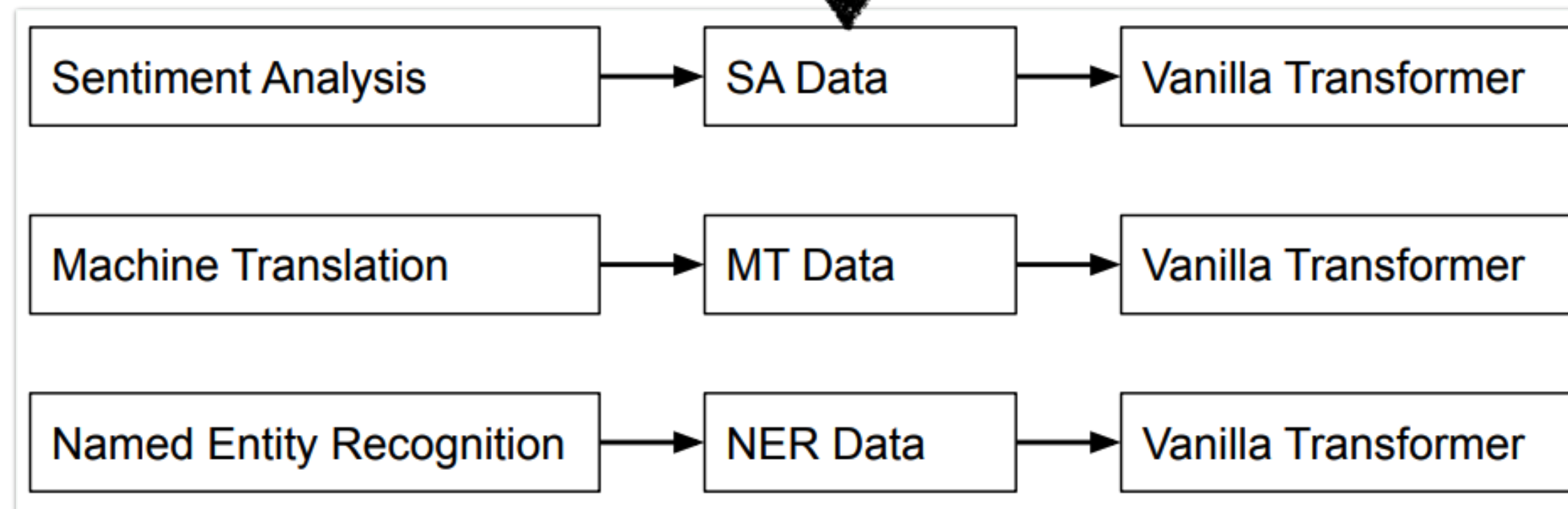
Traditional learning paradigm

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

Train Transformer or other models separately for each task

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

Supervised fine-tuning only
NO pre-training

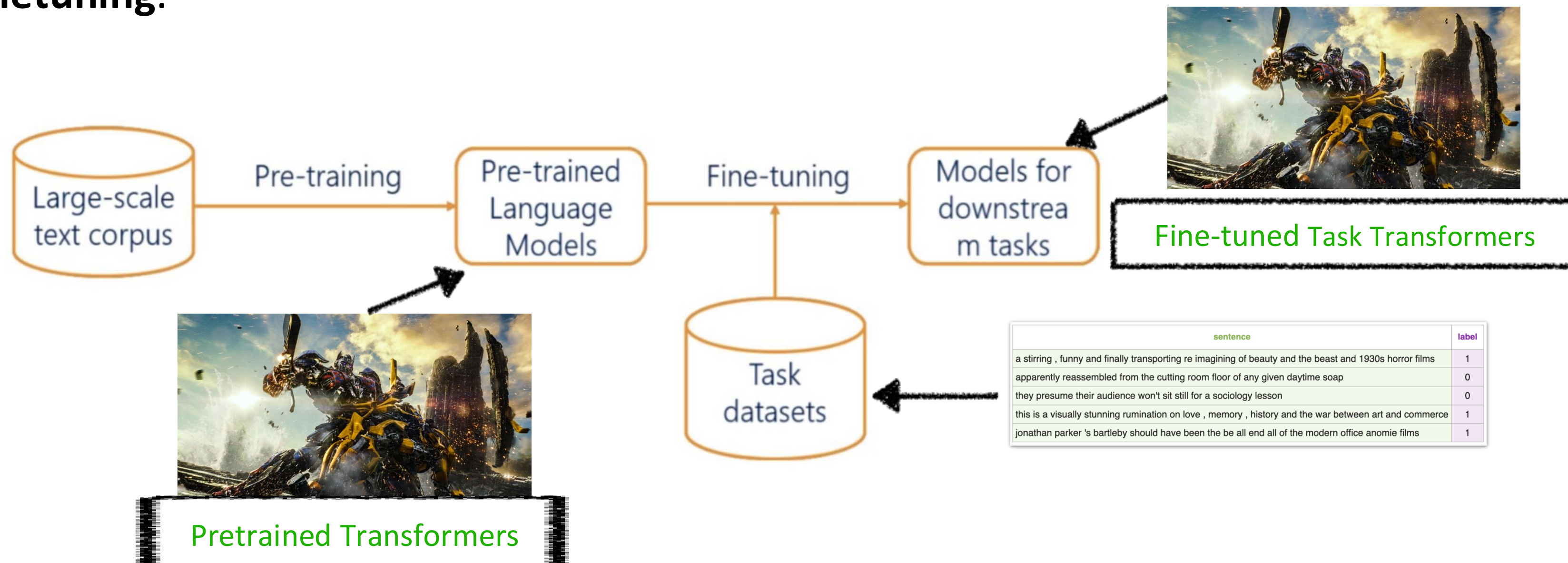


Modern learning paradigm

Pre-training + supervised training/fine-tuning

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

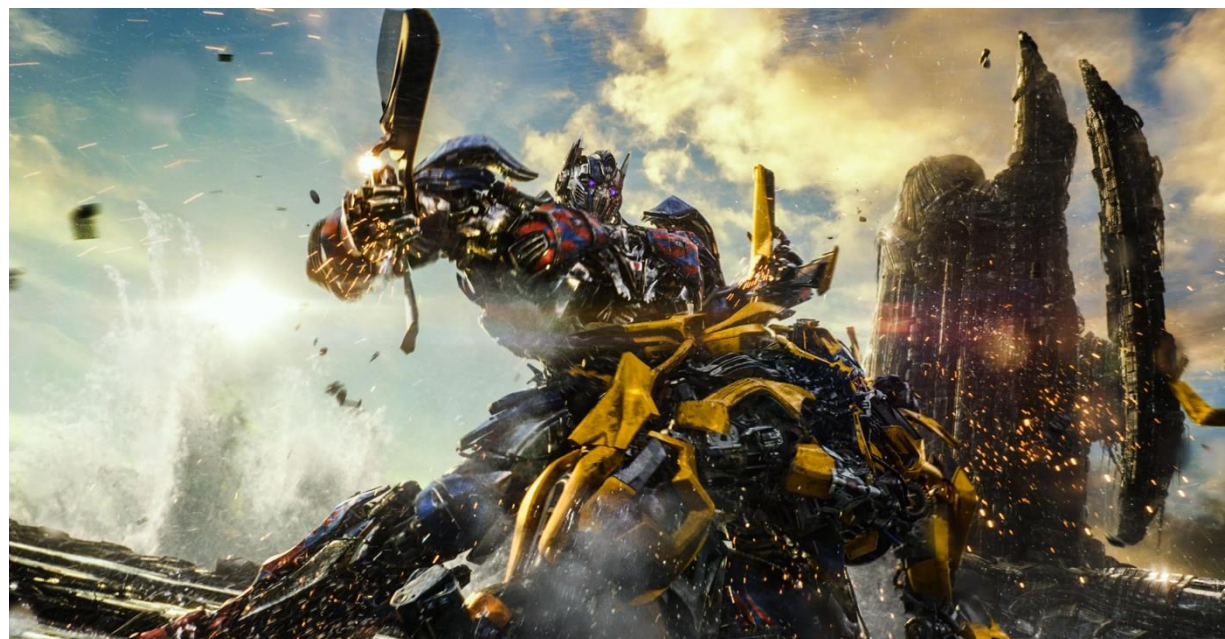
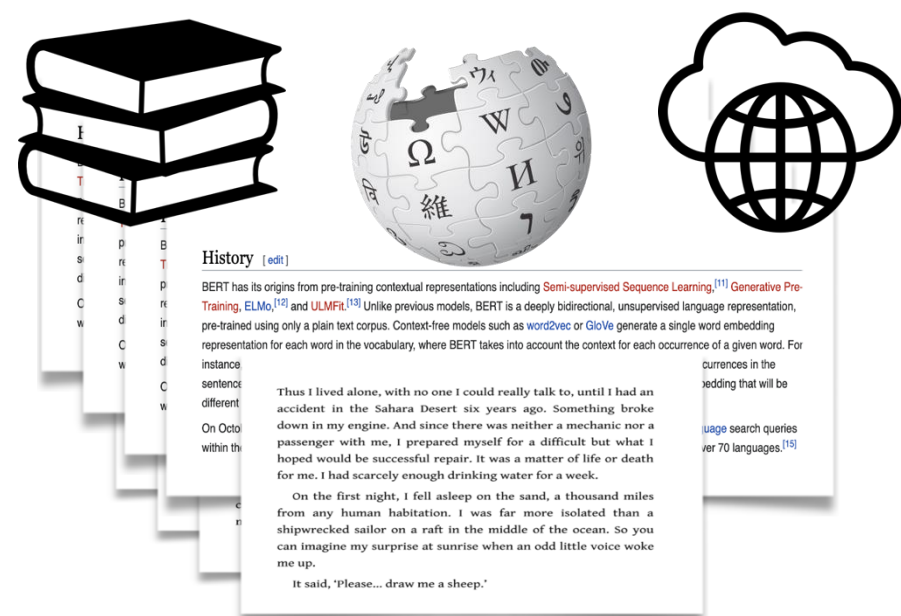
Then train the pretrained Transformer for a specific task using supervised learning. This is called **finetuning**.



Pretraining: training objectives?

During pretraining, we have a large text corpus (**no task labels**)

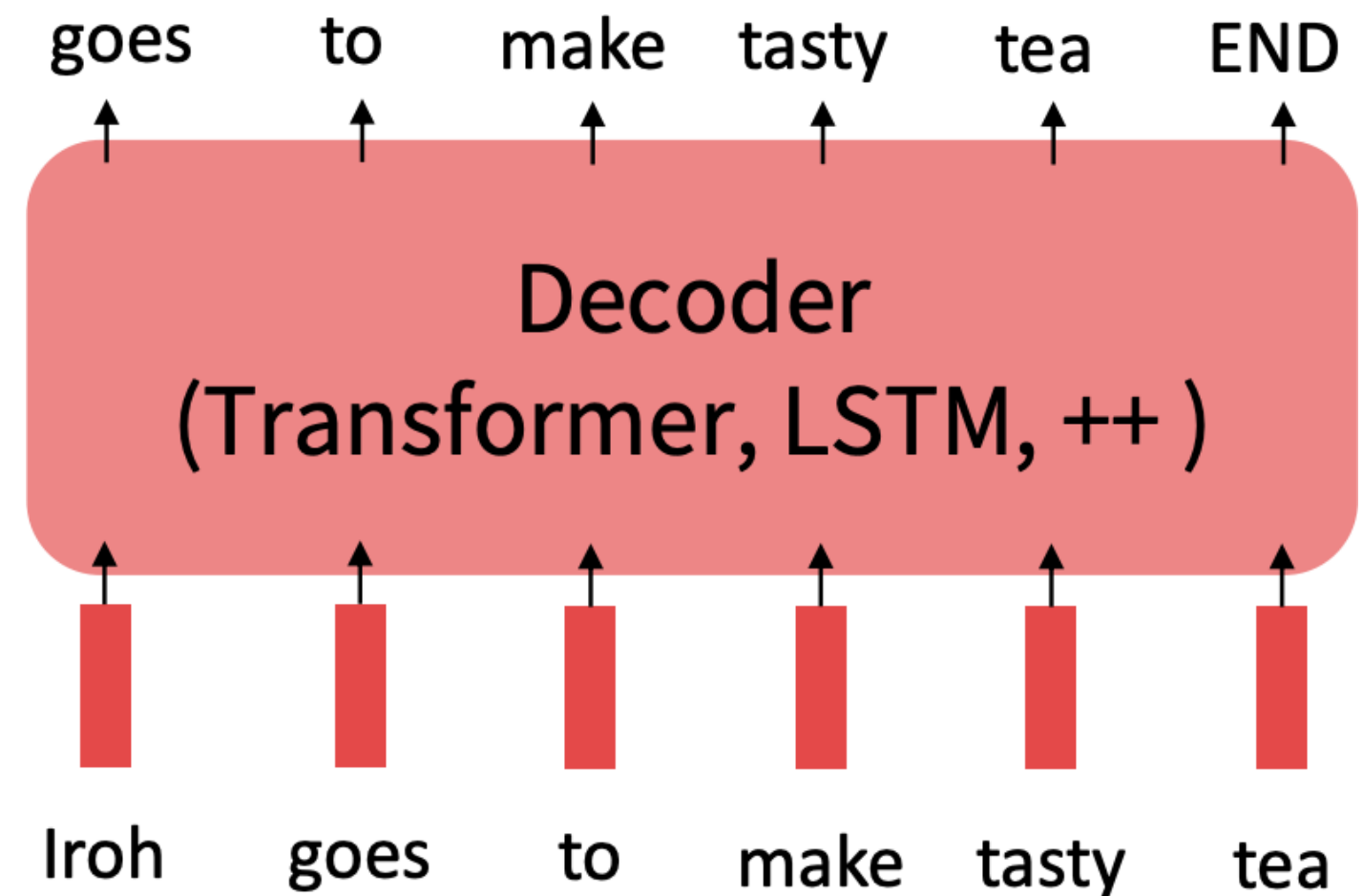
- **Key question: what labels or objectives used to train the vanilla Transformers?**



Training labels/objectives?

Pretraining: Language Models

- Recall the language modeling task:
 - Model $p_{\theta}(w_t | w_{1:t-1})$, the probability distribution over words given their past contexts.
 - There's lots of data for this! (In English.)
- Pretraining through language modeling:
 - Train a neural network to perform language modeling on a large amount of text.
 - Save the network parameters.



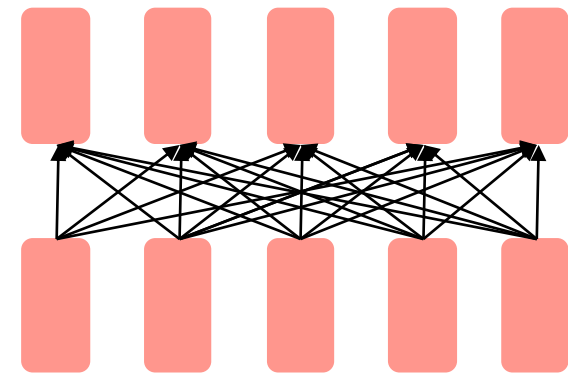
Semi-supervised Sequence Learning

Andrew M. Dai
Google Inc.
adai@google.com

Quoc V. Le
Google Inc.
qvl@google.com

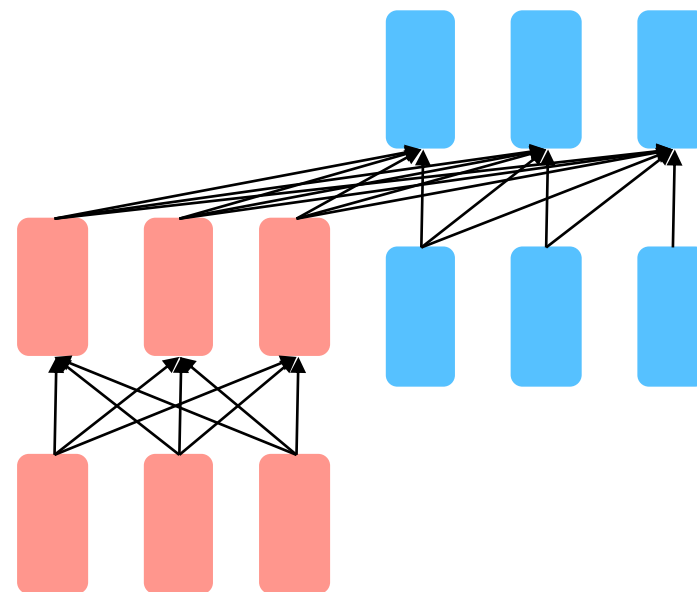
Pre-training architectures

Encoder



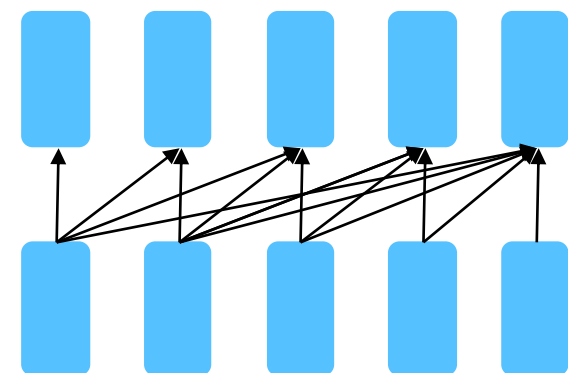
- Bidirectional; can condition on the future context

Encoder-Decoder



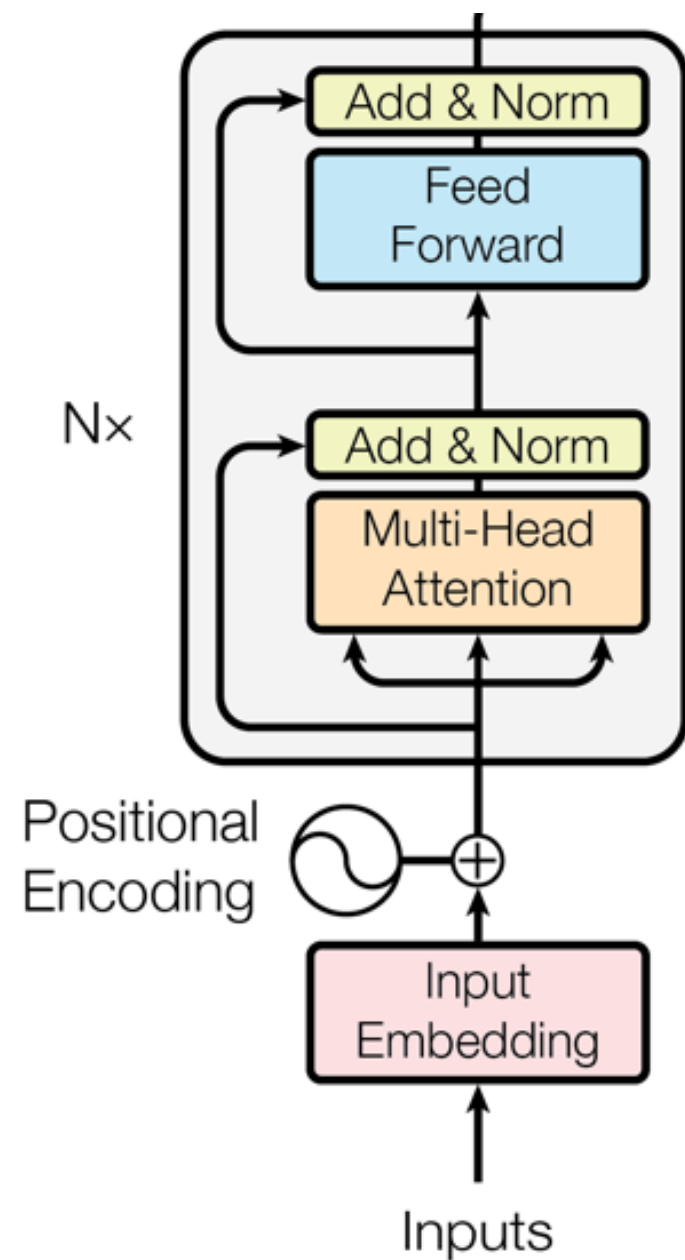
- Map two sequences of different length together

Decoder



- Language modeling; can only condition on the past context

BERT: Bidirectional Encoder Representations from Trans (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..

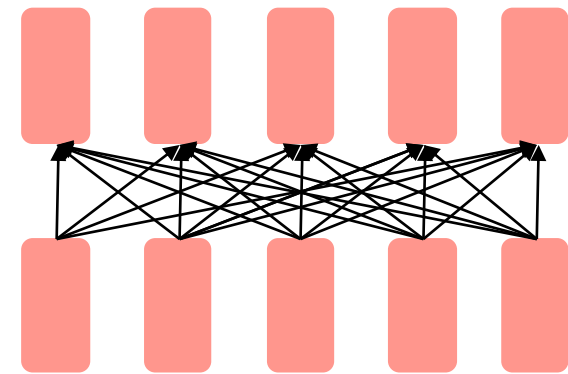


Use the output of the masked word's position to predict the masked word



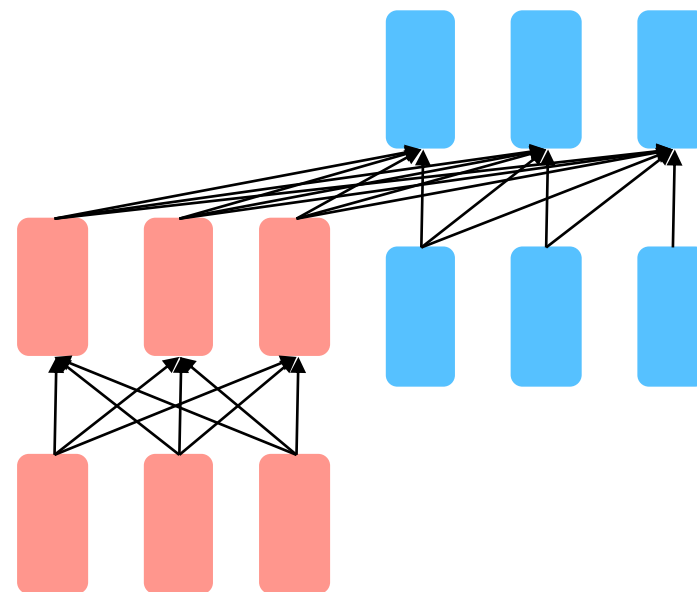
Pre-training architectures

Encoder



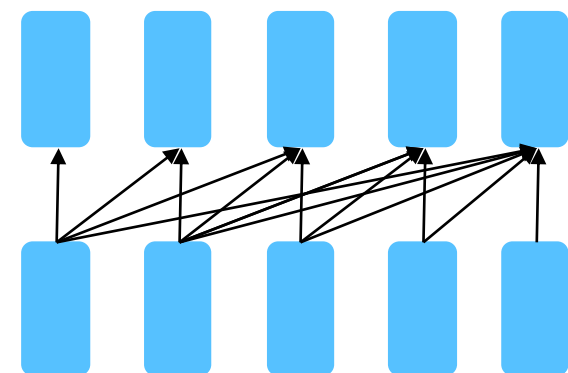
- Bidirectional; can condition on the future context

Encoder-Decoder



- Map two sequences of different length together

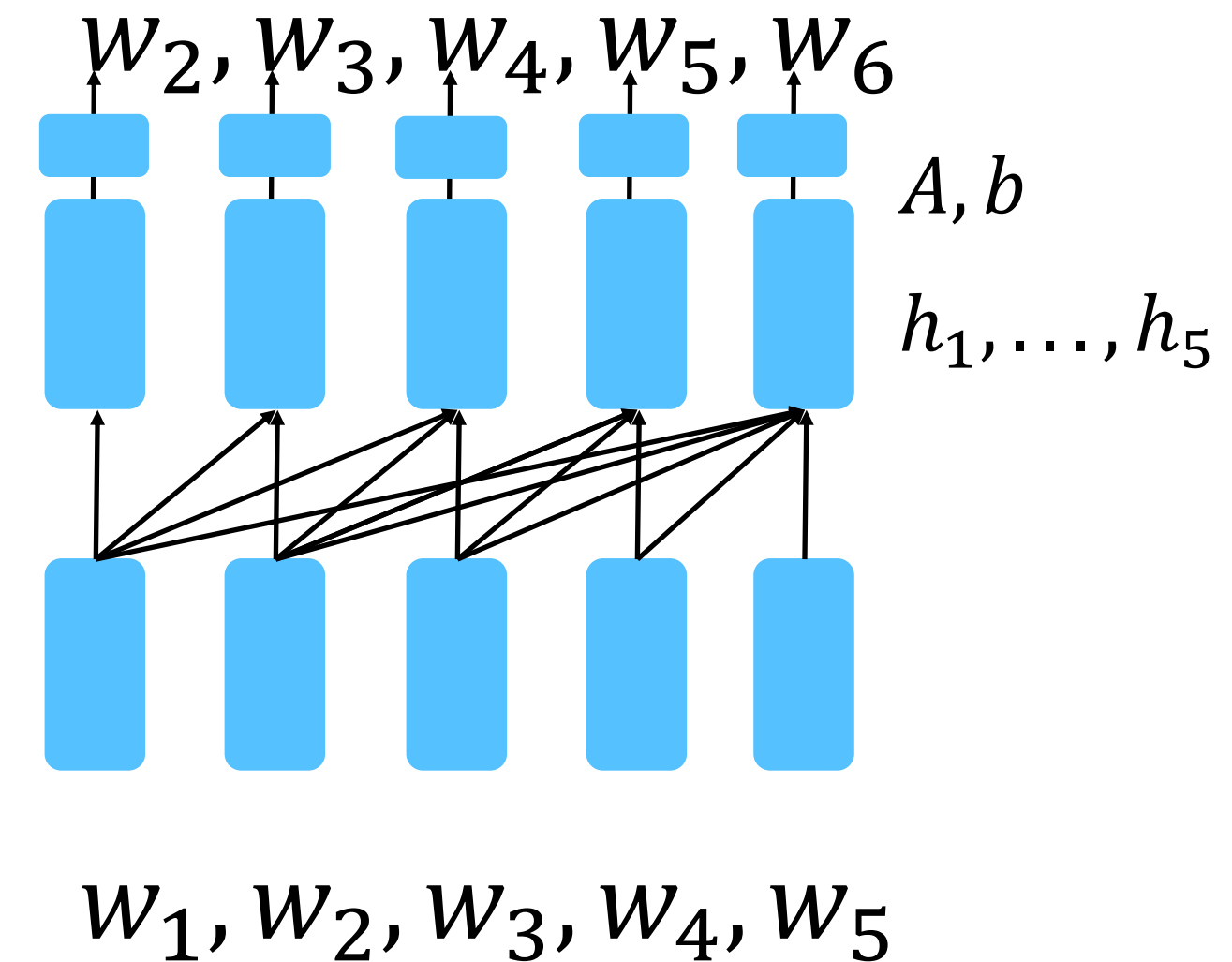
Decoder



- Language modeling; can only condition on the past context

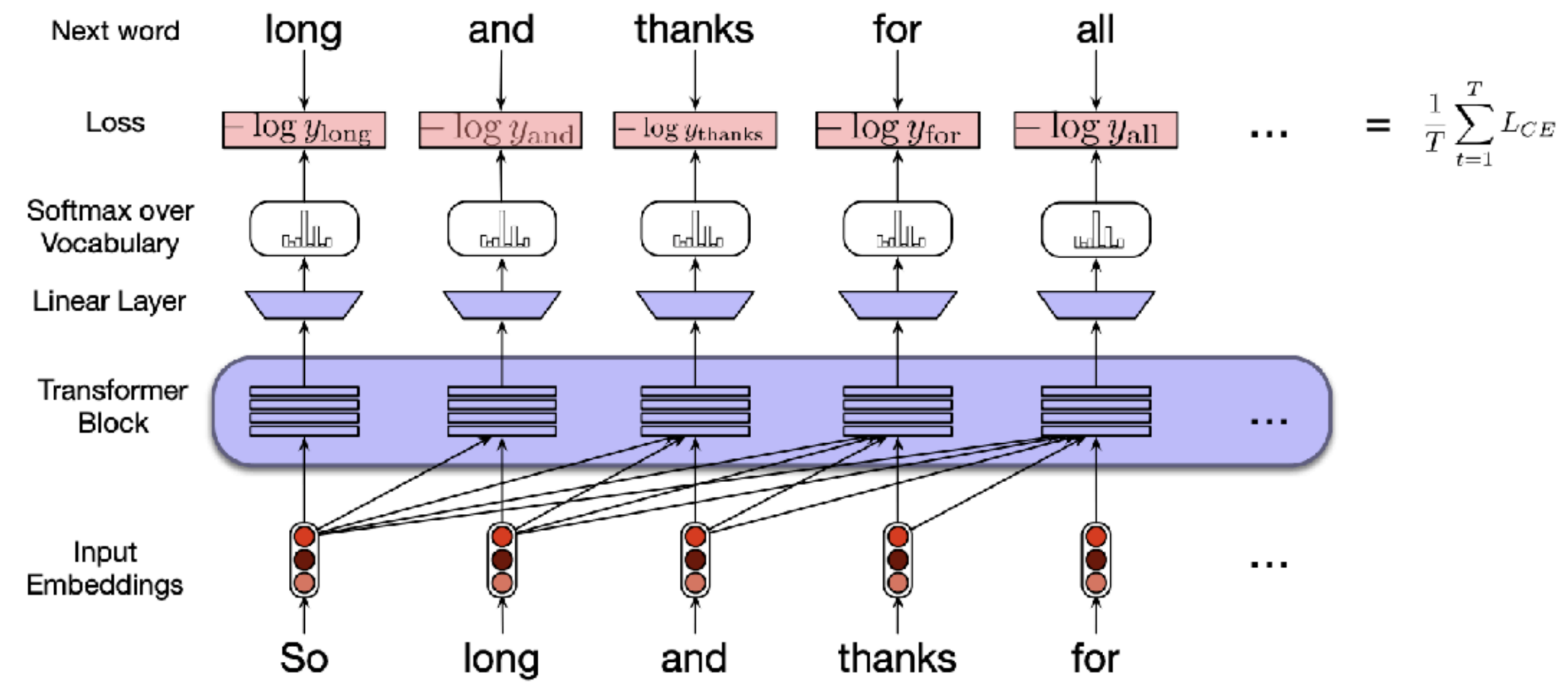
Decoder: training objective

- Many most famous generative LLMs are **decoder-only**
 - e.g., GPT1-5, Llama1-3
- **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 $\mathcal{X}$ to generate from left-to-right
- Can be fine-tuned for **natural language generation (NLG)** tasks, e.g., dialogue, summarization.



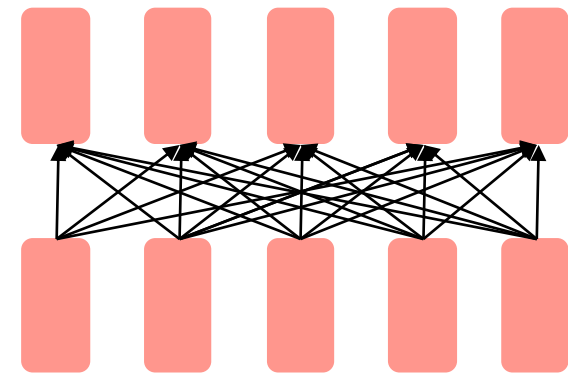
Generative Pre-Training (GPT)

- Use a **Transformer decoder** (unidirectional; left-to-right) instead of LSTMs
- Use **language modeling** as a pre-training objective
- Trained on longer segments of text (**512 BPE tokens**), not just single sentences



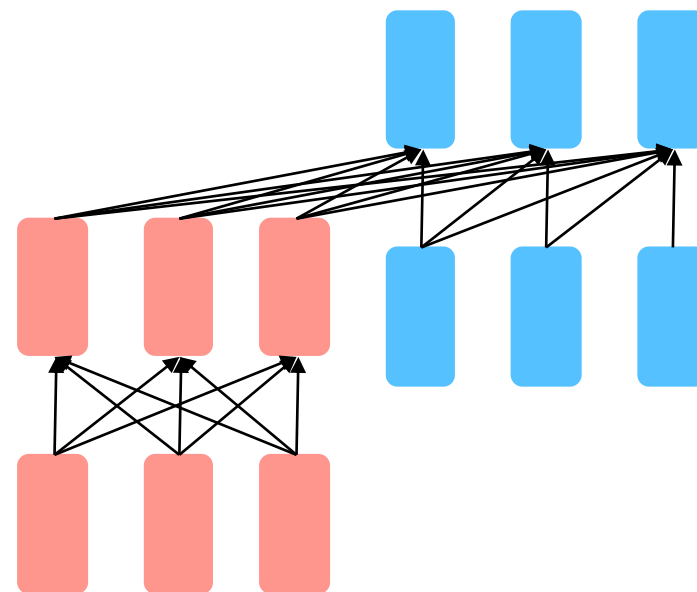
Pre-training architectures

Encoder



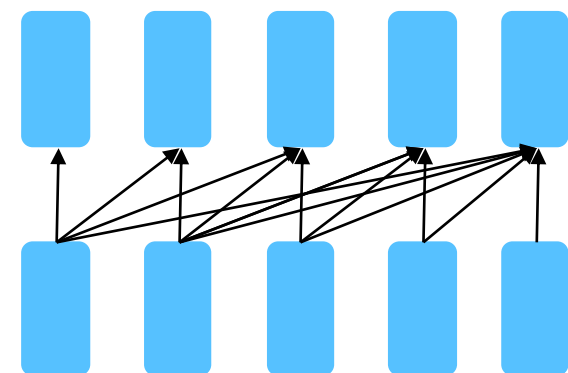
- Bidirectional; can condition on the future context

Encoder-Decoder



- Map two sequences of different length together

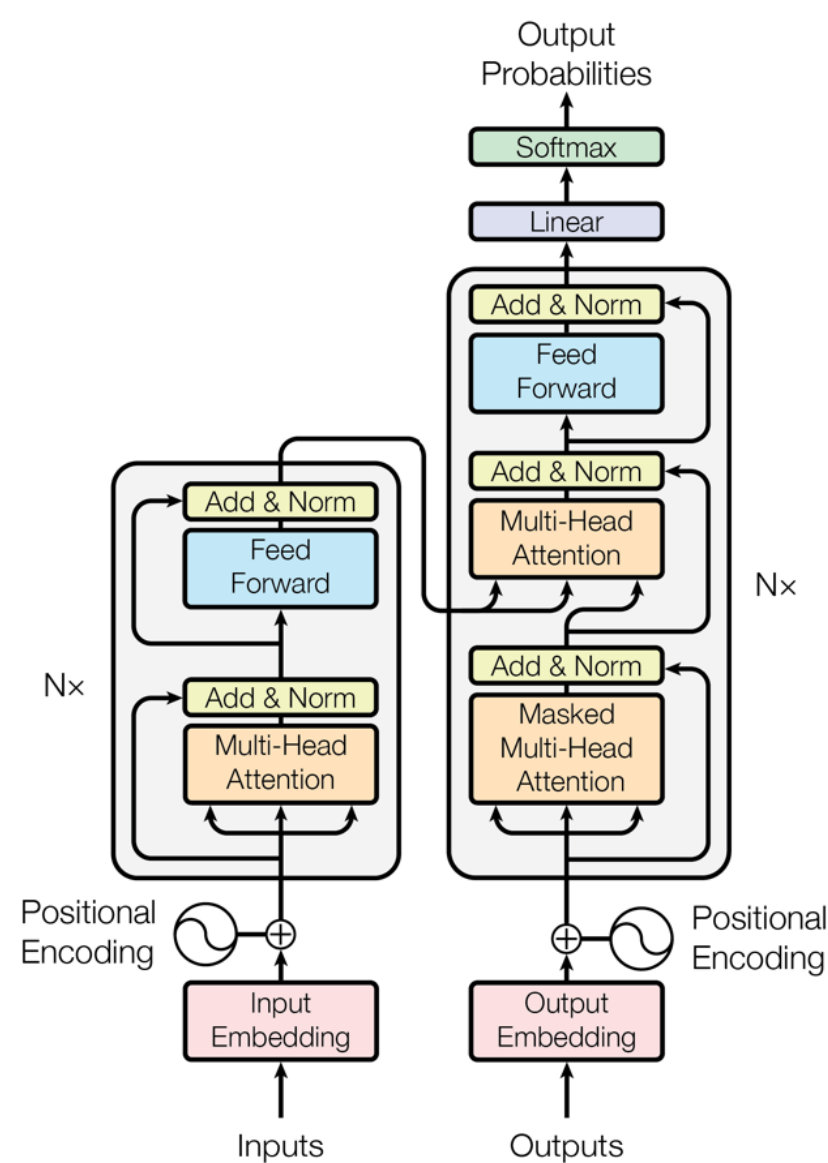
Decoder



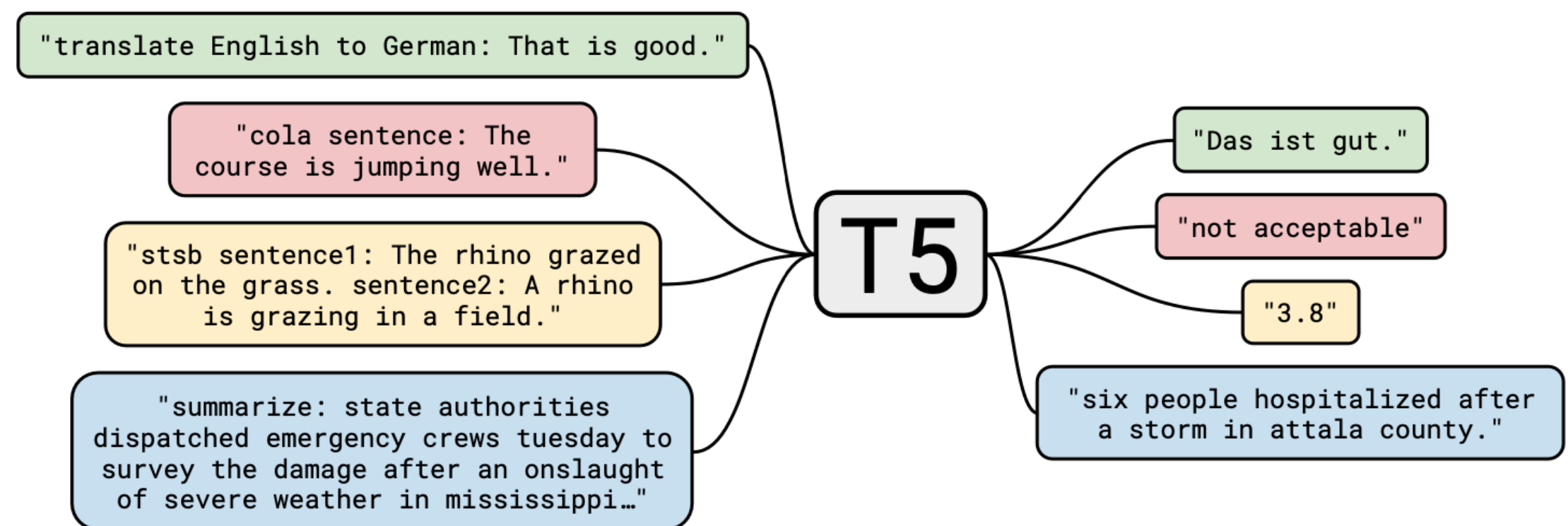
- Language modeling; can only condition on the past context

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!**

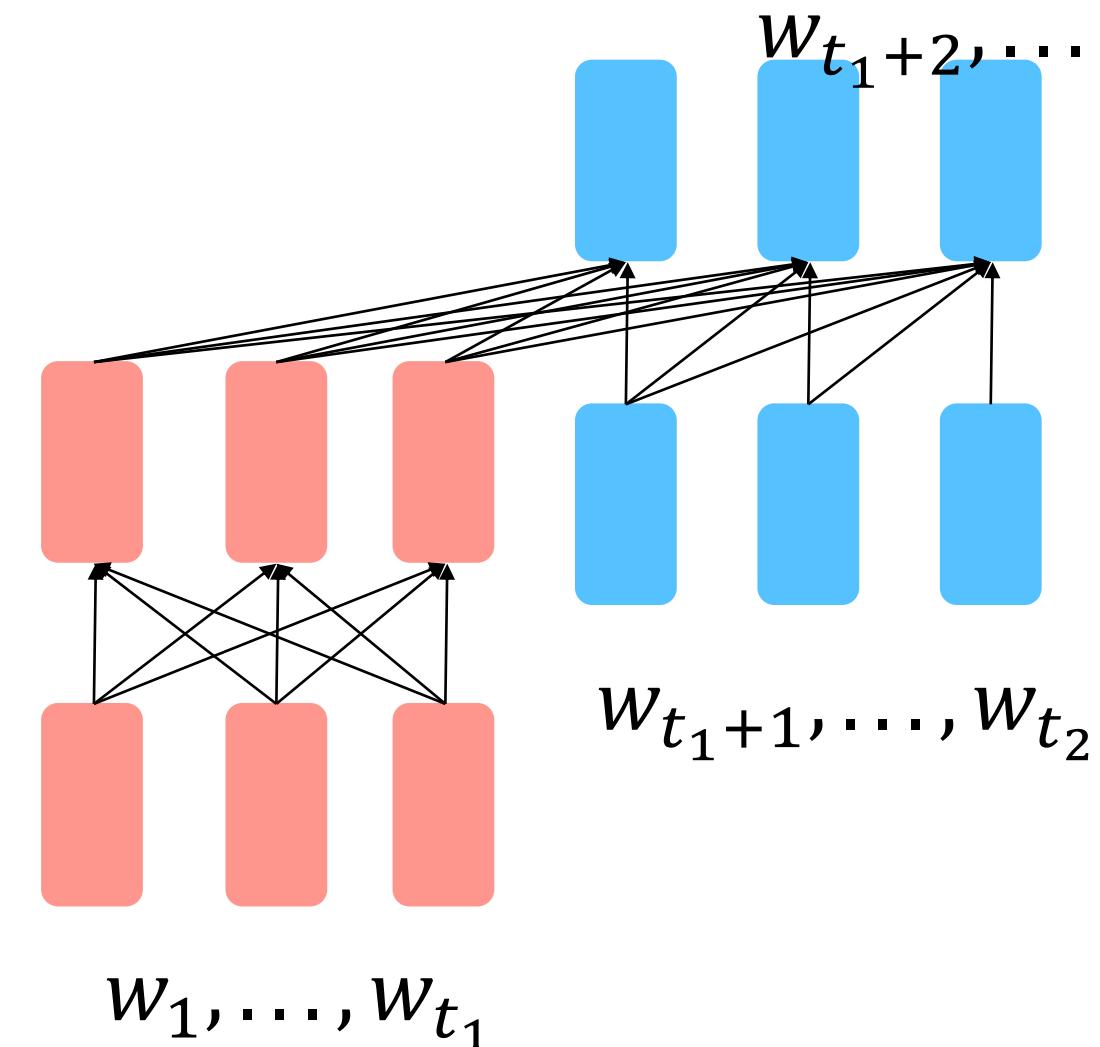


T5 = Text-to-Text Transfer Transformer



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**

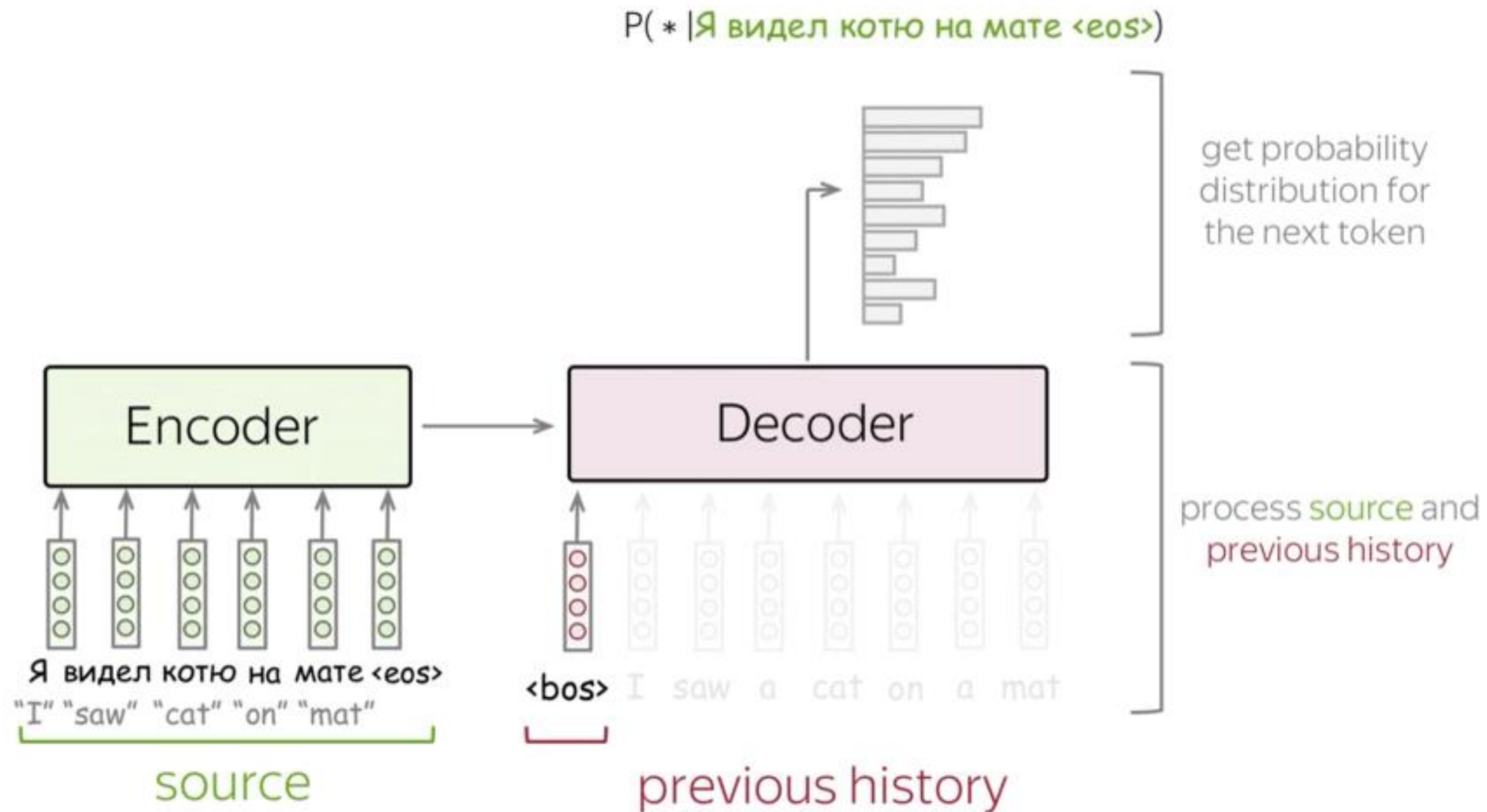


$$h_1, \dots, h_{t_1} = \text{Encoder}(w_1, \dots, w_{t_1})$$

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

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

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

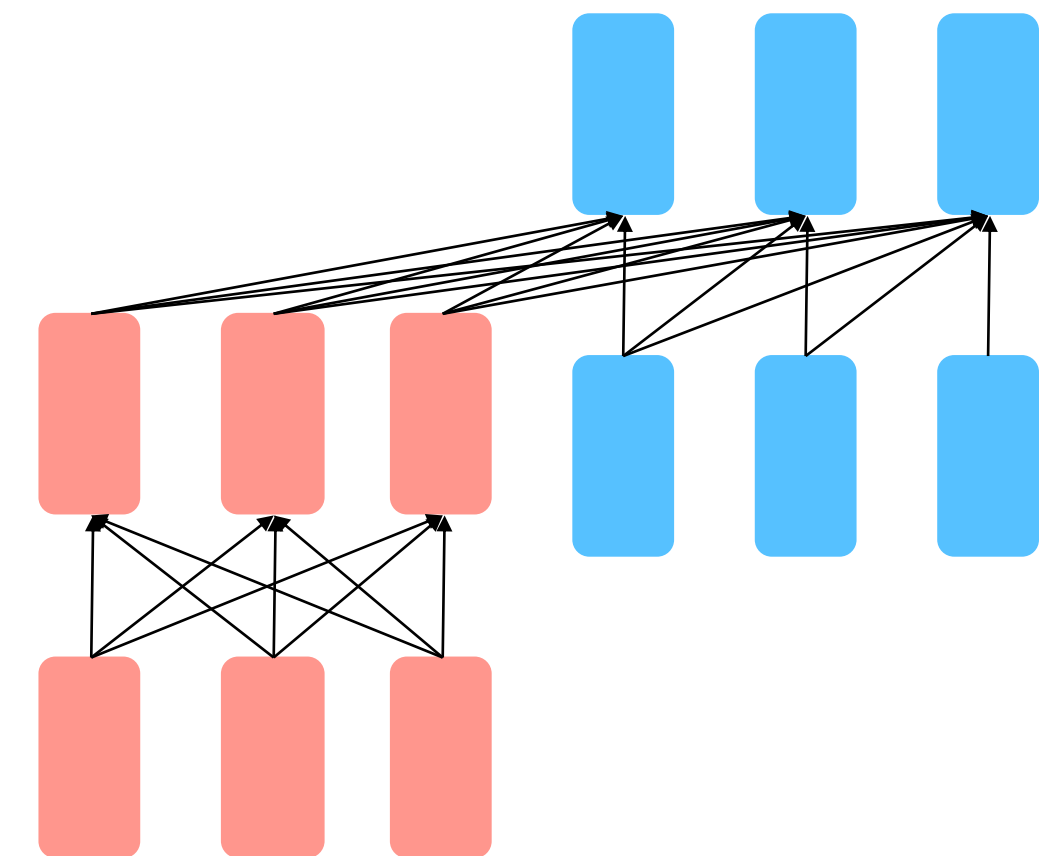
Thank you ~~for inviting~~ me to your party ~~last~~ week.

Inputs

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

Targets

`<X>` for inviting `<Y>` last `<Z>`



Encoder-decoder: T5

[[Raffel et al., 2018](#)]

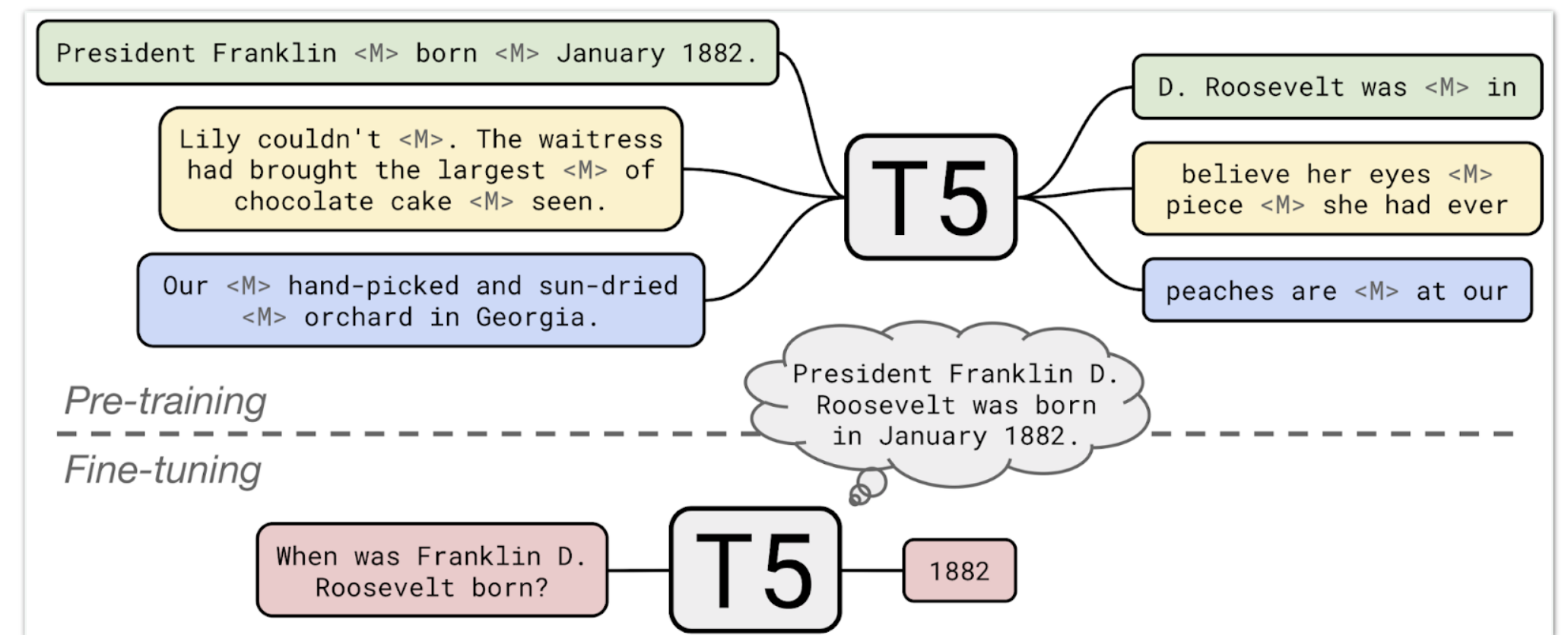
- **Encoder-decoders** works better than decoders
- **Span corruption (denoising)** objective works better than language modeling

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

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

Original text

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

Inputs

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

Targets

<X> for inviting <Y> last <Z>

Denoising span-mask prediction



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

Generated training examples

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
...		

Next token prediction

How to use these pre-trained models?

Transformers

Transformers

Search documentation

🔍

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

>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")

>>> def tokenize_function(examples):
...     return tokenizer(examples["text"], padding="max_length", truncation=True)

>>> tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

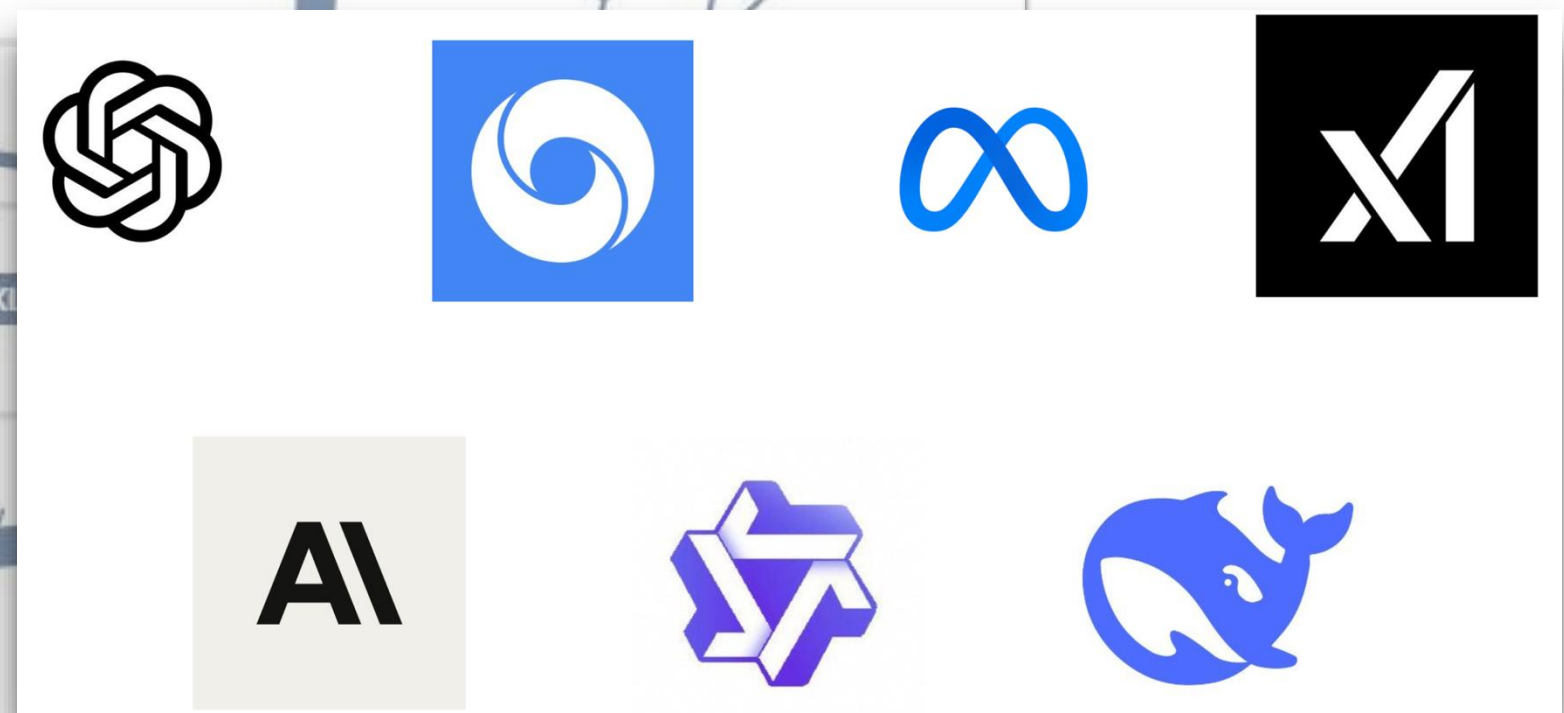
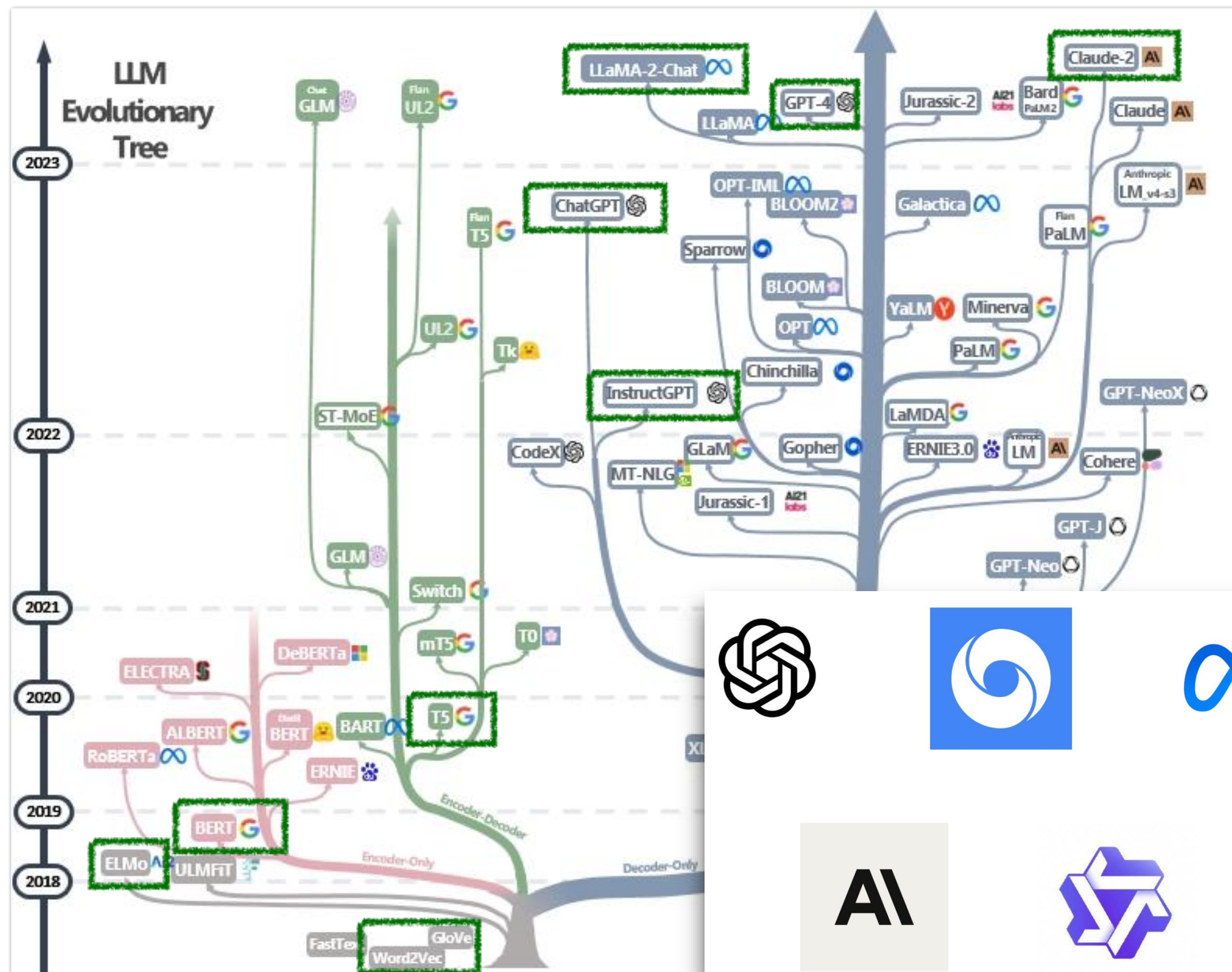
```
>>> from transformers import AutoModelForSequenceClassification

>>> 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 dominant the field at the moment
 - e.g., GPT, Opus, Qwen, Llama
- T5 (seq2seq) works well with multi-tasking
- **Picking the best model architecture remains an open research question!**

Evolution tree of pretrained LMs



Latest learning paradigm with LLMs

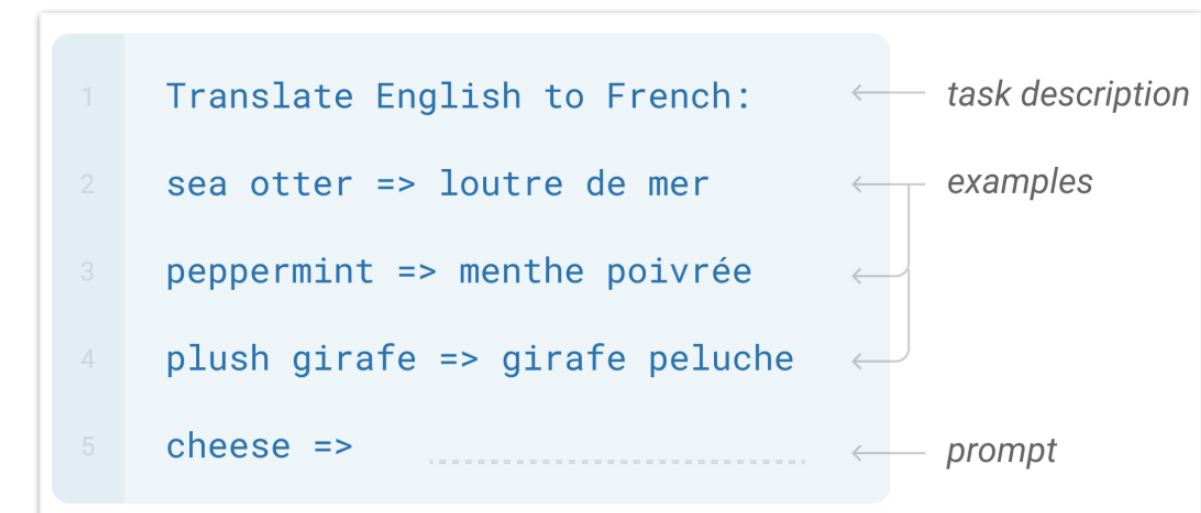
- Pre-training + prompting/in-context learning (no training this step)
 - First train a **large (>7~175B)** Transformer using a lot of general text using unsupervised learning. This is called **large** language model **pretraining**.

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**.



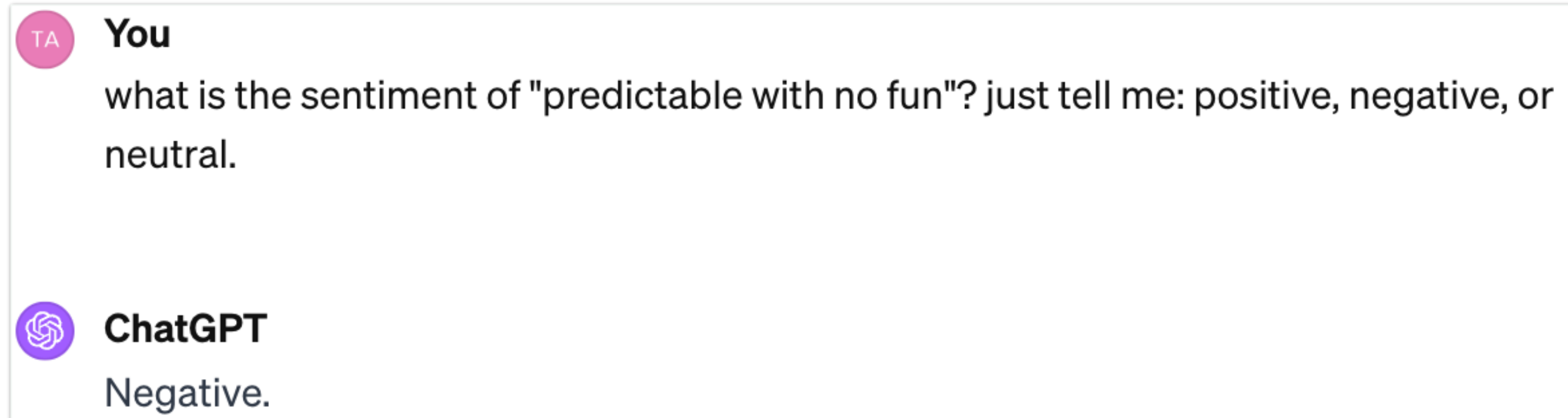
Zero-shot prompting



Few-shot prompting/in-context learning

Example: Prompting ChatGPT for sentiment analysis

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



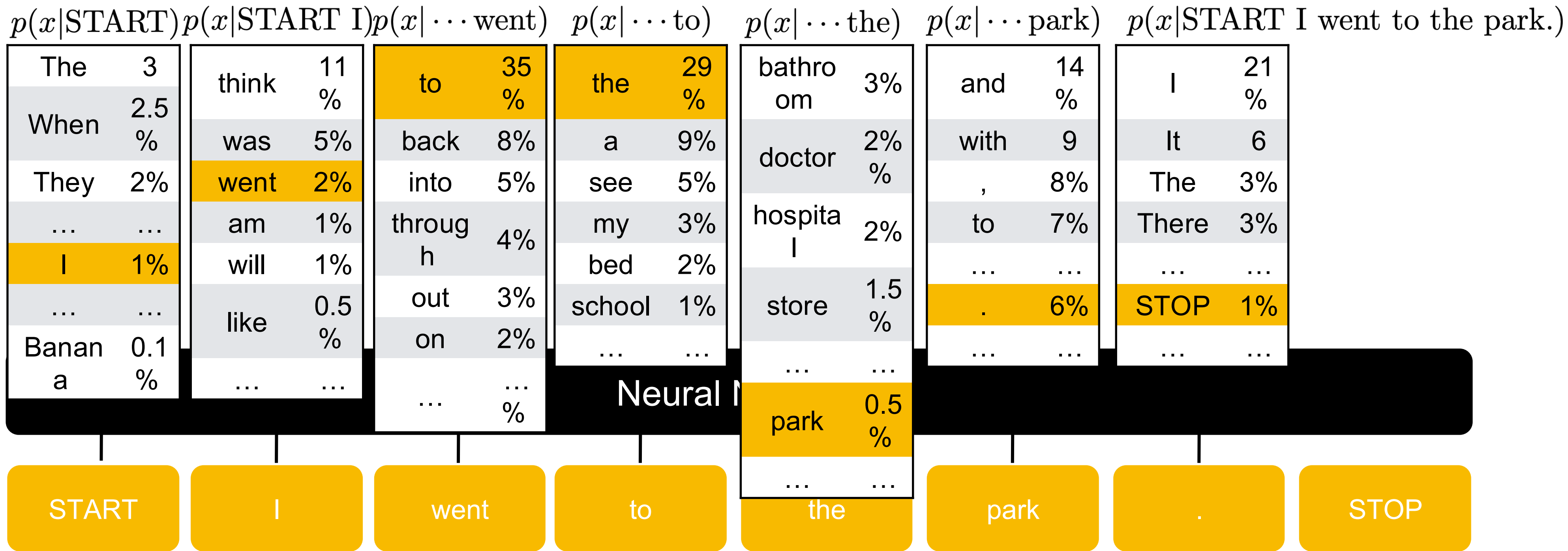
Already pretrained ChatGPT
No further training for sentiment analysis
Just prompting to conduct the task!



Tokenization

Neural language models: inputs/outputs

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



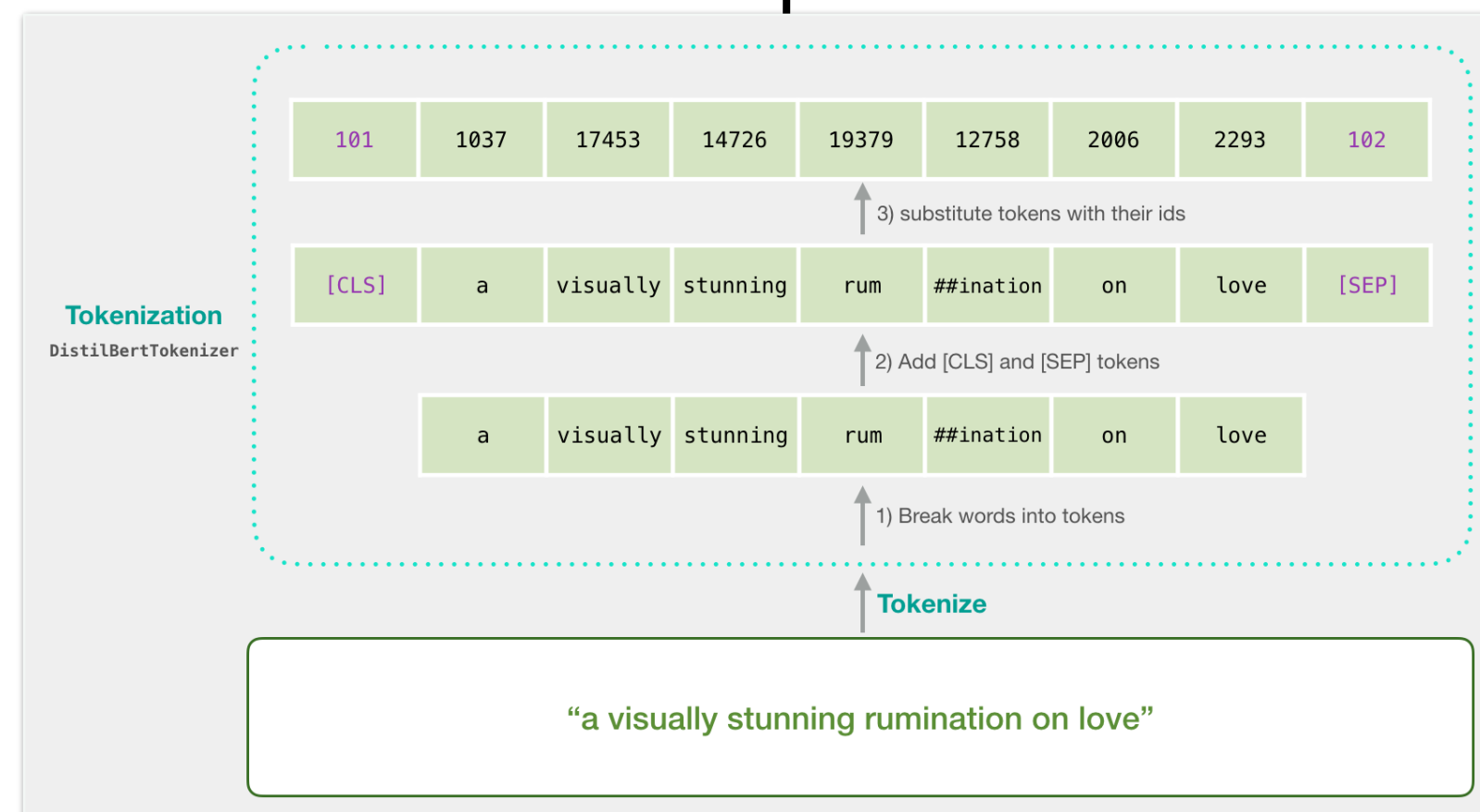
Tokenization to input vectors

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

Neural Network

Mapping each tokenized id into its corresponding embeddings

Tokenization:



START

I

went

to

the

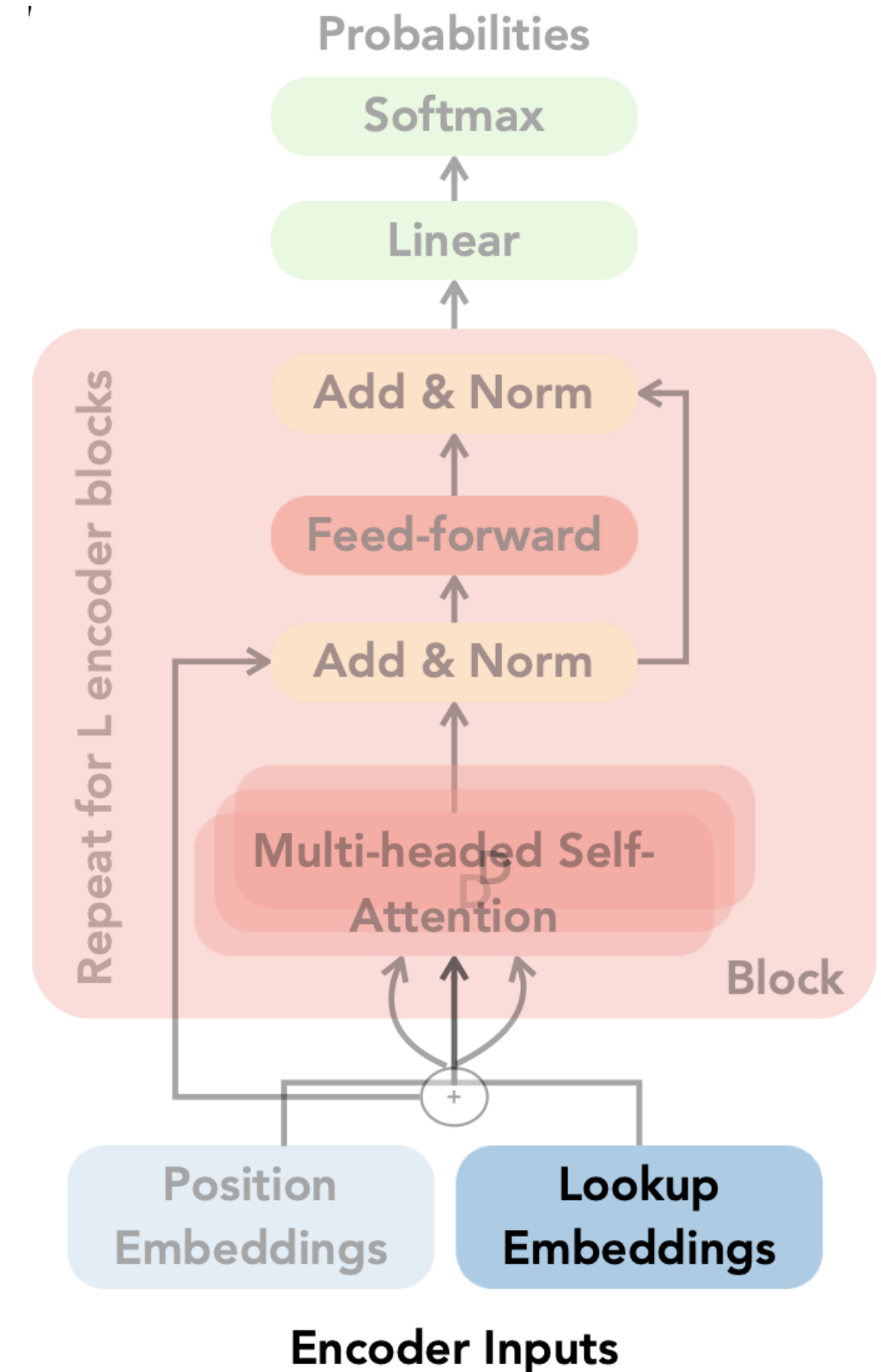
park

.

STOP

The Input Layer

- So far, we have made some assumptions about a language's vocabulary
- We assume a fixed vocabulary of tens of thousands of words, built from the training set
- However, even with the largest vocabulary, we may encounter out-of-vocabulary words at test time
- Our approach so far: map novel words seen at test time (OOV) to a single UNK



How to get the words?

Or, more accurately, the tokens?

- **Problem:** break the text into a sequence of discrete tokens
- For alphabetic languages such as English, deterministic scripts usually suffice to achieve accurate tokenization
- However, in languages such as Chinese and Swahili, words are typically composed of a small number of characters, without intervening whitespace

Word Structure in Language

- Finite vocabulary assumptions make even less sense in many languages.
 - Many languages exhibit complex morphology, or word structure.
 - The effect is more word types, each occurring fewer times.

-ambia = to tell

Example: Swahili verbs can have hundreds of conjugations, each encoding a wide variety of information. (Tense, mood, definiteness, negation, information about the object, ++)

Conjugation of -ambia																			[less ▲]		
Form Infinitive Positive form Imperative Habitual					Non-finite forms																
					Positive kuambia							Negative kutoambia									
					Simple finite forms																
					Singular ambia							Plural ambieni									
					huambia																
Complex finite forms																					
Polarity	Persons				Persons / Classes		Classes														
	1st		2nd		3rd / M-wa		M-mi		Ma		Ki-vi		N		U		Ku		Pa		Mu
	Sg.	Pl.	Sg.	Pl.	Sg. / 1	Pl. / 2	3	4	5	6	7	8	9	10	11 / 14	15 / 17	16	18			
Past																			[less ▲]		
Positive	niliambia naliambia	tuliambia twaliambia	uliambia waliambia	mliambia mwaliambia	aliambia	waliambia	uliambia	iliambia	liliambia	yaliambia	kiliambia	viliambia	iliambia	ziliambia	uliambia	kuliambia	paliambia	muliambia			
Negative	sikuambia	hatukuambia	hukuambia	hamkuambia	hakuambia	hawakuambi a	haukuambia	haikuambia	halikuambia	hayakuambi a	hakikuambia	havikuambia	haikuambia	hazikuambia	haukuambia	hakukuambi a	hapakuambi a	hamukuambi a			
Present																			[less ▲]		
Positive	ninaambia naambia	tunaambia	unaambia	mnaambia	anaambia	wanaambia	unaambia	inaambia	linaambia	yanaambia	kinaambia	vinaambia	inaambia	zinaambia	unaambia	kunaambia	panaambia	munaambia			
Negative	siambii	hatuambii	huambii	hamambii	haambii	hawaambii	hauambii	haiambii	haliambii	hayaambii	hakiambii	haviambii	haiambii	haziambii	hauambii	hakuambii	hapaambii	hamuambii			
Future																			[less ▲]		
Positive	nitaambia	tutaambia	utaambia	mtaambia	ataambia	wataambia	utaambia	itaambia	litaambia	yataambia	kitaambia	vitaambia	itaambia	zitaambia	utaambia	kutaambia	pataambia	mutaambia			
Negative	sitaambia	hatutaambia	hutaambia	hamtaambia	hataambia	hawataambi a	hautaambia	haitaambia	halitaambia	hayataambia	hakitaambia	havitaambia	haitaambia	hazitaambia	hautaambia	hakutaambia	hapataambia	hamutaambi a			
Subjunctive																			[less ▲]		
Positive	niambie	tuambie	uambie	mambie	aambie	waambie	uambie	iambie	liambie	yaambie	kiambie	viambie	iambie	ziambie	uambie	kuambie	paambie	muambie			
Negative	nisiambie	tusiambie	usiambie	msiambie	asiambie	wasiambie	usiambie	isiambie	lisiambie	yasiambie	kisiambie	visiambie	isiambie	zisiambie	usiambie	kusiambie	pasiambie	musiambie			
Present Conditional																			[less ▲]		
Positive	ningeambia	tungeambia	ungeambia	mngeambia	angeambia	wangeambia	ungeambia	ingeambia	lingeambia	yangeambia	kingeambia	vingeambia	ingeambia	zingeambia	ungeambia	kungeambia	pangeambia	mungeambia			
Negative	nisingeambi a singeambia	tusingeambi a hatungeambi a	usingeambi a hungeambia	msingeambi a hamngeambi a	asingeambi a hangeambia	wasingeambi a hawangeambi a	usingeambi a haungeambi a	isingeambi a haingeambia	lisingeambi a halingeambi a	yasingeambi a hayangeambi a	kisingeambi a hakingeambi a	vingeambi a havingeambi a	isingeambi a haingeambi a	zisingeambi a hazingeambi a	ungeambi a haungeambi a	kungeambi a hakungeambi a	pasingeambi a hapangeambi a	mungeambi a hamungeambi a			
Past Conditional																			[less ▲]		
Positive	ningaliambia	tungaliambia	ungaliambia	mngaliambia	angaliambia	wangaliambi a	ungaliambia	ingaliambia	lingaliambia	yangaliambi a	kingaliambia	vingaliambia	ingaliambia	zingaliambia	ungaliambia	kungaliambi a	pangaliambi a	mungaliambi a			
Negative	nisingaliambi a singaliambia	tusingaliambi a hatungaliambi a	usingaliambi a hungaliambi a	msingaliambi a hamngaliambi a	asingaliambi a hangaliambi a	wasingaliambi a hawangaliambi a	usingaliambi a haungaliambi a	isingaliambi a haingaliambi a	lisingaliambi a halingaliambi a	yasingaliambi a hayangaliambi a	kisingaliambi a hakingaliambi a	vingaliambi a havingaliambi a	isingaliambi a haingaliambi a	zisingaliambi a hazingaliambi a	usingaliambi a haungaliambi a	kusingaliambi a hakungaliambi a	pasingaliambi a hapangaliambi a	musingaliambi a hamungaliambi a			
Conditional Contrary to Fact																			[less ▲]		
Positive	ningeliambia	tungeliambia	ungeliambia	mngeliambia	angeliambia	wangeliambi a	ungeliambia	ingeliambia	lingeliambia	yangeliambi a	kingeliambia	vingeliambia	ingeliambia	zingeliambia	ungeliambia	kungeliambi a	pangeliambi a	mungeliambi a			
Gnomic																			[less ▲]		
Positive	naambia	twaambia	waambia	mwaambia	aambia	waambia	waambia	yaambia	laambia	yaambia	chaambia	vyaambia	yaambia	zaambia	waambia	kwaambia	paambia	mwaambia			
Perfect																			[less ▲]		

Vocabulary: word-level

- For the n-gram model, our vocabulary was comprised of all of the words in a language
- Some problems with this:
 - **can be quite large** - ~470,000 words Webster's English Dictionary (3rd edition)
 - **Language is changing all of the time** - 690 words were added to Merriam Webster's in September 2023 ("rizz", "goated", "mid")
 - **Long tail of infrequent words**. Many words just occur a few times
 - **Some words may not appear** in a training set of documents
 - **No modeled relationship between words** - e.g., "run", "ran", "runs", "runner" are all separate entries despite being linked in meaning

Character-level?

What about representing text with characters?

$$V = \{a, b, c, \dots, z\} \quad |V| = 26$$

- (Maybe add capital letters, punctuation, spaces, ...)
- Pros:
 - Small vocabulary size (for English)
 - Complete coverage (unseen words are represented by letters)
- Cons:
 - Encoding becomes very long - # chars instead of # words
 - Poor inductive bias for learning

~~Word Character~~ Subword tokenization!

How can we combine the high coverage of character-level representation with the efficiency of word-level representation?

Subword tokenization! (e.g., Byte-Pair Encoding)

- Start with character-level representations
- Build up representations from there

Original BPE Paper (Sennrich et al., 2016)

<https://arxiv.org/abs/1508.07909>

Subword Modeling

- Subword modeling encompasses a wide range of methods for reasoning about structure below the word level
 - Subwords may be words, parts of words, characters, bytes
- The dominant modern paradigm is to learn a vocabulary of parts of words (subword tokens)
- At training and testing time, each word is split into a sequence of known subwords
- Different algorithms:
 - Byte-Pair Encoding
 - WordPiece Modeling
 - Follow different strategies. Often contain prepending / appending special tokens (##, </w>)

Hello how are U tday?



hello how are u tday?



[hello, how, are, u, tday, ?]



[hello, how, are, u, td, ##ay, ?]

Byte-pair encoding

- Byte-pair encoding is a simple, effective strategy for defining a subword vocabulary
- Adapted for word segmentation from data compression technique (Gage, 1994)
 - Instead of merging frequent pairs of bytes, we merge characters or character sequences

Byte-pair encoding

- Algorithm:
 1. Start with a vocabulary containing only characters and an “end-of-word” symbol.
 2. Using a corpus of text, find the most common adjacent characters “a,b”; add “ab” as a subword.
 3. Replace instances of the character pair with the new subword; repeat until desired vocabulary size.

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es								

Frequency

d-e (3)	l-o (7)	t-</w> (8)
e-r (2)	n-e (5)	w-</w> (5)
e-s (8)	o-w (7)	w-e (7)
e-w (5)	r-</w> (2)	w-i (3)
i-d (3)	s-t (8)	

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>

Vocabulary

d	e	i	l	n	o	s	t	w
e s	e s t							

Frequency

d-es (3)	l-o (7)	w-</w> (5)
e-r (2)	n-e (5)	w-es (5)
e-w (5)	o-w (7)	w-e (2)
e s -t (8)	r-</w> (2)	w-i (3)
i-d (3)	t-</w> (8)	

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es	est	est</w>	lo	low	low</w>	ne	new	newest</w>

After 10 merges

Byte-pair encoding

- At test time, first split words into sequences of characters, then apply the learned operations to merge the characters into larger, known symbols
- Originally used in NLP for machine translation; now a similar method (WordPiece) is used in pretrained models.

Word structure and subword models

- Common words end up being a part of the subword vocabulary, while rarer words are split into (sometimes intuitive, sometimes not) components.
- In the worst case, words are split into as many subwords as they have characters.

	word		vocab mapping	embedding
Common words	hat	→	hat	
	learn	→	learn	
Variations	taaaaasty	→	taa## aaa## sty	  
misspellings	laern	→	la## ern##	 
novel items	Transformerify	→	Transformer## ify	 

Byte-pair encoding: ChatGPT example

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, drizzly 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 tozz 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

Tokens Characters
239 1109

[7368, 757, 57704, 1764, 301, 13, 4427, 1667, 4227, 2345, 37593, 4059, 1268, 1317, 24559, 2345, 69666, 2697, 477, 912, 3300, 304, 856, 53101, 11, 323, 4400, 4040, 311, 2802, 757, 389, 31284, 11, 358, 3463, 358, 1053, 30503, 922, 264, 2697, 323, 1518, 279, 30125, 727, 961, 315, 279, 1917, 13, 1102, 374, 264, 1648, 358, 617, 315, 10043, 1022, 279, 87450, 268, 323, 58499, 279, 35855, 13, 43633, 358, 1505, 7182, 7982, 44517, 922, 279, 11013, 26, 15716, 433, 374, 264, 41369, 11, 1377, 73825, 6841, 304, 856, 13836, 26, 15716, 358, 1505, 7182, 4457, 3935, 6751, 7251, 985, 1603, 78766, 83273, 11, 323, 12967, 709, 279, 14981, 315, 1475, 32079, 358, 3449, 26, 323, 5423, 15716, 856, 6409, 981, 636, 1778, 459, 8582, 1450, 315, 757, 11, 430, 433, 7612, 264, 3831, 16033, 17966, 311, 5471, 757, 505, 36192, 36567, 1139, 279, 8761, 11, 323, 1749, 2740, 50244, 1274, 753, 45526, 1022, 2345, 3473, 11, 358, 2759, 433, 1579, 892, 311, 10616, 636, 311, 9581, 439, 5246, 439, 358, 649, 13, 1115, 374, 856, 28779, 369, 40536, 323, 5041, 13, 3161, 264, 41903, 67784, 356, 4428, 3872, 5678, 5304, 813, 20827, 26, 358, 30666, 1935, 311, 279, 8448, 13, 2684, 374, 4400, 15206, 304, 420, 13, 1442, 814, 719, 7020, 433, 11, 4661, 682, 3026, 304, 872, 8547, 11, 1063, 892, 477, 1023, 11, 87785, 1022, 7151, 672, 1890, 16024, 7119, 279, 18435, 449, 757, 13]

TEXT TOKEN IDS

Byte-pair encoding: usage

- Basically state of the art in tokenization
- Used in all modern left-to-right large language models (LLMs), including ChatGPT

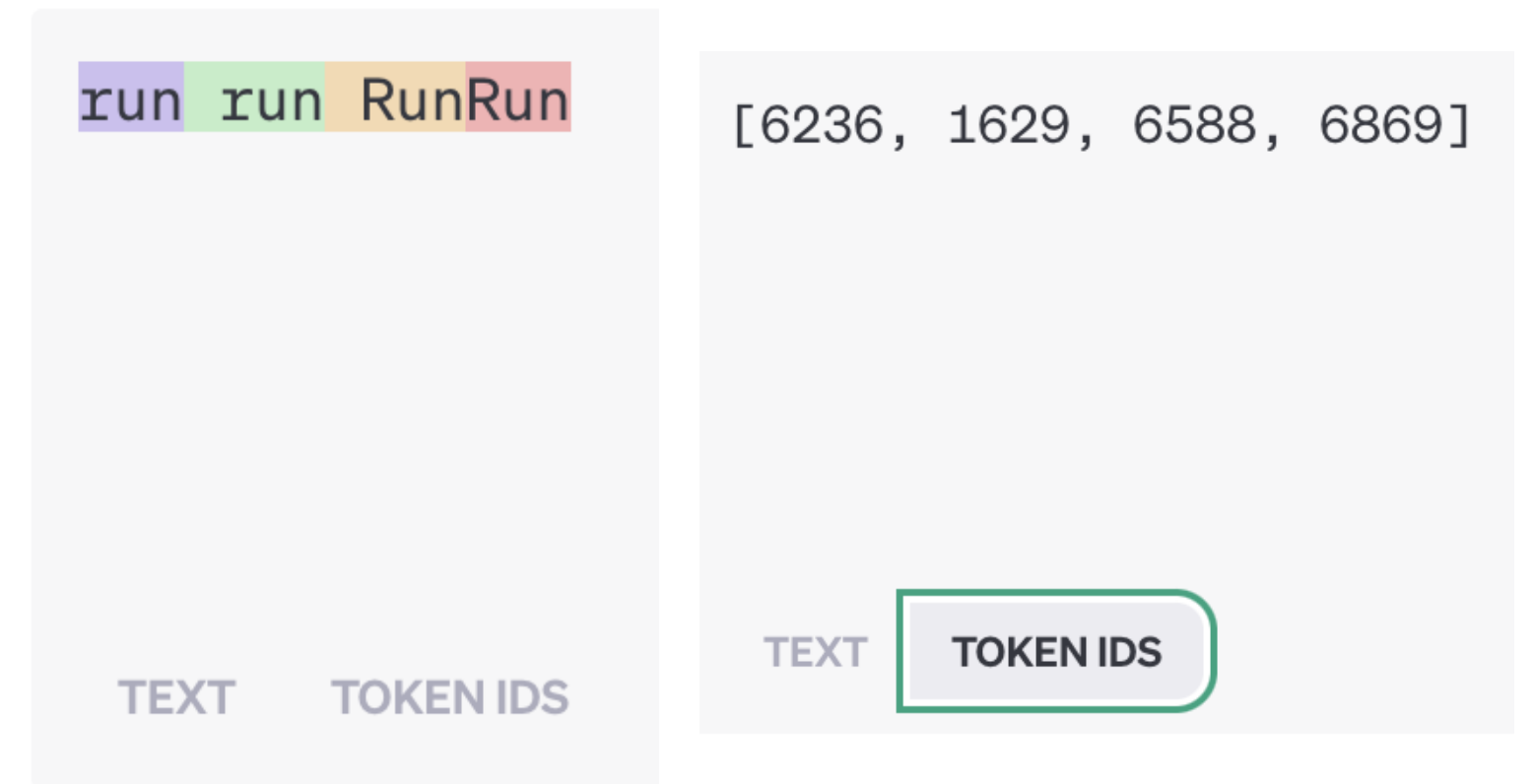
Model/Tokenizer	Vocabulary Size
GPT-3.5/GPT-4/ChatGPT	100k
GPT-2/GPT-3	50k
Llama2	32k
Falcon	65k

Byte-pair encoding: properties

- Efficient to run (greedy vs. global optimization)
- Lossless compression
- Potentially some shared representations - e.g., the token “hug” could be used both in “hug” and “hugging”

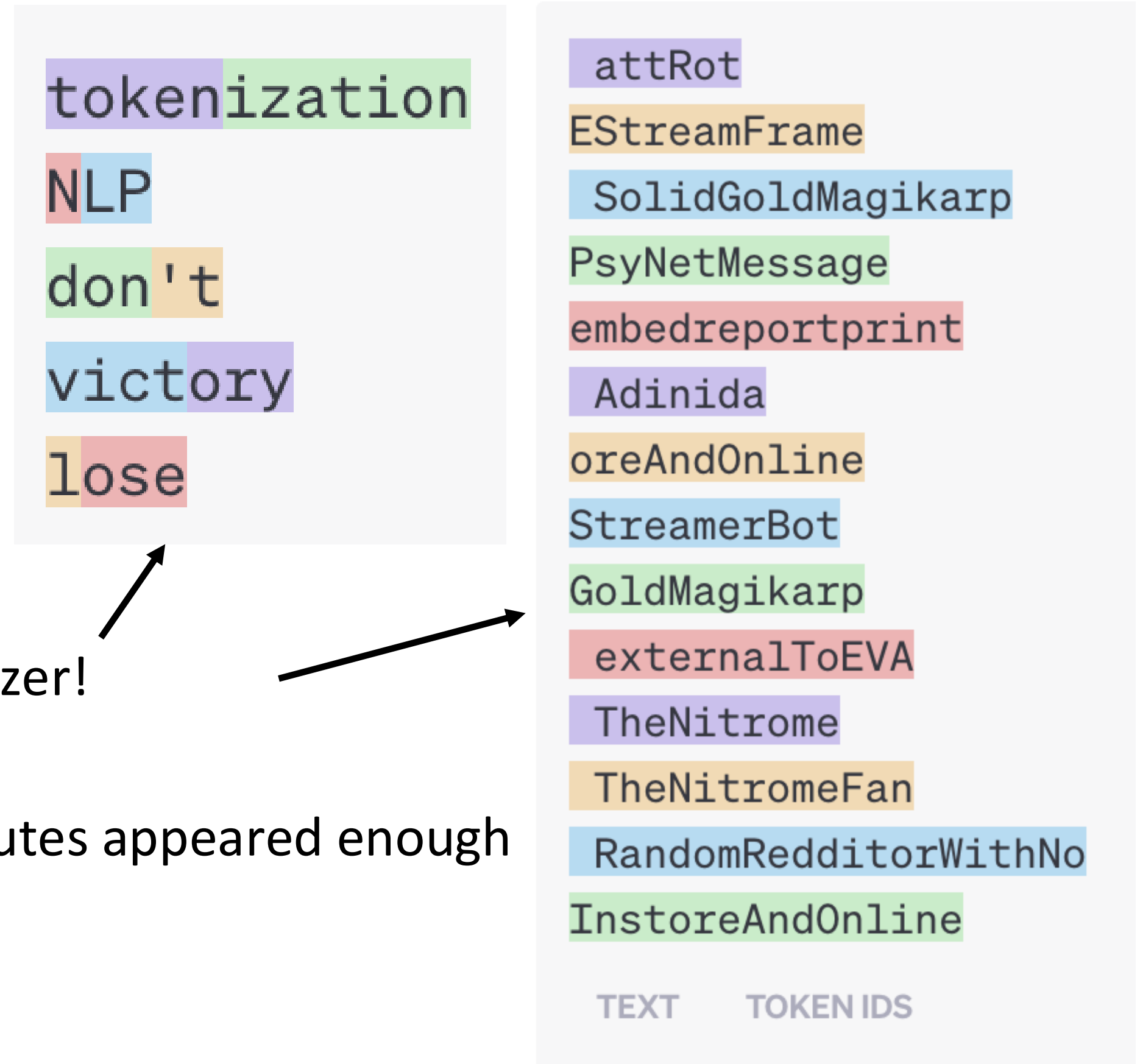
Weird properties of tokenizers

- Token != word
- Spaces are part of token
 - “run” is a different token than “ run”
- Not invariant to case changes
 - “Run” is a different token than “run”



Weird properties of tokenizers

- Token != word
- Spaces are part of token
 - “run” is a different token than “ run”
- Not invariant to case changes
 - “Run” is a different token than “run”
- Tokenization fits statistics of your data
 - e.g., while these words are multiple tokens...
 - These words are all 1 token in GPT-3’s tokenizer!
 - *Why?*
 - Reddit usernames and certain code attributes appeared enough in the corpus to surface as its own token!



Tokenization: Frequently Asked Questions

- Where does the token “##ing” come from?
 - In WordPiece tokenization, all non-starting characters are initialized as ##x.
 - Like: h, ##e, ##l, ##l, ##o.
 - Upon merging, only the first segment keeps its ##.
- How is tokenization done in Chinese?
 - Follows the same broad overall algorithm, but the initial split into characters involve language-specific rules
 - e.g. stroke-level tokenization [Si et al., 2023]

Source: <https://huggingface.co/learn/nlp-course/chapter6/6?fw=pt>

Natural Language Generation

LLM inference

= We have a trained model and we want to use it.

Question: What is the difference between **inference**, **generation**, and **decoding**?

Inference

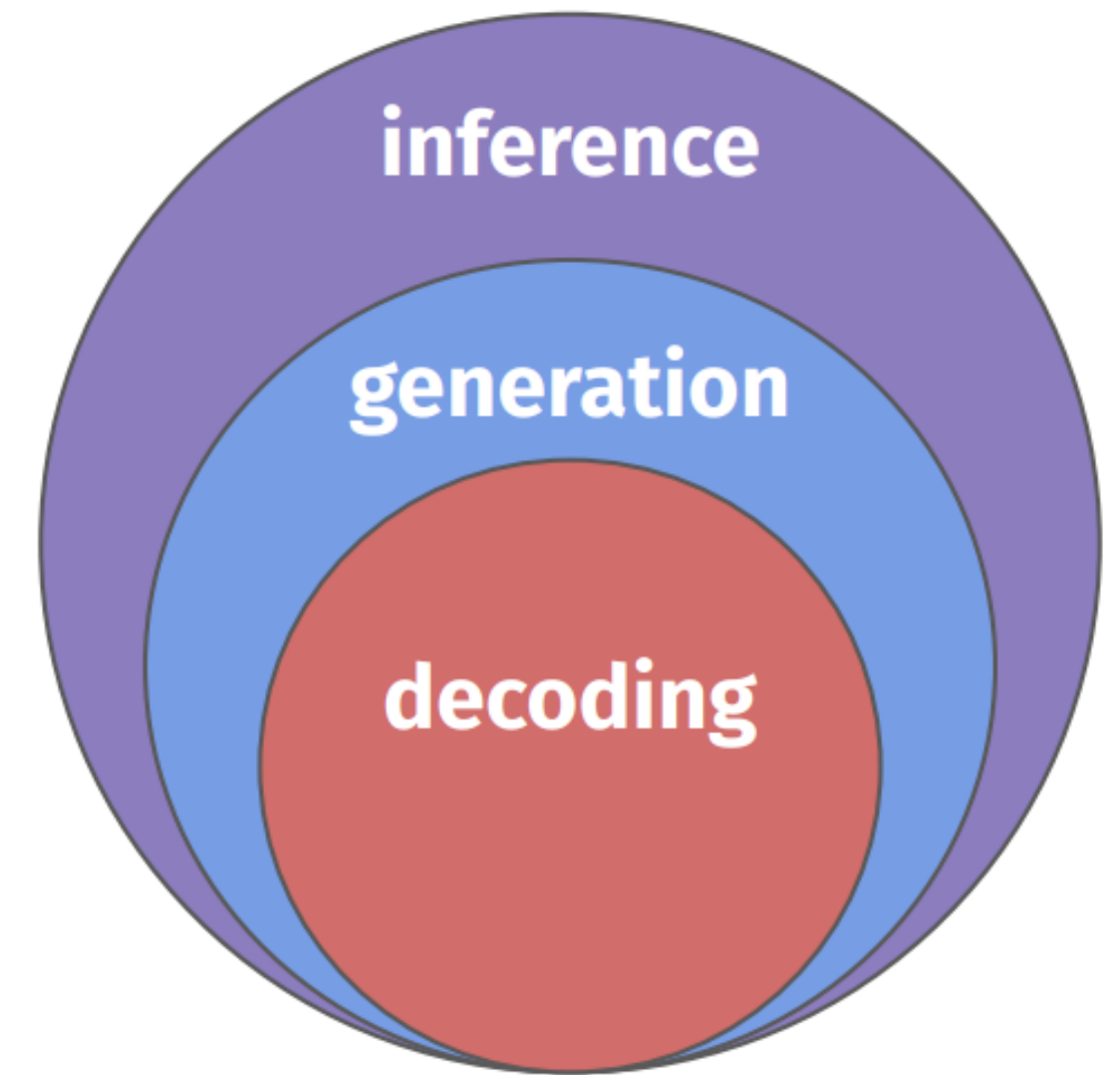
The concept of using a trained model for **making predictions** on new data (for classification, sequence tagging, text generation, ...).

Generation

The process of using a trained model for **producing a sequence of tokens**.

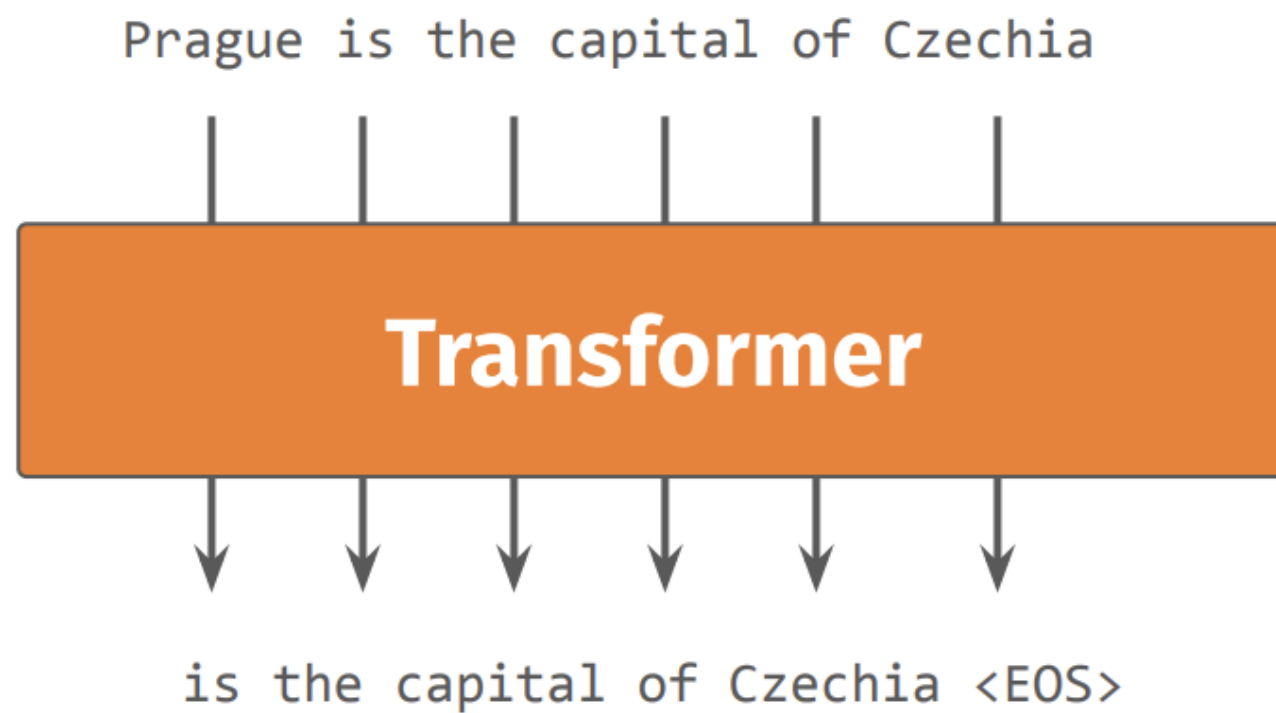
Decoding

The algorithm of **selecting the next token** using the model's internal representation.



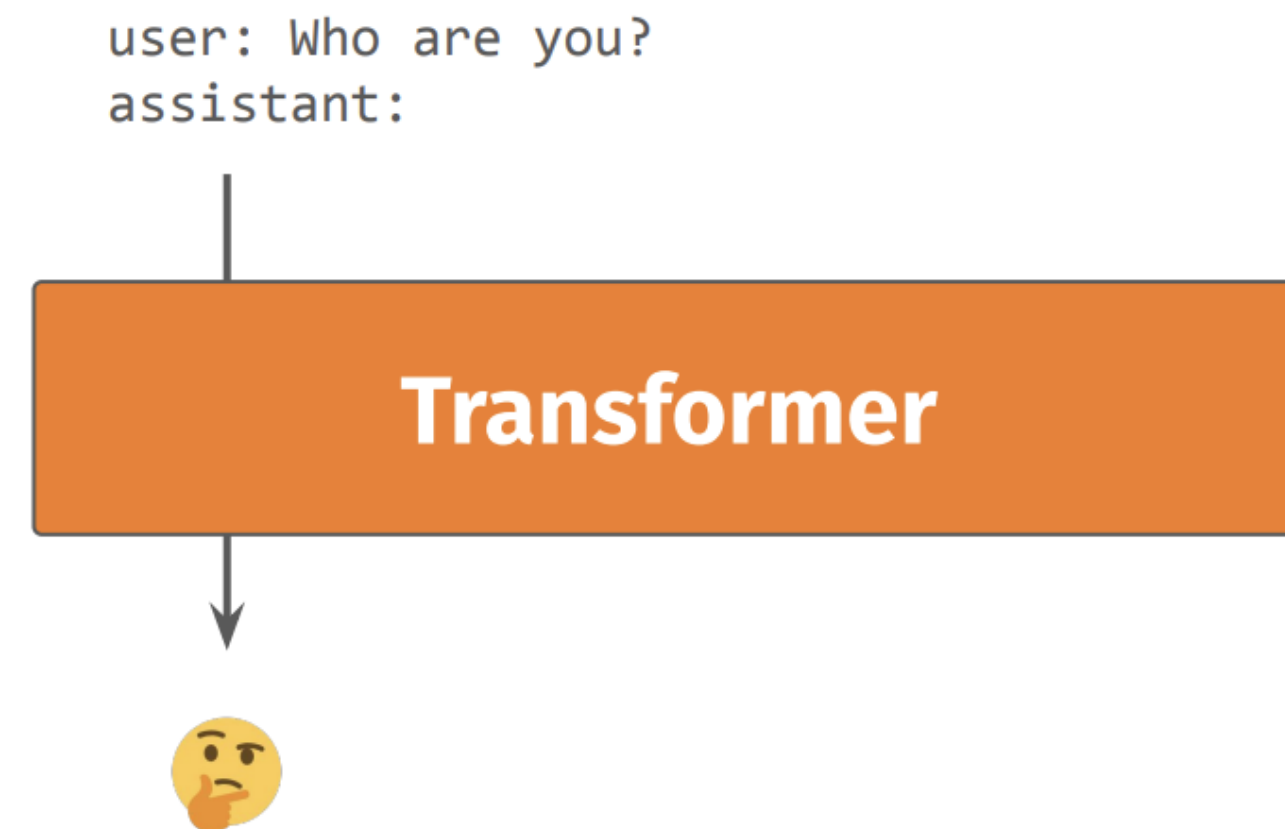
Training vs. Inference

Training



Teacher forcing: We know what token should come next, so we use it to train the model.

Inference



Decoding: We need to select what token should come next.

Natural Language Generation

- Natural language understanding and natural language generation are two sides of the same coin
 - In order to generate good language, you need to understand language
 - If you understand language, you should be able to generate it (with some effort)
- NLG is the workhorse of many classic and novel applications
 - AI Assistants
 - Translators
 - Search summarizers



NLG Use Cases

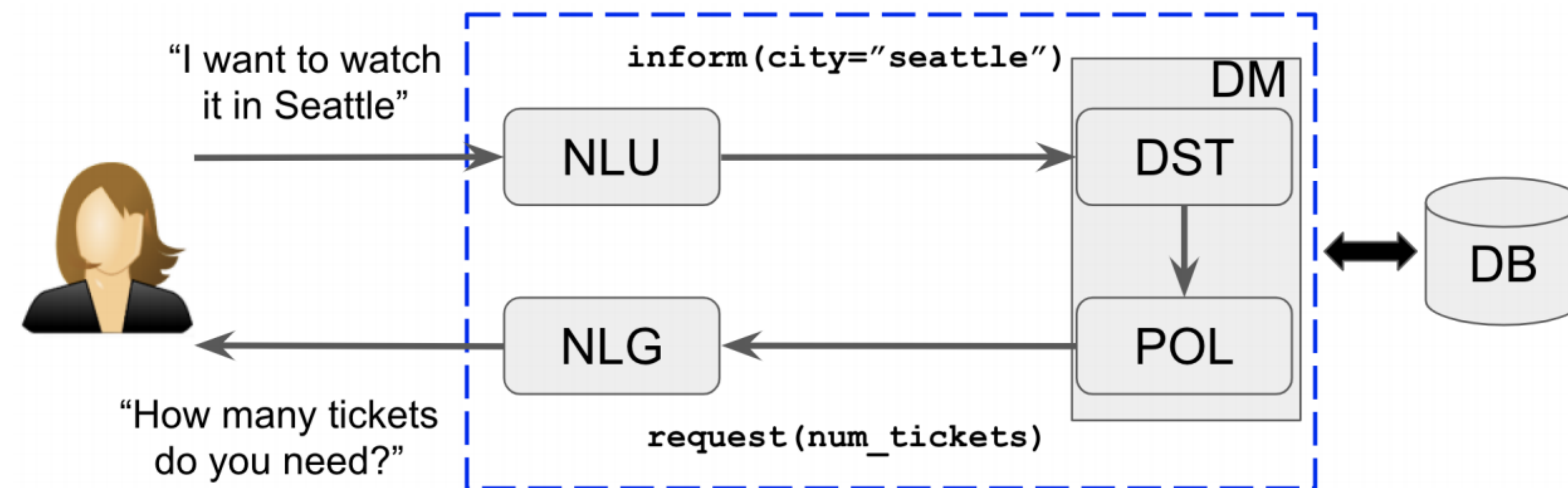
Simple and Effective Multi-Paragraph Reading Comprehension

Christopher Clark, Matt Gardner · Computer Science · ACL · 29 October 2017

TLDR We propose a state-of-the-art pipelined method for training neural paragraph-level question answering models on document QA data. [Expand](#)

👍 236 PDF · View PDF on arXiv Save Alert Cite Research Feed

Summarization



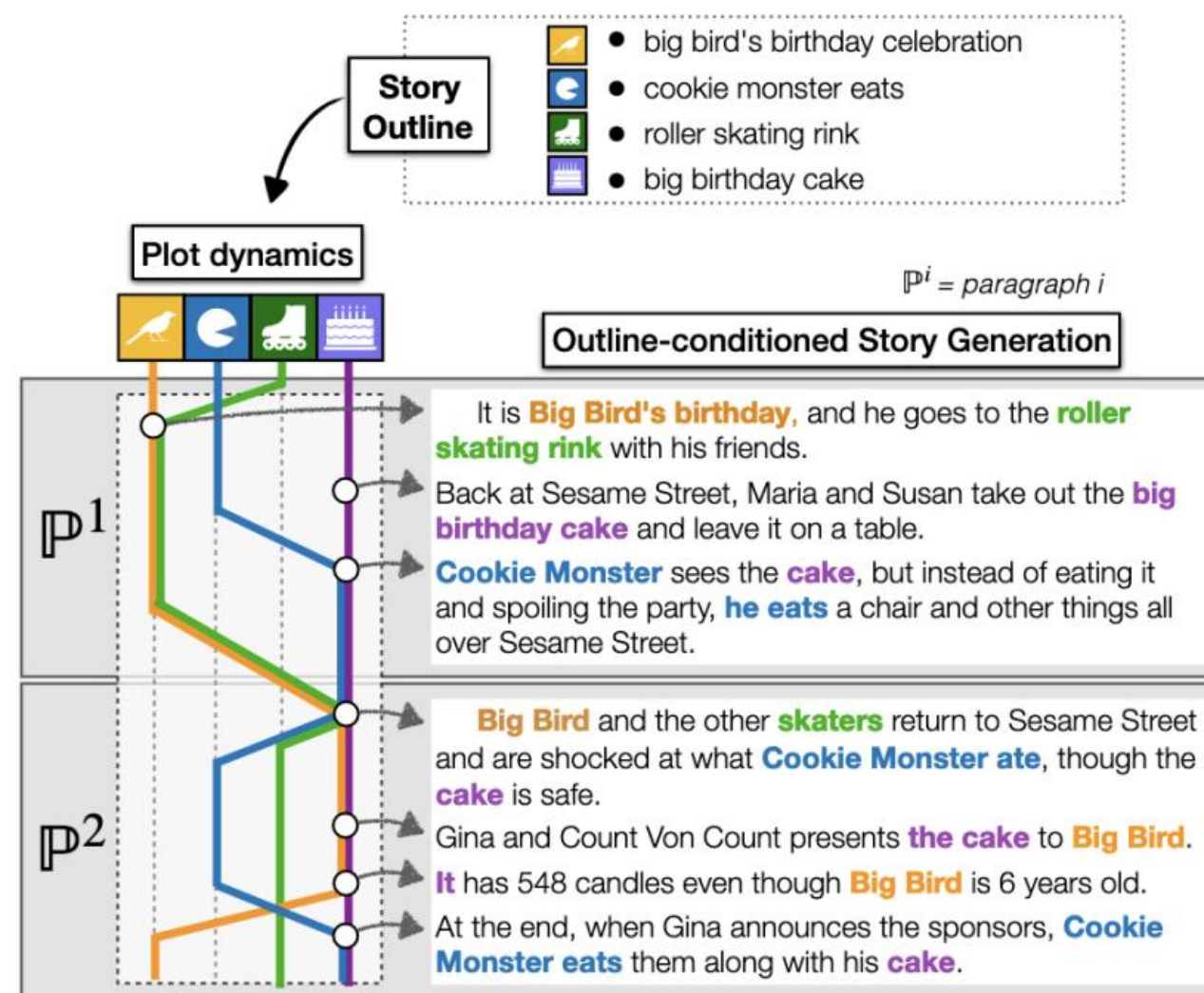
Task-driven
Dialog



Chitchat
Dialog

More Interesting NLG Uses

Creative stories



Rashkin et al., 2020

Data-to-text

Table Title: Robert Craig (American football)
Section Title: National Football League statistics
Table Description: None

YEAR	TEAM	ATT	RUSHING				NO.	RECEIVING			
			YDS	AVG	LNG	TD		YDS	AVG	LNG	TD
1983	SF	176	725	4.1	71	8	48	427	8.9	23	4
1984	SF	155	649	4.2	28	4	71	675	9.5	64	3
1985	SF	214	1050	4.9	62	9	92	1016	11	73	6
1986	SF	204	830	4.1	25	7	81	624	7.7	48	0
1987	SF	215	815	3.8	25	3	66	492	7.5	35	1
1988	SF	310	1502	4.8	46	9	76	534	7.0	22	1
1989	SF	271	1054	3.9	27	6	49	473	9.7	44	1
1990	SF	141	439	3.1	26	1	25	201	8.0	31	0
1991	RAI	162	590	3.6	15	1	17	136	8.0	20	0
1992	MIN	105	416	4.0	21	4	22	164	7.5	22	0
1993	MIN	38	119	3.1	11	1	19	169	8.9	31	1
Totals	-	1991	8189	4.1	71	56	566	4911	8.7	73	17

Craig finished his eleven NFL seasons with 8,189 rushing yards and 566 receptions for 4,911 receiving yards.

Parikh et al., 2020

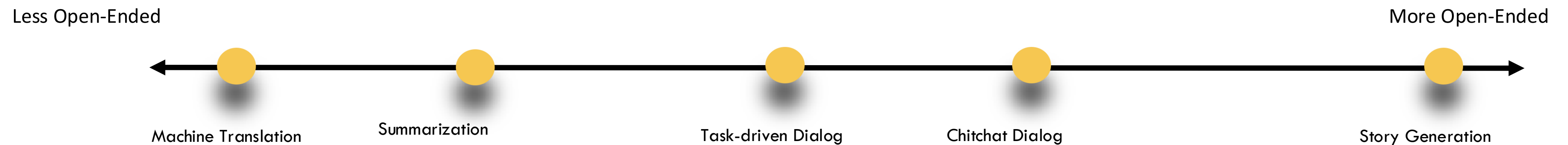
Visual description



Two children are sitting at a table in a restaurant. The children are one little girl and one little boy. The little girl is eating a pink frosted donut with white icing lines on top of it. The girl has blonde hair and is wearing a green jacket with a black long sleeve shirt underneath. The little boy is wearing a black zip up jacket and is holding his finger to his lip but is not eating. A metal napkin dispenser is in between them at the table. The wall next to them is white brick. Two adults are on the other side of the short white brick wall. The room has white circular lights on the ceiling and a large window in the front of the restaurant. It is daylight outside.

Krause et al., 2017

Broad Spectrum of NLG Tasks



Open-ended generation: the output distribution still has high freedom.

Non-open-ended generation: the input mostly determines the output generation.

Broad Spectrum of NLG Tasks

Less Open-Ended

More Open-Ended

