

Language Modeling

Pierre Colombo

`pierre.colombo@centralesupelec.fr`

MICS - CentraleSupélec

Advanced Natural Language Processing



CentraleSupélec

Lectures Outline

- 1. The Basics of Natural Language Processing**
- 2. Representing Text with Vectors**
- 3. Deep Learning Methods for NLP**
- 4. Classification for NLP / Revision of Transformers**
- 5. Generative AI for NLP / Revision of Transformers**
- 6. Introduction to RAG / Introduction to Distillation**
- 7. Badass Language Modeling: CroissantLLM / TowerLLM**

Labs Outline

1. Describe Statistically large scale corpora
2. Statistical Based and Word2vec Based Retriever
3. Task-Specific Modelling with Neural Networks
4. Task-Specific Modelling with Neural Networks (II)
5. Machine Translation
6. Paper Presentations (4-5-6-7)
7. Paper Presentations (8-9-10)

Today Lecture Outline

- Deep Learning Framework
- Language Modeling
- Encoder/Decoder & Decoder for Language Modeling
- Pretraining / Finetuning Paradigm
- Self-Attention Mechanism and the Transformer Architecture

Framework: Language Modeling

Given $(t_1, \dots, t_D) \in V^D$, our goal is to estimate: $P(t_{n+1} | t_1, \dots, t_n)$

We saw how to estimate that with n-gram models

To do better:

→ Use a Deep-Learning Model

Framework: Language Modeling

Given $(t_1, \dots, t_D) \in V^D$, our goal is to estimate :

$$p(t_{n+1} | t_1, \dots, t_n)$$

Framework

We want to find dnn_{θ}

$$\begin{aligned} dnn_{\theta} : \quad V^D &\rightarrow [0, 1]^V \\ (t_1, \dots, t_D) &\mapsto \hat{p} \end{aligned}$$

$$\text{s.t. } \hat{p} = (p_i)_{i \in [0, V-1]}, \quad \forall i \quad p_i \in [0, 1] \text{ and } \sum_i p_i = 1$$






Decoder only and Seq2Seq for Language Modeling

Word structure and subword models

Let's take a look at the assumptions we've made about a language's vocabulary.

We assume a fixed vocab of tens of thousands of words, built from the training set.

All *novel* words seen at test time are mapped to a single UNK.

	word		vocab mapping	embedding
Common words	hat	→	pizza (index)	
	learn	→	tasty (index)	
Variations	taaaaasty	→	UNK (index)	
misspellings	laern	→	UNK (index)	
novel items	Transformerify	→	UNK (index)	

The byte- pair encoding algorithm

Subword modeling in NLP encompasses a wide range of methods for reasoning about structure below the word level. (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.

Byte - pair encoding is a simple, effective strategy for defining a subword vocabulary.










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 vocab size.

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	 

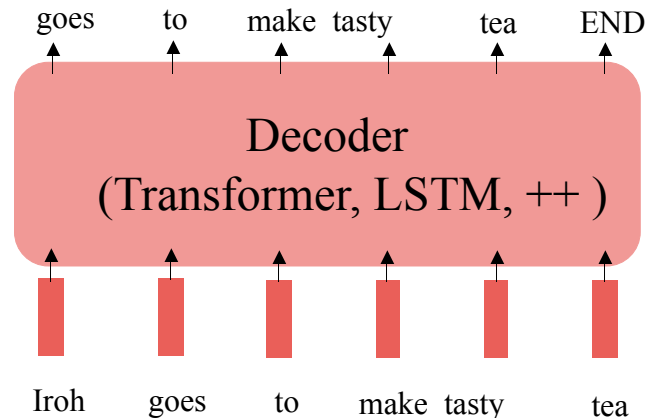
Pretraining through language modeling [Dai and Le, 2015]

Recall the **language modeling** task:

- Model $p(t_{n+1} | t_{n-1}, \dots, t_1)$, the probability distribution over words given their past contexts.
- There's lots of data for this! (In English.)

Pretraining through language modeling:

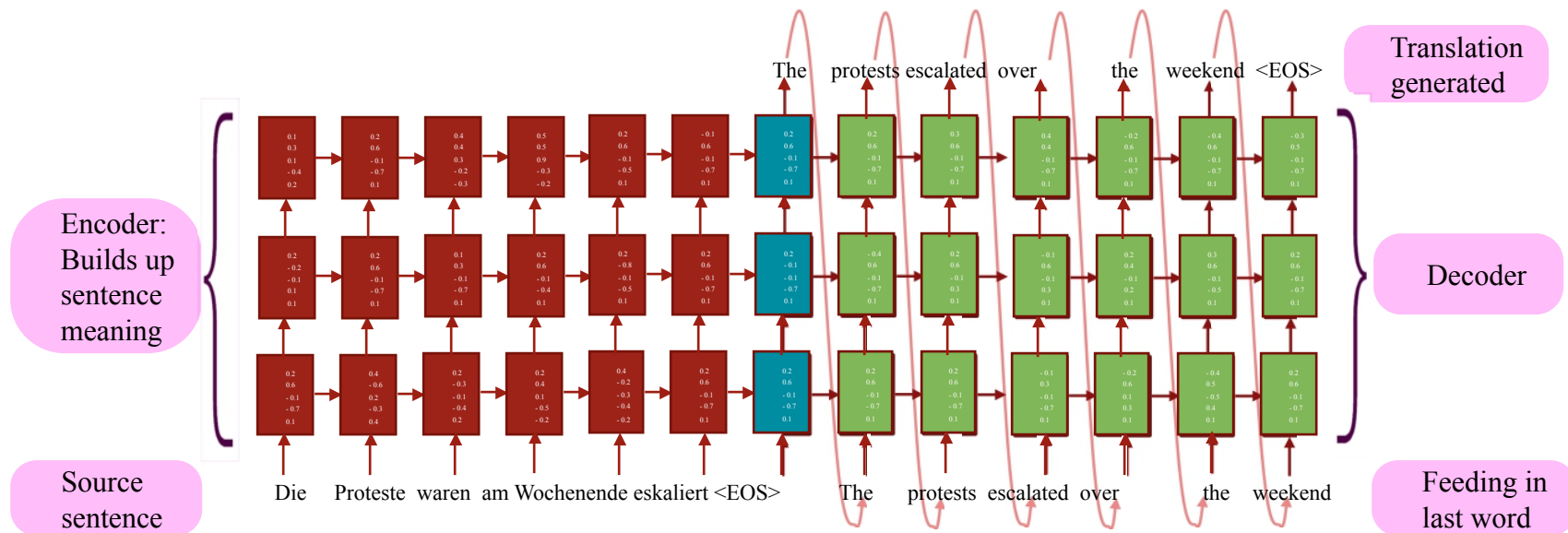
- Train a neural network to perform language modeling on a large amount of text.
- Save the network parameters.



Multi-layer deep encoder-decoder

[Sutskever et al. 2014; Luong et al. 2015]

The hidden states from RNN layer i are the inputs to RNN layer $i+1$



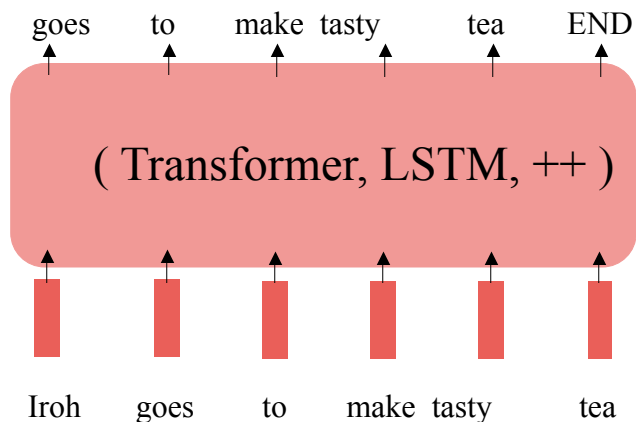
Pretraining Finetuning Paradigm

The Pretraining / Finetuning Paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

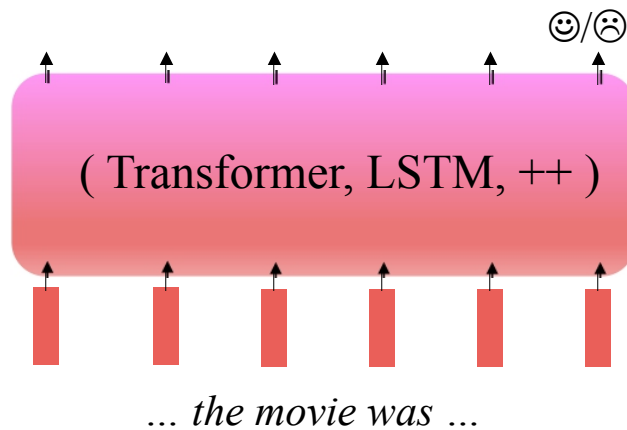
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



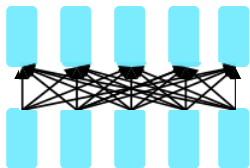
Step 2: Finetune (on your task)

Not many labels; adapt to the task!



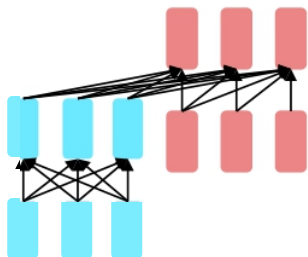
Pretraining for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



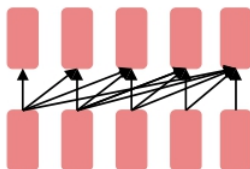
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



**Encoder-
Decoders**

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

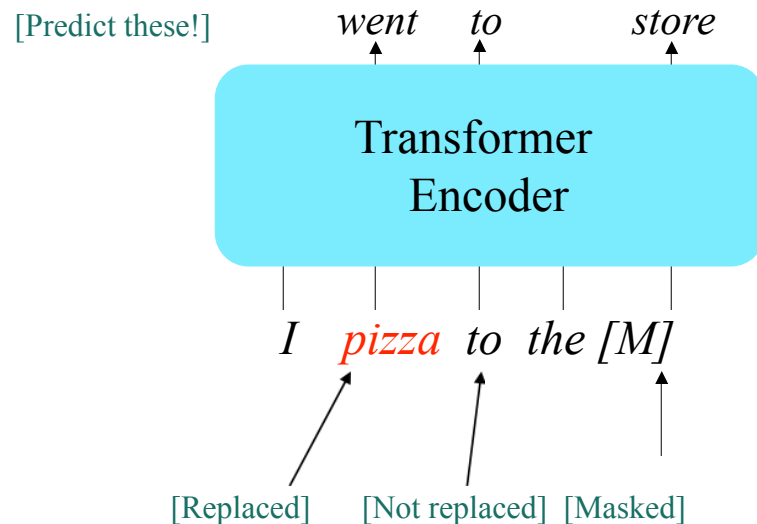
Pretraining Finetuning Paradigm: Case Study of Encoder

BERT: Bidirectional Encoder Representations from Transformers

Devlin et al., 2018 proposed the “Masked LM” objective and **released the weights of a pretrained Transformer**, a model they labeled BERT.

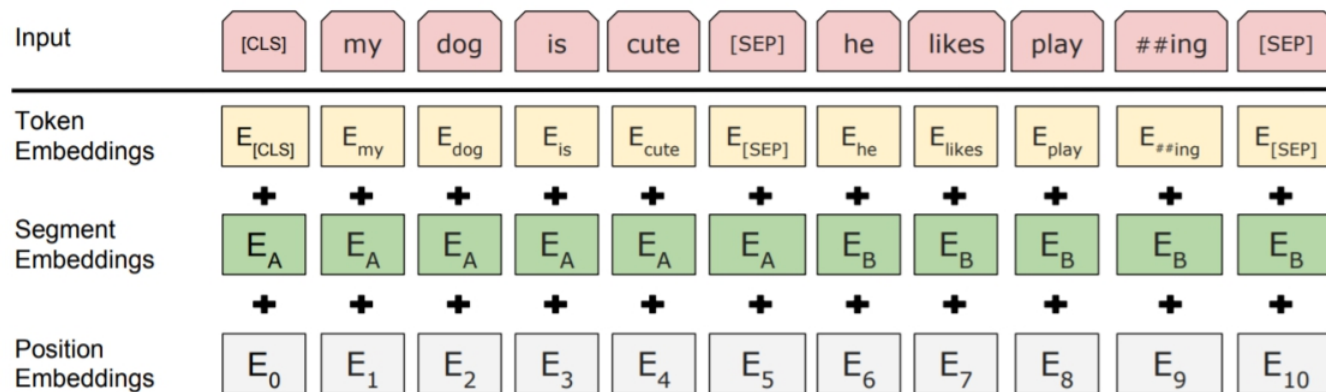
Some more details about Masked LM for BERT:

- Predict a random 15% of (sub)word tokens.
 - Replace input word with [MASK] 80% of the time
 - Replace input word with a random token 10% of the time
 - Leave input word unchanged 10% of the time (but still predict it!)
- Why? Doesn't let the model get complacent and not build strong representations of non - masked words. (No masks are seen at fine - tuning time!)



BERT: Bidirectional Encoder Representations from Transformers

- The pretraining input to BERT was two separate contiguous chunks of text:



- BERT was trained to predict whether one chunk follows the other or is randomly sampled.
 - Later work has argued this “next sentence prediction” is not necessary.

BERT: Bidirectional Encoder Representations from Transformers

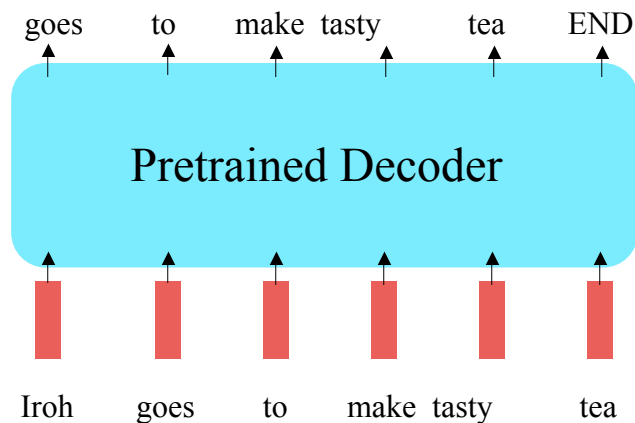
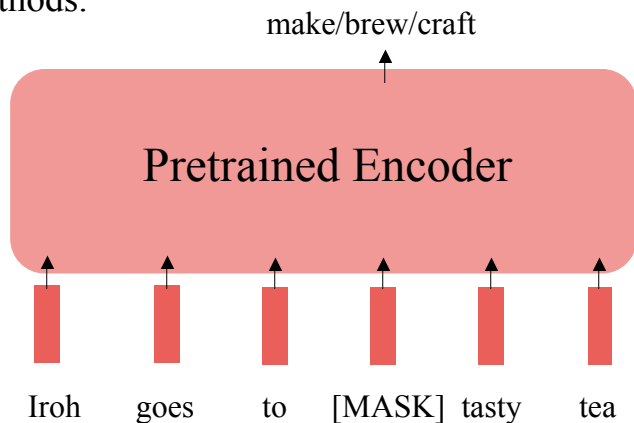
Details about BERT

- Two models were released:
 - BERT- base: 12 layers, 768 - dim hidden states, 12 attention heads, 110 million params.
 - BERT- large: 24 layers, 1024 - dim hidden states, 16 attention heads, 340 million params.
- Trained on:
 - BooksCorpus (800 million words)
 - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
 - BERT was pretrained with 64 TPU chips for a total of 4 days.
 - (TPUs are special tensor operation acceleration hardware)
- Finetuning is practical and common on a single GPU
 - “Pretrain once, finetune many times.”

Limitations of pretrained encoders

Those results looked great! Why not use pretrained encoders for everything?

If your task involves generating sequences, consider using a pretrained decoder; BERT and other pretrained encoders don't naturally lead to nice autoregressive (1 - word- at - a-time) generation methods.

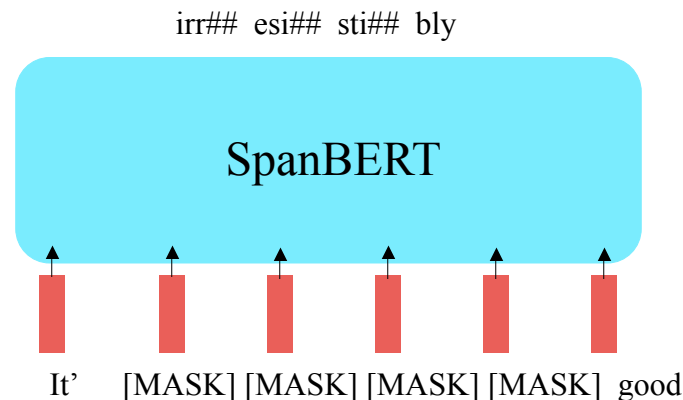
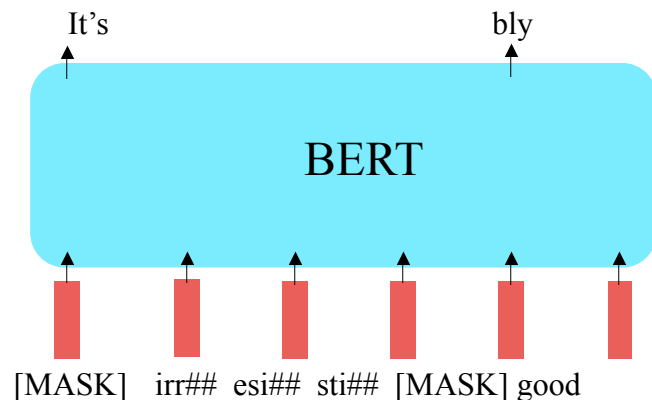


Extensions of BERT

You'll see a lot of BERT variants like RoBERTa , SpanBERT , +++

Some generally accepted improvements to the BERT pretraining formula:

- RoBERTa : mainly just train BERT for longer and remove next sentence prediction!
- SpanBERT: masking contiguous spans of words makes a harder, more useful pretraining task



Extensions of BERT

A takeaway from the RoBERTa paper: more compute, more data can improve pretraining even when not changing the underlying Transformer encoder.

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

Pretraining Finetuning Paradigm: Case Study of Encoder-Decoder

Pretraining encoder- decoders: what pretraining objective to use?

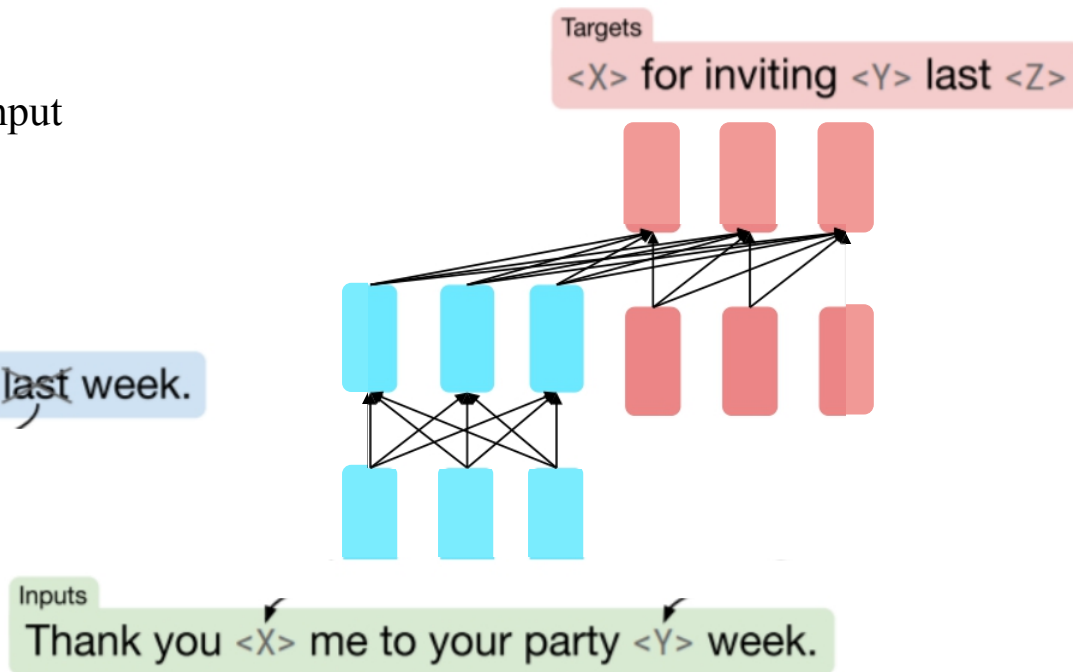
What [Raffel et al., 2018](#) found to work best was **span corruption**. Their model: **T5**.

Replace different - length spans from the input with unique placeholders; decode out the spans that were removed!

Original text

Thank you for inviting me to your party last week.

This is implemented in text preprocessing: it's still an objective that looks like **language modeling** at the decoder side.



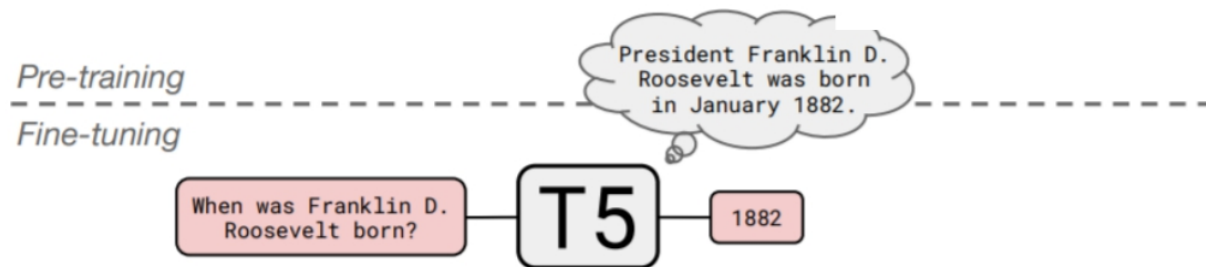
Pretraining encoder- decoders: what pretraining objective to use?

[Raffel et al., 2018](#) found encoder - decoders to work better than decoders for their tasks, and span corruption (denoising) to work better than language modeling.

Architecture	Objective	Params	Cost	GLUE	CNNNDM	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

Pretraining encoder- decoders: what pretraining objective to use?

A fascinating property of T5: it can be finetuned to answer a wide range of questions, retrieving knowledge from its parameters.



NQ: Natural Questions

WQ: WebQuestions

TQA: Trivia QA

All “open - domain”
versions

	NQ	WQ	TQA		
			dev	test	
<u>Karpukhin et al. (2020)</u>	41.5	42.4	57.9	–	
T5.1.1-Base	25.7	28.2	24.2	30.6	220 million params
T5.1.1-Large	27.3	29.5	28.5	37.2	770 million params
T5.1.1-XL	29.5	32.4	36.0	45.1	3 billion params
T5.1.1-XXL	32.8	35.6	42.9	52.5	11 billion params
<u>T5.1.1-XXL + SSM</u>	35.2	42.8	51.9	61.6	

Pretraining Finetuning Paradigm: Case Study of Decoder

Generative Pretrained Transformer (GPT) [Radford et al., 2018]

2018's GPT was a big success in pretraining a decoder!

- Transformer decoder with 12 layers, 117M parameters.
- 768- dimensional hidden states, 3072 - dimensional feed - forward hidden layers.
- Byte - pair encoding with 40,000 merges
- Trained on BooksCorpus: over 7000 unique books.
 - Contains long spans of contiguous text, for learning long- distance dependencies.
- The acronym “GPT” never showed up in the original paper; it could stand for “Generative PreTraining” or “Generative Pretrained Transformer”

Generative Pretrained Transformer (GPT) [Radford et al., 2018]

How do we format inputs to our decoder for **finetuning tasks**?

Natural Language Inference: Label pairs of sentences as *entailing/contradictory/neutral*

Premise: *The man is in the doorway*
Hypothesis: *The person is near the door* } **entailment**

Radford et al., 2018 evaluate on natural language inference.

Here's roughly how the input was formatted, as a sequence of tokens for the decoder.

[START] *The man is in the doorway* [DELIM] *The person is near the door* [EXTRACT]

The linear classifier is applied to the representation of the [EXTRACT] token.

Increasingly convincing generations (GPT2) [Radford et al., 2018]

We mentioned how pretrained decoders can be used **in their capacities as language models**.

GPT- 2, a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

GPT- 3, In- context learning, and very large models

So far, we've interacted with pretrained models in two ways:

- Sample from the distributions they define (maybe providing a prompt)
- Fine -tune them on a task we care about, and take their predictions.

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

GPT-3 is the canonical example of this. The largest T5 model had 11 billion parameters.

GPT-3 has 175 billion parameters.

GPT- 3, In- context learning, and very large models

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

The in - context examples seem to specify the task to be performed, and the conditional distribution mocks performing the task to a certain extent.

Input (prefix within a single Transformer decoder context):

“ thanks -> merci
hello -> bonjour
mint -> menthe
otter -> ”

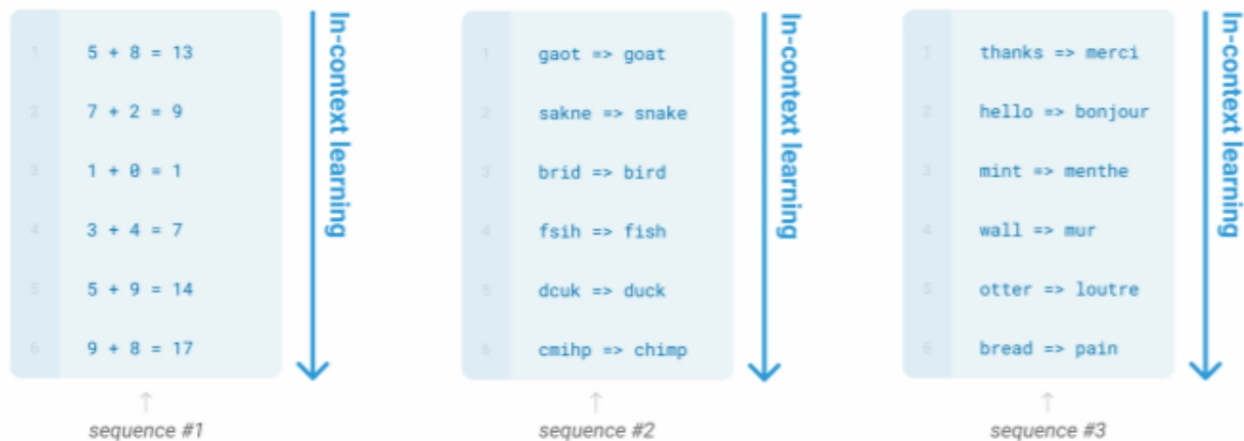
Output (conditional generations):

loutre ...”

GPT- 3, In- context learning, and very large models

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

Learning via SGD during unsupervised pre-training




Scaling Efficiency: how do we best use our compute

GPT-3 was **175B parameters** and trained on **300B** tokens of text.

Roughly, the cost of training a large transformer scales as **parameters*tokens**

Did OpenAI strike the right parameter-token data to get the best model? No.

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
<i>Chinchilla</i>	70 Billion	1.4 Trillion

—  This 70B parameter model is better than the much larger other models!

The prefix as task specification and scratch pad: chain - of- thought

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

What kinds of things does pretraining teach?

There's increasing evidence that pretrained models learn a wide variety of things about the statistical properties of language. Taking our examples from the start of class:

- *Stanford University is located in _____, California.* [Trivia]
- *I put ____ fork down on the table.* [syntax]
- *The woman walked across the street, checking for traffic over ____ shoulder.* [coreference]
- *I went to the ocean to see the fish, turtles, seals, and _____.* [lexical semantics/topic]
- *Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was ____.* [sentiment]
- *Iroh went into the kitchen to make some tea. Standing next to Iroh , Zuko pondered his destiny. Zuko left the _____.* [some reasoning – this is harder]
- *I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____* [some basic arithmetic; they don't learn the Fibonacci sequence]
- Models also learn – and can exacerbate racism, sexism, all manner of bad biases.
- More on all this in the interpretability lecture!