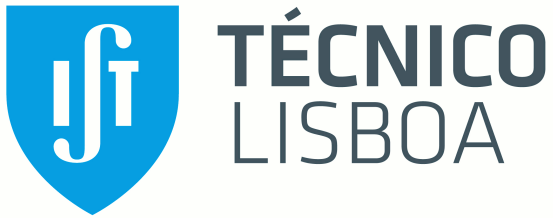
Lecture 11: Self-Supervised Learning and Large Pretrained Models

André Martins, Francisco Melo, Mário Figueiredo



Deep Learning Course, Winter 2022-2023

Announcements

- Next Thursday (January 12): guest lecture by Chrysoula Zerva.

  Grande Auditório, Centro de Congressos, 13:30

Today's Roadmap

Previous lecture: sequence-to-sequence models and transformers.

Today: large pretrained models (BERT, GPT3, etc.) and how to use them for downstream tasks.

- Contextualized representations
- Self-supervised learning
- Pretraining and finetuning
- Adaptors and prompting.

Pointers for Today's Class

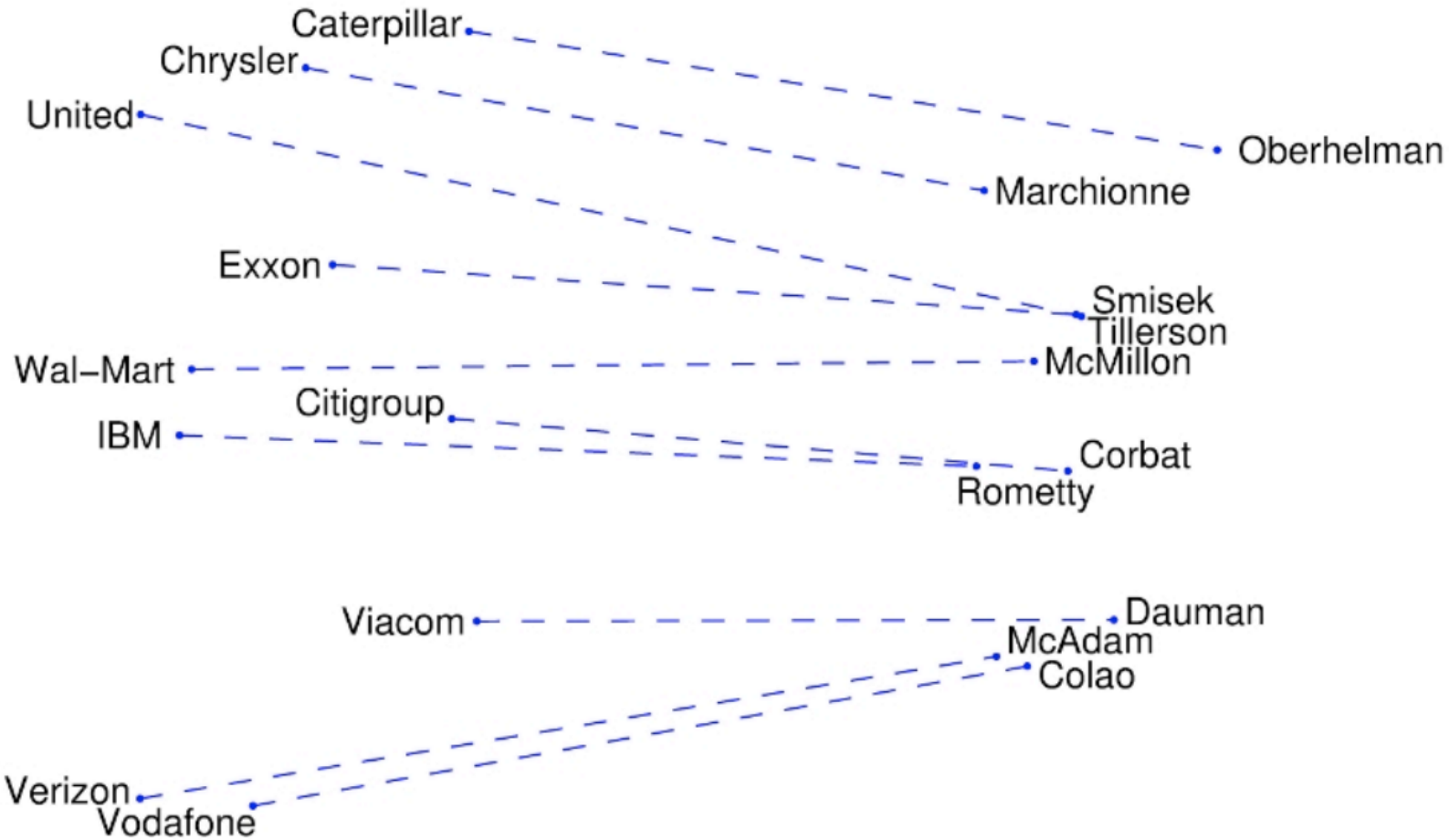
- John Hewitt's lecture on pretrained transformer models: `http://web.stanford.edu/class/cs224n/slides/cs224n-2021-lecture10-pretraining.pdf`

Outline

From Static to Contextualized Word Embeddings

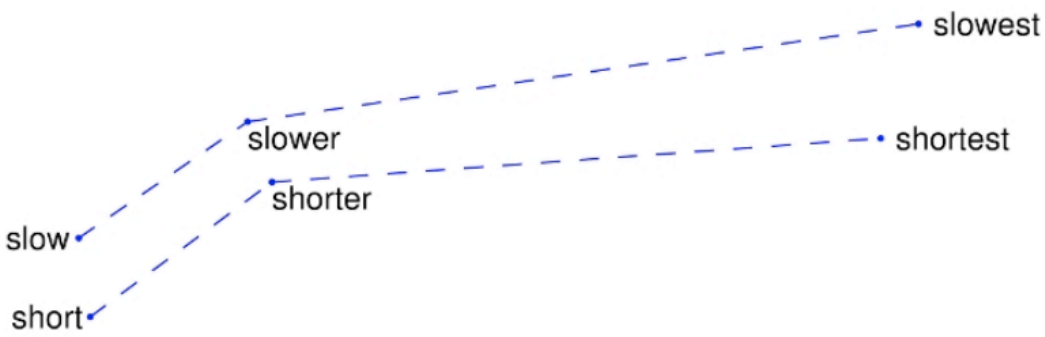
- In the representation learning lecture, we saw how to obtain word representations (embeddings) (e.g. word2vec, GloVe)
- Each word in the vocabulary is represented by a vector, regardless of its context (a "static" vector)
- Today: how to obtain contextualized embeddings.

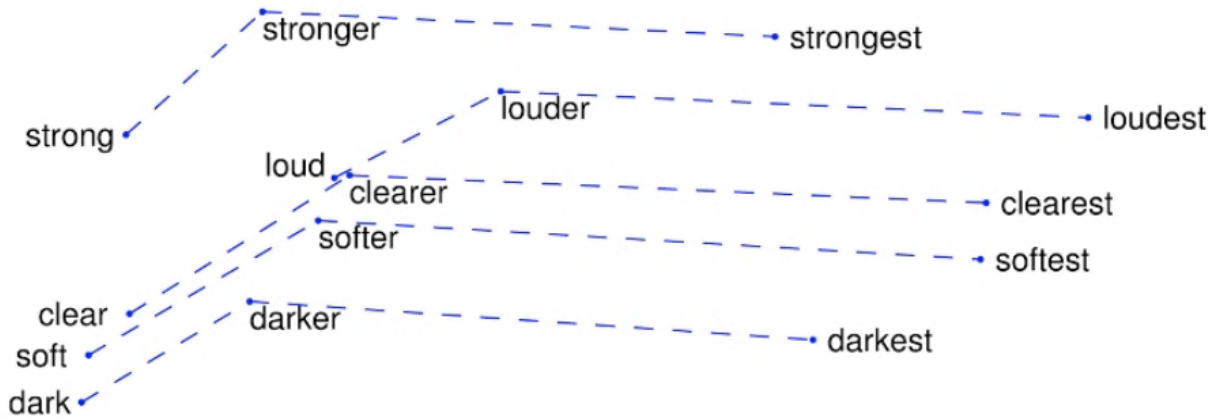
GloVe Visualizations: Company → CEO



(Slide credit to Richard Socher)

GloVe Visualizations: Superlatives





(Slide credit to Richard Socher)

Word Embeddings: Some Open Problems

- Can we do word embeddings for multiple languages in the same space?
- How to capture polysemy (e.g. "bear" vs "bear"; "flies" vs "flies")?
- Can we compute embeddings on-the-fly, depending on the context?

Contextualized Embeddings

- Words can have different meanings, depending on which context they appear in.
- In 2018, a model called ELMo learned context-dependent embeddings and achieved impressive results on 6 NLP downstream tasks (Peters et al., 2018).
- This was the first of a series of models named after Sesame Street characters (more to come).

Embeddings from Language Models (ELMo) (Peters et al., 2018)

Key idea:

- Pre-train a BiLSTM language model on a large dataset
- Save all the parameters at all layers, not only the embeddings
- Then, for your downstream task, tune a scalar parameter for each layer, and pass the entire sentence through this encoder.

Later several models have been proposed (BERT, GPT) with even more impressive performance.

Outline

Pretraining through Language Modeling

Recall the language modeling task:

- Model $p_\theta(y_t \mid y_{1:(t-1)})$, the probability distribution of words given their past contexts.
- There's lots of data for this! No *labels* are necessary, just raw text.
- This is called unsupervised pretraining or self-supervised learning.

Pretraining through language modeling:

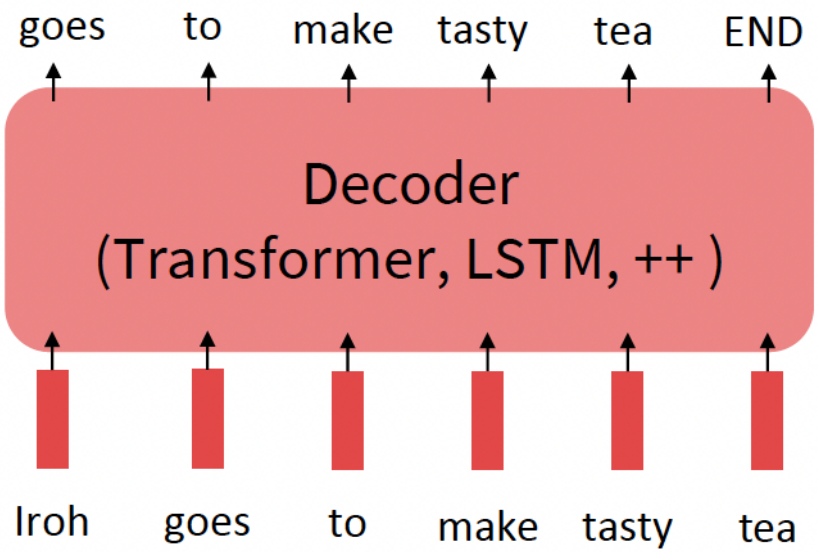
- Train a NN to perform language modeling on a large amount of text.
- Save the network parameters.

Pretraining and Fine-tuning

Pretraining can be very effective by serving as parameter initialization.

Step 1: Pretrain (e.g. on LM)

Lots of text; learn general things!

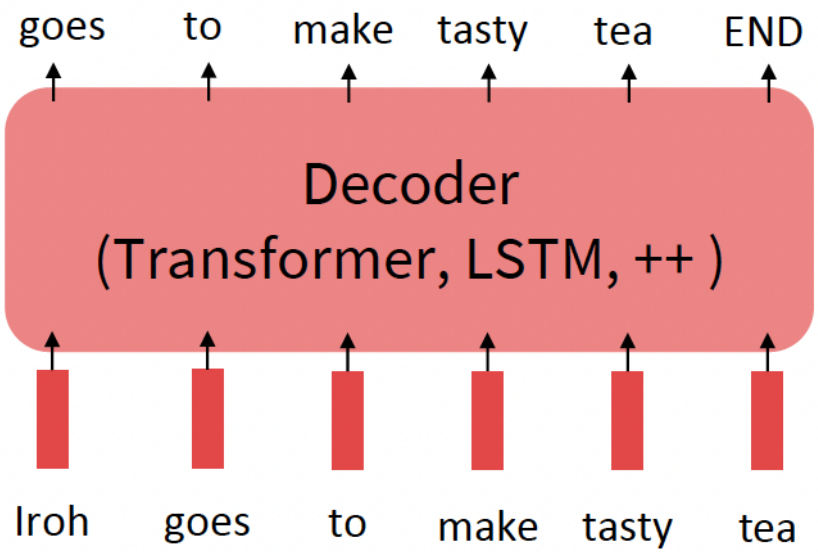


Pretraining and Fine-tuning

Pretraining can be very effective by serving as parameter initialization.

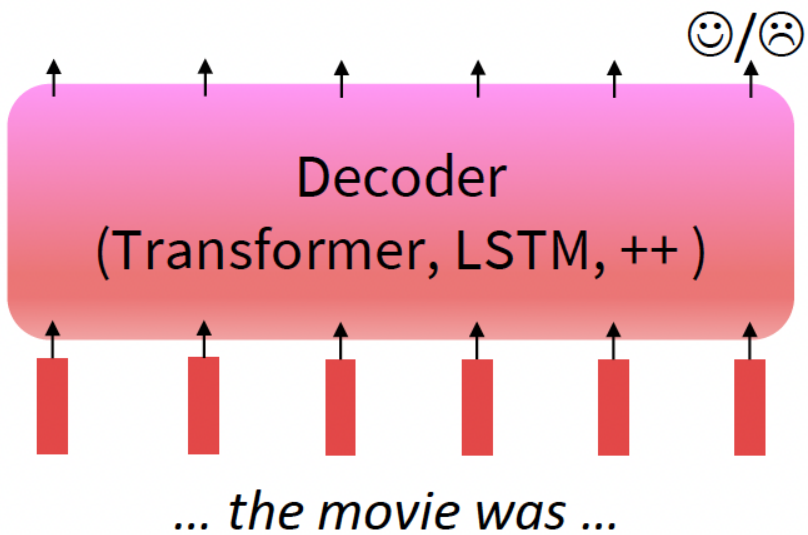
Step 1: Pretrain (e.g. on LM)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!

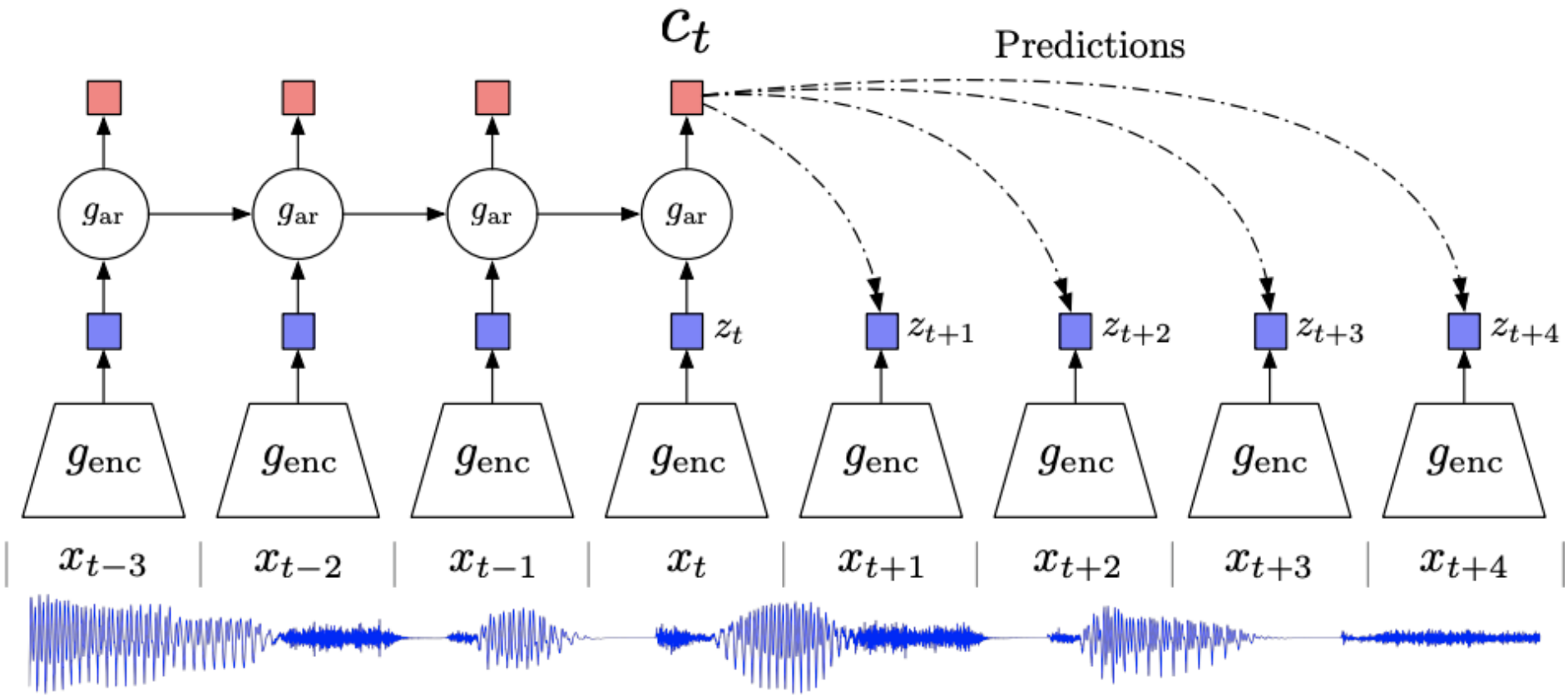


Self-Supervised Learning

Pretraining on language model task is a form of self-supervised learning

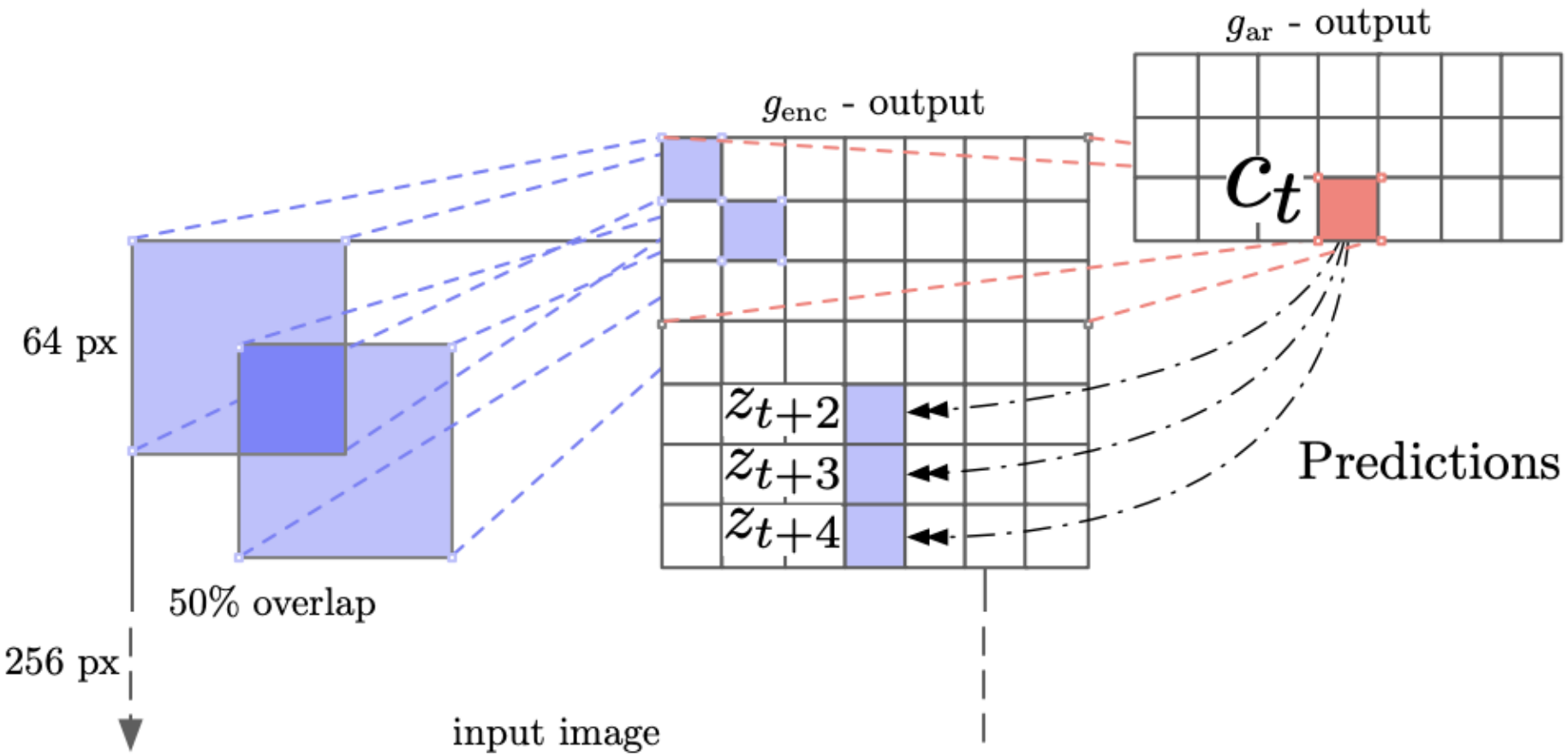
- Take raw (unlabeled) data, remove information and train a model to recover that information
- In the case of language modeling, the information removed is the next word; the model is trained to predict future words given the context
- Other strategies: *mask* words (later)
- This can be done with signals, images too, not just NLP
- For example, take images, obfuscate a region, and train a model to predict the missing region (image completion)

Constrastive Predictive Coding (Speech)



(From Oord et al. (2018))

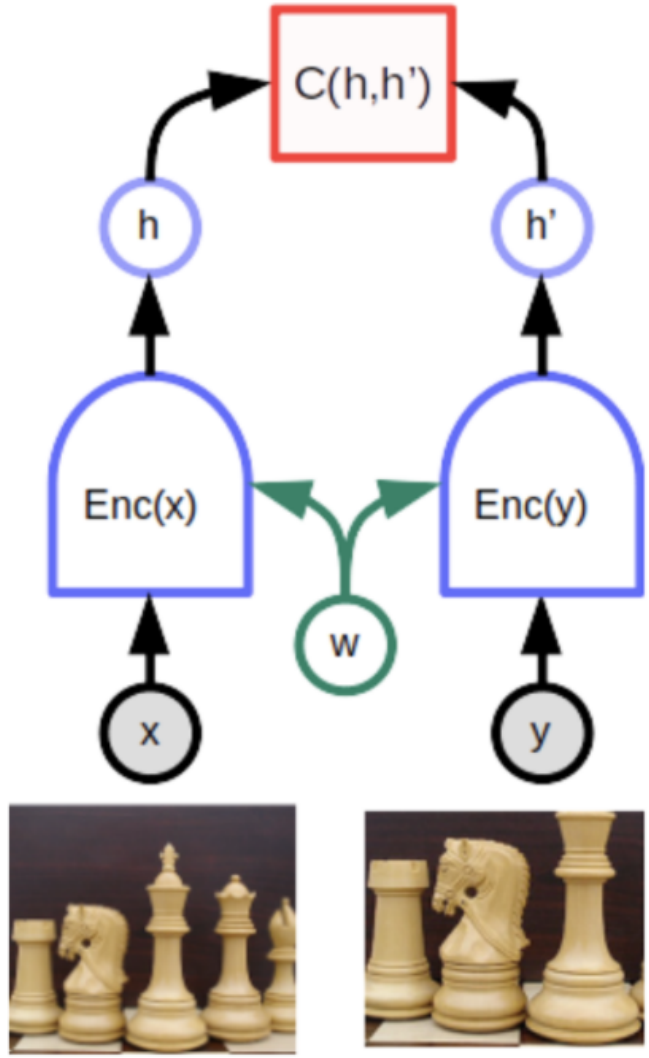
Constrastive Predicting Coding (Images)



(From Oord et al. (2018))

Siamese Networks

- Rotate, translate, or scale existing image.
- Minimize the distance between the two representations.



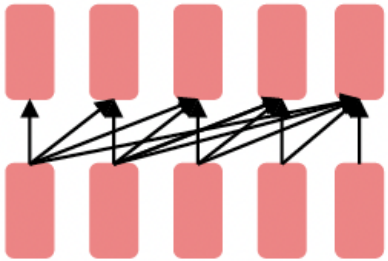
(From `https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/`)

Why Does This Work?

Let's look at pretraining and fine-tuning from a "training neural nets" perspective.

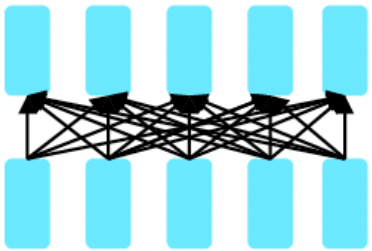
- Pretraining leads to parameters $\hat{\theta} \approx \arg\min_\theta L_{\text{pretrain}}(\theta)$
- Fine-tuning approximates $\arg\min_\theta L_{\text{finetune}}(\theta)$, starting at $\hat{\theta}$
- Pretraining helps because SGD stays (relatively) close to $\hat{\theta}$ during fine-tuning.
- Pretraining on large datasets exposes the model to many words and contexts not seen in the fine-tuning data (a form of weak supervision).
- Hopefully, the fine-tuning local minima near $\hat{\theta}$ tend to generalize well!

Three Architectures for Pretraining



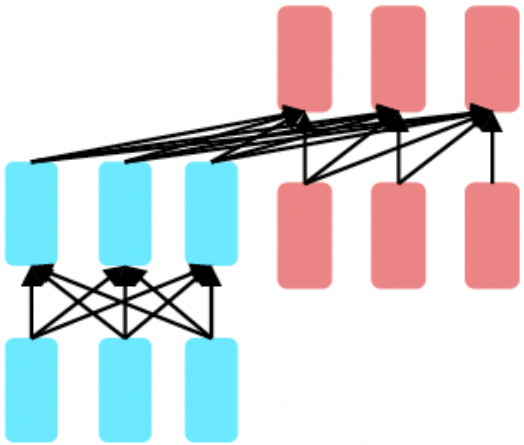
Decoders

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



Encoders

- Bidirectional context $\Rightarrow$ can condition on future!
- Wait, how do we pretrain them?



Encoder-Decoders

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

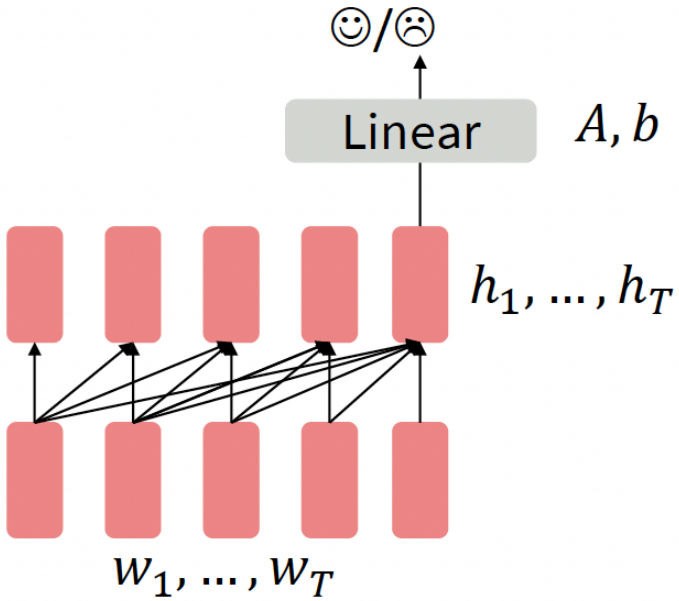
Pretrained Decoders

- When using language model pretrained decoders, we can ignore that they were trained to model $p_\theta(x_t \mid x_{1:(t-1)})$

- Fine-tuning by training a classifier on the last hidden state.

$$\boldsymbol{h}_1, \ldots, \boldsymbol{h}_L = \text{Decoder}(x_1, \ldots, x_L)$$
$$\boldsymbol{y} = \text{softmax}(\boldsymbol{A}\boldsymbol{h}_L + \boldsymbol{b})$$

where $\boldsymbol{A}$ and $\boldsymbol{b}$ are randomly initialized and learned by the downstream task.



Pretrained Decoders

Two common choices for fine-tuning:

- *Freeze* the pretrained model and train only $\boldsymbol{A}$ and $\boldsymbol{b}$
- Or fine-tune everything, letting gradients backpropagate through the whole network.

Pretrained decoders are particularly useful for generation tasks:

- Summarization
- Machine Translation
- Dialogue
- etc.

The family of GPT models developed by OpenAI is an example.

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

- 2018's GPT was a big success in pretraining a decoder!
- Transformer decoder with 12 layers.
- 768-D hidden states, 3072-D feed forward hidden layers.
- Byte pair encoding with 40,000 merges (vocabulary size)
- Trained on BooksCorpus over 7000 books.
- Contains long spans of contiguous text, for learning long distance dependencies.

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

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

Radford et al. (2018) evaluates on **Natural Language Inference** (NLI):

- Label pairs of sentences as entailment/contradiction/neutral

  Premise: "The man is in the doorway"
  Hypothesis: "The person is near the door" $\Rightarrow$ *entailment*

The input is formatted to become a sequence of tokens for the decoder:
[START] The man is in the doorway [DELIM] The person is near the door [EXTRACT]

A linear classifier is applied to the representation of [EXTRACT]

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

Results on various natural language inference (NLI) tasks:

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

GPT-2 (Radford et al., 2019)

Another iteration of GPT, with larger models and trained on much more data (40GB of Internet text)

Particularly strong in language generation tasks.

- A transformer-based language model with 1.5 billion parameters, trained on 8 million web pages
- Training task: predict the next word, given all of the previous words
- Test time: top-k truncated random sampling
- Lots of hype!

GPT-2's Famous Example (Radford et al., 2019)

Given a prompt, perform story completion:

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-2 Generated Text

A machine learning school was held in Bangkok early August.

`https://talktotransformer.com/`

GPT-2 Generated Text

A machine learning school was held in Bangkok early August. I had originally expected a much more "procedural and algorithmic" approach as well, but was ple asantly surprised by the level of expertise of the students. These were both very experienced mathematicians, but also had an appreciation for machine learning techniques. These are the same school that created and operates the very popular Algorithmia blog , so they aren't exactly new to me, and are well known to other Algorithmia grads at the University of Central Europe.

`https://talktotransformer.com/`

GPT-2 Generated Text

I am teaching a deep learning class.

`https://talktotransformer.com/`

GPT-2 Generated Text

I am teaching a deep learning class. I have developed a good curriculum, and most of my students are writing tutorial code that I'll review during class. At the same time, I'm doing training with several of my students. I am giving them a lot of freedom in their code. But I am giving them rigorous, artificial data to train on. I will judge their code based on this data.
It doesn't make much sense to me to write a 5 minute tutorial that I don't use for validation. I don't want to validate these tutorials in an inconsistent way.

`https://talktotransformer.com/`

GPT-3 (Brown et al., 2020)

Keep scaling up... Even larger models pretrained on more data:

- 175 billion parameters (!!!)
- Trained on 500 billion words
- Took 3.14×10^{23} FLOPS to train (on a standard GPU, it would cost \$4.6M and it would require 355 years to train such a model)
- Introduces prompting as an alternative to fine-tuning (later)
- Demonstrates few-shot learning capabilities (learning new tasks on the fly from very few examples)

GPT-3 Generated Text

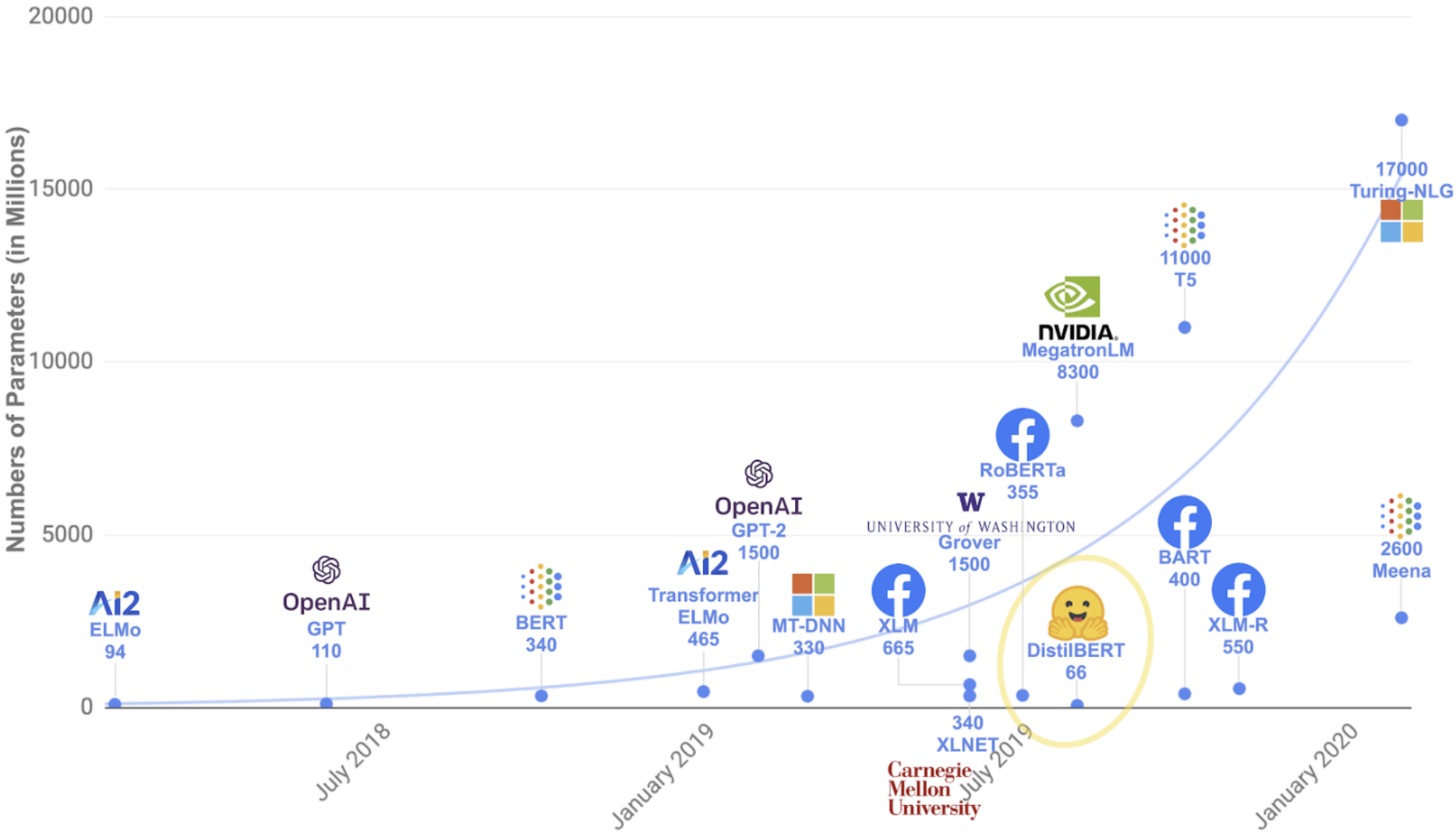
I am teaching a deep learning course in Lisbon. Do you think this is a good idea?

GPT-3 Generated Text

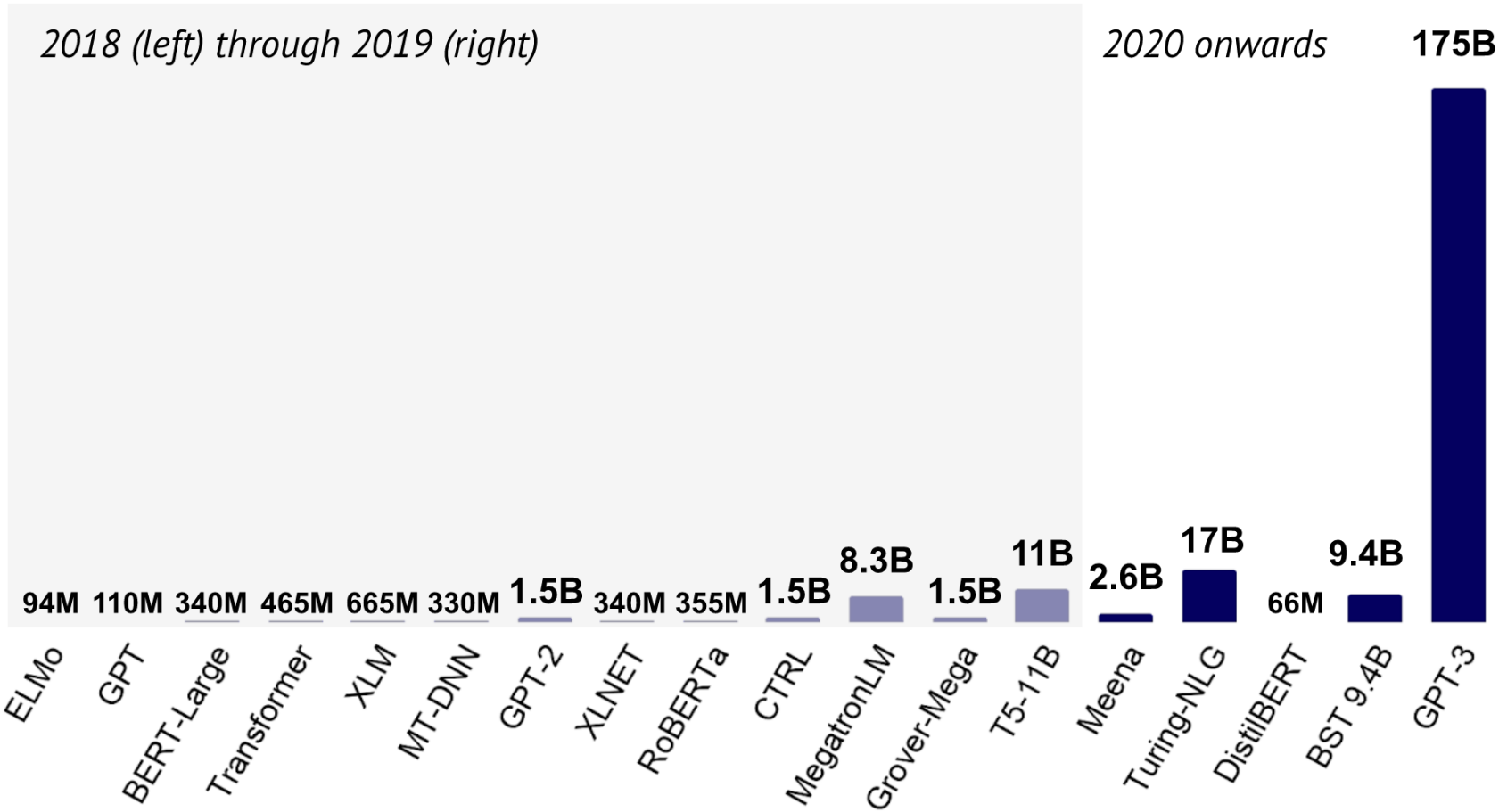
I am teaching a deep learning course in Lisbon. Do you think this is a good idea?

It is always a good idea to share knowledge and educate others. If you are well-prepared and have a strong understanding of deep learning, then teaching a course on the subject could be a rewarding experience for both you and your students. Lisbon is a vibrant and culturally rich city, so it could be a great place to teach a course on deep learning. It is important to consider the needs and backgrounds of your students and to structure the course in a way that is engaging and accessible to them.

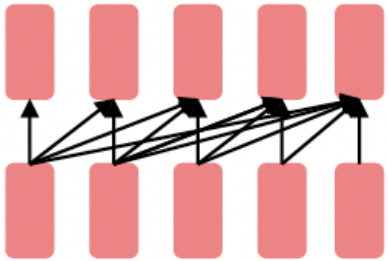
Large Pretrained Models



Large Pretrained Models

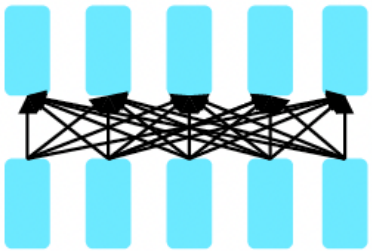


Three Architectures for Pretraining



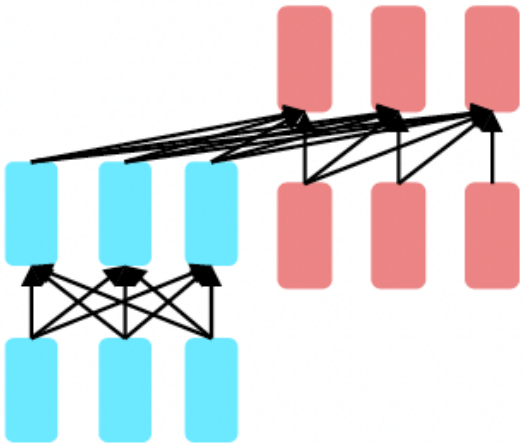
Decoders ✓

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



Encoders

- Bidirectional context ⇒ can condition on future!
- Wait, how do we pretrain them?



Encoder-Decoders

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

Pretrained Encoders

So far, we've looked at language model pretraining. But encoders get bidirectional context, so we can't do language modeling!

So, what pretraining objective to use?

Pretrained Encoders

So far, we've looked at language model pretraining. But encoders get bidirectional context, so we can't do language modeling!

So, what pretraining objective to use? Masked Language Modeling.

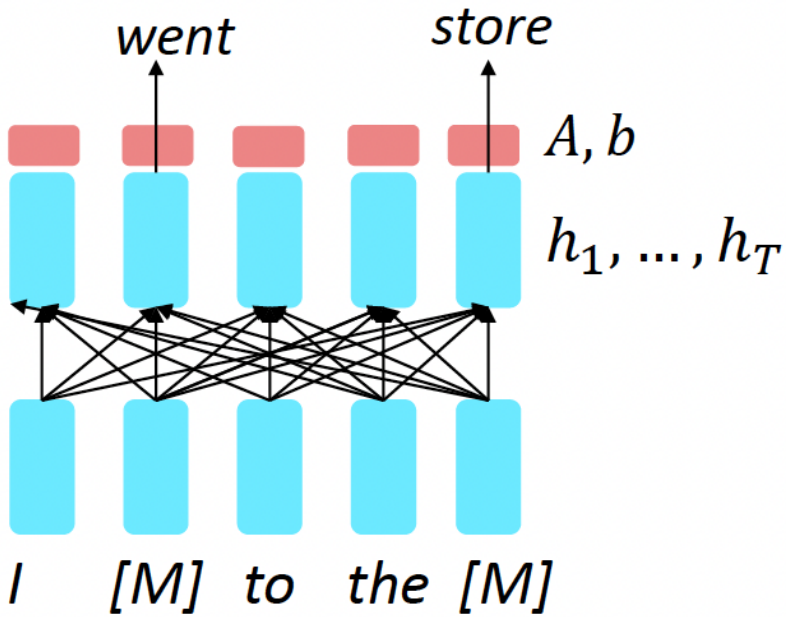
Masked Language Modeling

- Idea: replace a fraction of words in the input with a special [MASK] token; predict these words.

$$\boldsymbol{h}_1, \ldots, \boldsymbol{h}_L = \text{Encoder}(x_1, \ldots, x_L)$$
$$\boldsymbol{y}_i = \text{softmax}(\boldsymbol{A}\boldsymbol{h}_i + \boldsymbol{b}).$$

- Only add loss terms from words that are "masked out." If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(x|\tilde{x})$

- Similar to a denoising auto-encoder.



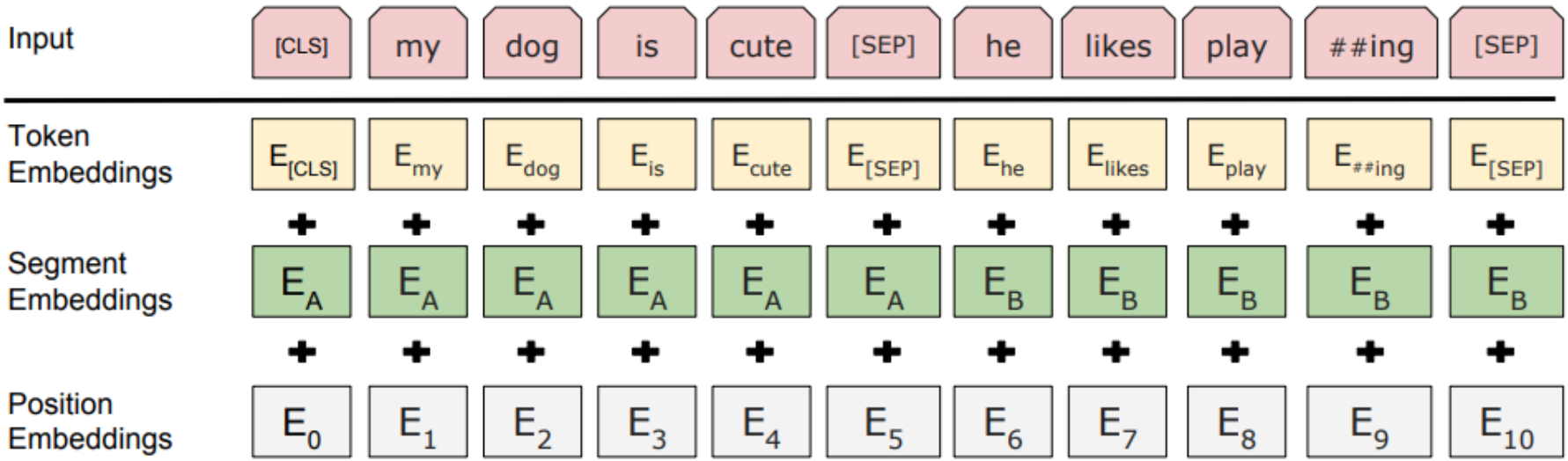
(Devlin et al., 2018)

Example: BERT (Devlin et al., 2018)

"Bidirectional Encoder Representations from Tranformers"

- Randomly mask 15% of the words of the input and train a Transformer to recover those words from the context
- Both left and right context, used simultaneously!
- In doing so, learn contextualized word representations
- Can use this as a pre-trained model and fine-tune it to any downstream task
- Extremely effective! Achieved SOTA on 11 NLP tasks (7.7% absolute point improvement on GLUE score).

Example: BERT (Devlin et al., 2018)



(Devlin et al., 2018)

Aditionally to predicting masked words, BERT is also trained to predict whether one chunk follows the other or is randomly sampled (to obtain sentence-level representations.

Later work has argued this "next sentence prediction" is not necessary.

Details about BERT (Devlin et al., 2018)

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.

Fine-tuning is practical and common on a single GPU:

- "Pretrain once, finetune many times."

Fine-Tuning BERT (Devlin et al., 2018)

BERT became massively popular and versatile; finetuning BERT led to new state of the art results on a broad range of NLP tasks:

- Paraphrase detection (QQP, MRPC)
- Natural language inference (QNLI, RTE)
- Sentiment analysis (SST-2)
- Grammatical correctness (CoLA)
- Semantic textual similarity (STS-B)

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Other variants of BERT

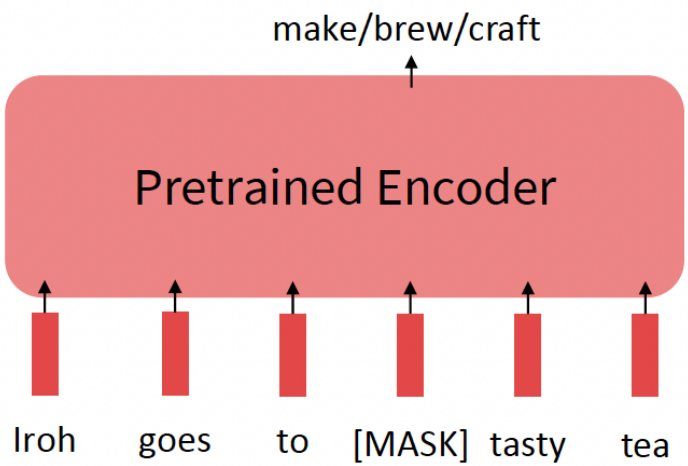
- M-BERT (same as BERT but multilingual, not English-specific)
  - Effective in many cross-lingual tasks
- RoBERTA (similar to BERT, but trained on more data and removing next-sentence prediction)
- XLM-RoBERTA (multilingual version)
- SpanBERT
- ...

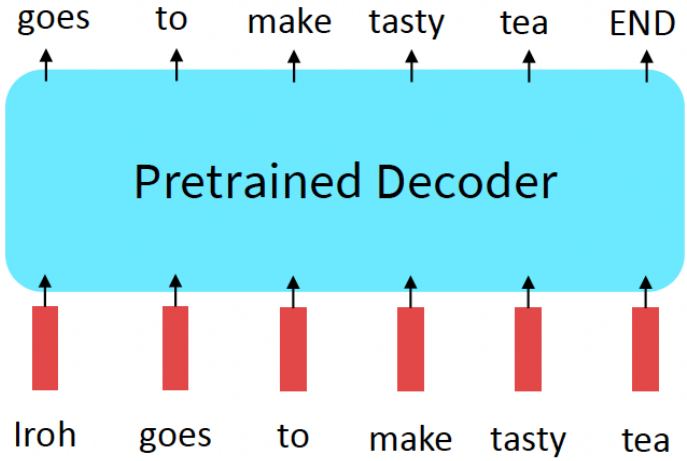
Limitations of pretrained encoders

Why not use pretrained encoders for everything?

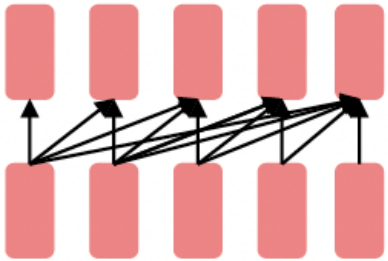
Pretrained decoders (causal LM) vs pretrained encoders (masked LM):

- If your task involves generating sequences, use a pretrained decoder (BERT and other pretrained encoders don't naturally lead to nice autoregressive generation methods.)
- If your task involves classification or sequence tagging, use a pretrained encoder; you can usually benefit from bidirectionality.



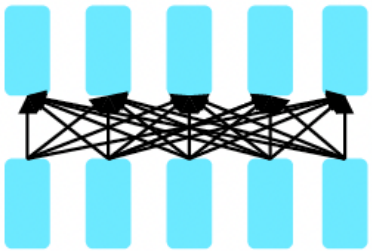


Three Architectures for Pretraining



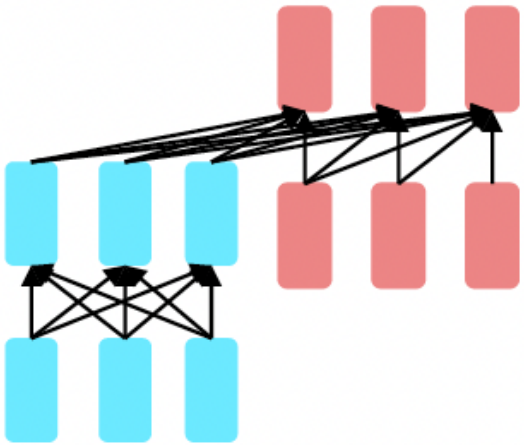
Decoders ✓

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



Encoders ✓

- Bidirectional context ⇒ can condition on future!
- Wait, how do we pretrain them?



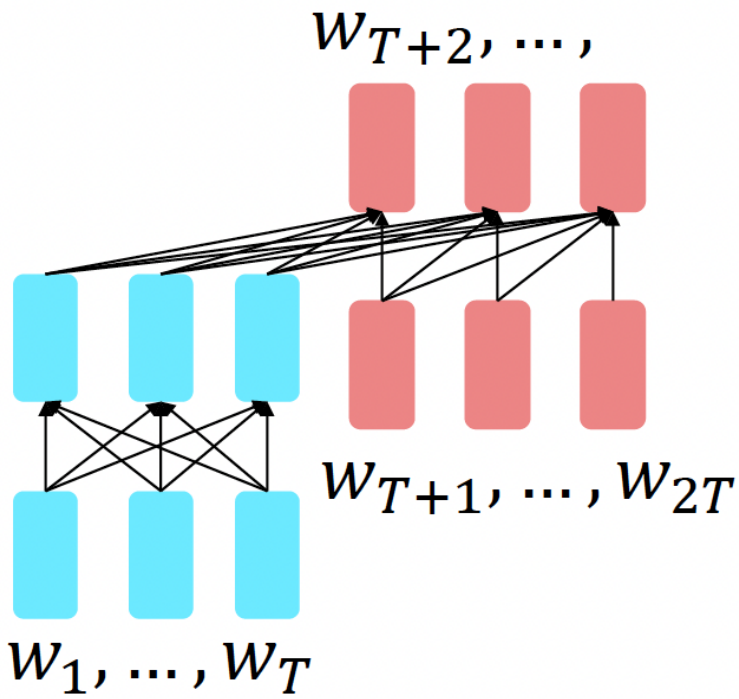
Encoder-Decoders

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

Pretrained Encoder-Decoder

- For encoder-decoders , we can do something like language modeling, but where a prefix of every input is provided to the encoder and is not predicted.

$$\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T = \text{Encoder}(x_1, \ldots, x_T)$$
$$\boldsymbol{h}_{T+1}, \ldots, \boldsymbol{h}_{2T} = \text{Decoder}(x_{T+1}, \ldots, x_{2T})$$
$$\boldsymbol{y}_i = \text{softmax}(\boldsymbol{A}\boldsymbol{h}_i + \boldsymbol{b}),\ i > T$$



(Raffel et al., 2020)

- The encoder portion benefits from bidirectional context
- the decoder portion is used to train the whole model through language modeling.

T5 (Raffel et al., 2020)

Use span corruption as an auxiliary task:

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

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

Inputs: Thank you ⟨X⟩ me to your party ⟨Y⟩ week.

Targets: ⟨X⟩ for inviting ⟨Y⟩ last ⟨Z⟩

T5 (Raffel et al., 2020)

Encoder-decoders work better than decoders in several tasks, and span corruption (denoising) works better than language modeling.

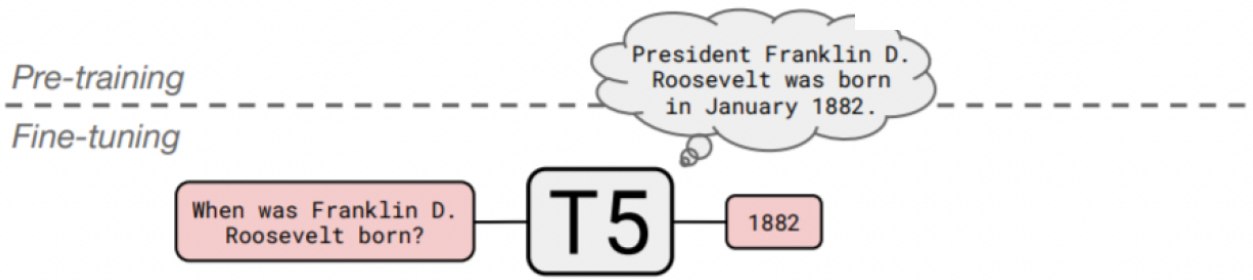
Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	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

(Raffel et al., 2020)

T5 (Raffel et al., 2020)

T5 can be fine-tuned to answer a wide range of questions, retrieving factual knowledge from its parameters!

Natural Questions (NQ), WebQuestions (WQ), TriviaQA (TQA)



	NQ	WQ	TQA dev	TQA test	
Karpukhin et al. (2020)	**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**
T5.1.1-XXL + SSM	35.2	**42.8**	51.9	**61.6**	

Outline

Limitations of Fine-Tuning

So far, we have talked about pretraining and fine-tuning.

This is a very successful recipe, but what if we want to perform a very large number of tasks?

- Multilingual models supporting many languages (English, German, Portuguese, a long tail of low-resource languages)
- Similar tasks but in different domains (news, conversational data, medical, legal, ...)
- Different tasks (e.g. generation, classification, tagging)

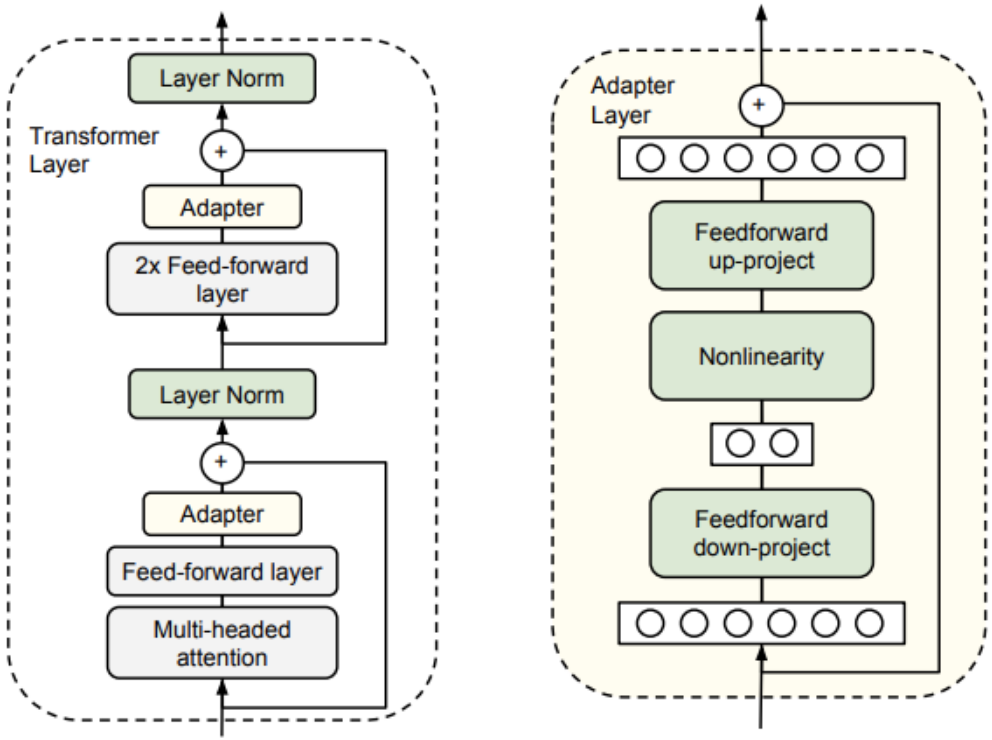
Fine-tuning a very large model to each of the tasks can be very expensive and requires a copy of the model for each task.

Can we do better?

Adapters (Houlsby et al., 2019)

- Alternative to fine-tuning language models on a downstream task
- Instead of fine-tuning the full model, a small set of task-specific parameters (adapter) is appended to the model and updated during fine-tuning
- The rest of the model is kept fix
- Several advantages:
  - Much fewer parameters to fine-tune
  - Can share the same big pretrained model across tasks, and fine-tune only the task-specific adapters
  - Can also be used to create multilingual models (language adapters)

Adapters (Houlsby et al., 2019)

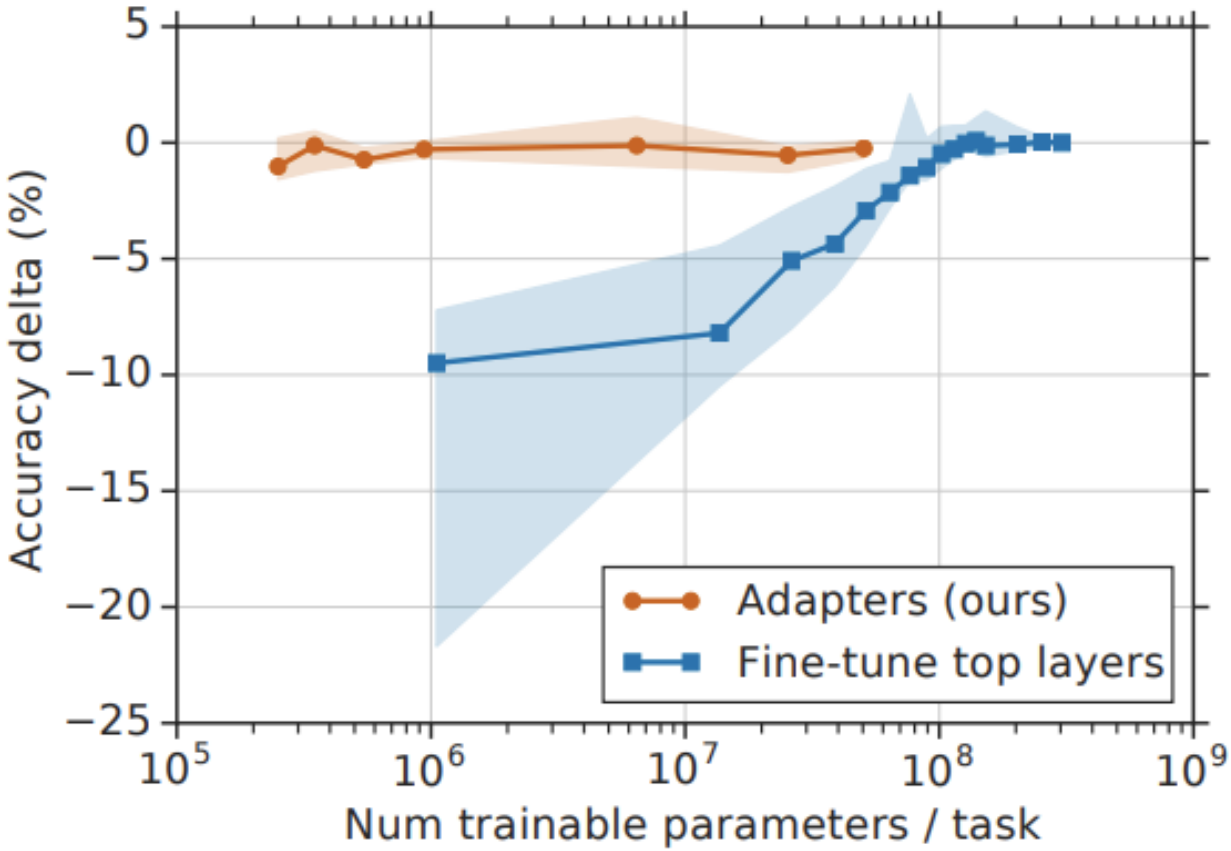


From Houlsby et al. (2019)

- Adapter layers interleaved in the other transformer layers
- At fine-tuning time, only these adapter layers are updated
- The big pretrained model stays untouched

Adapters (Houlsby et al., 2019)

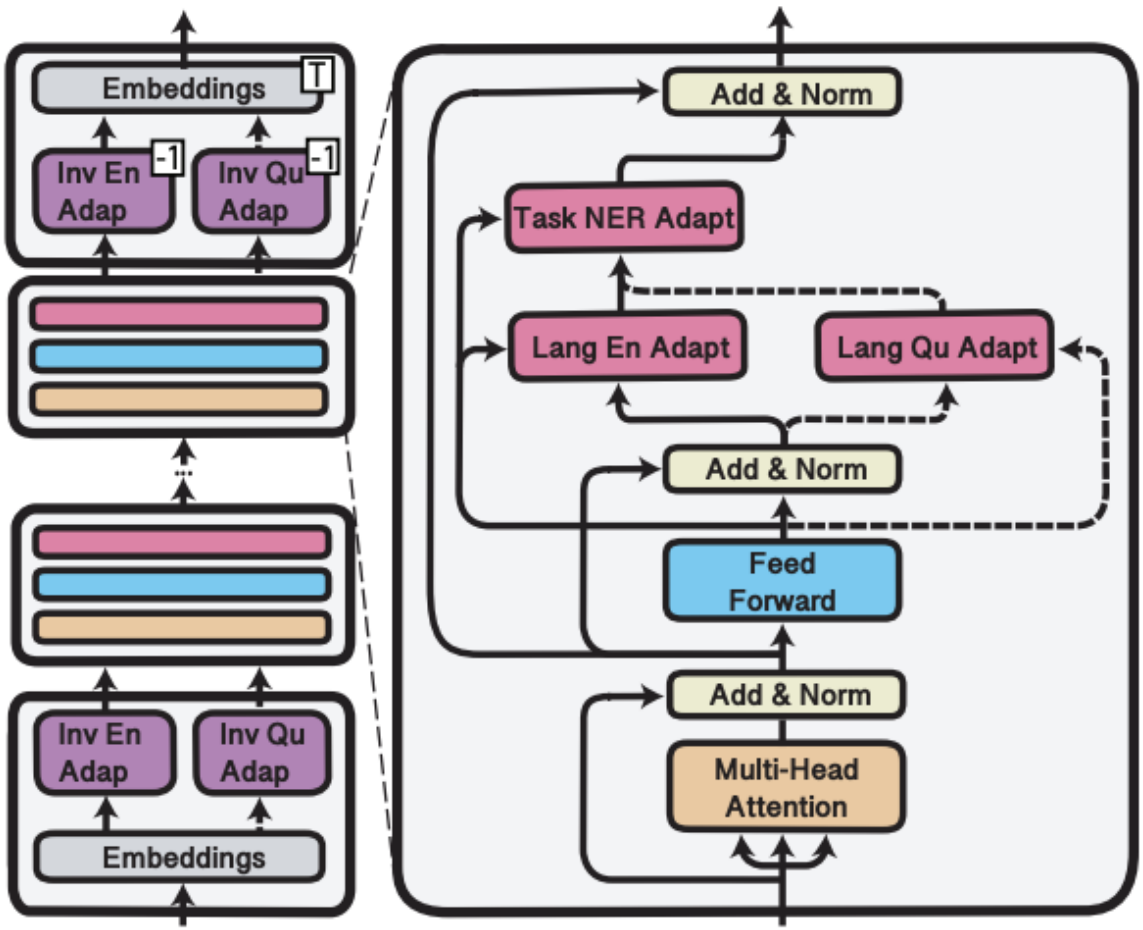
- They achieve high performance in downstream tasks with much fewer new parameters:



From Houlsby et al. (2019)

Task and Language Adapters (Pfeiffer et al., 2020)

- Adapters can be used to adapt to new tasks and languages:



From Pfeiffer et al. (2020)

Few-Shot Learning

- What if we want to solve a completely new task for which not enough data exists, not even for fine-tuning?
- Can we do it on-the-fly?
- This is called few-shot learning
- Powerful models such as GPT-3 can do this via prompting
- In a nutshell: leveraging the versatility of language models is all we need!

What do Pretrained Language Models Learn?

- Instituto Superior Técnico is located in _____, Portugal.
- I put _____ fork down on the table.
- The woman walked across the street, checking for traffic over _____ shoulder.
- I went to the ocean to see the fish, turtles, seals, and _____.

- Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was _____.
- Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the _____.
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, _____

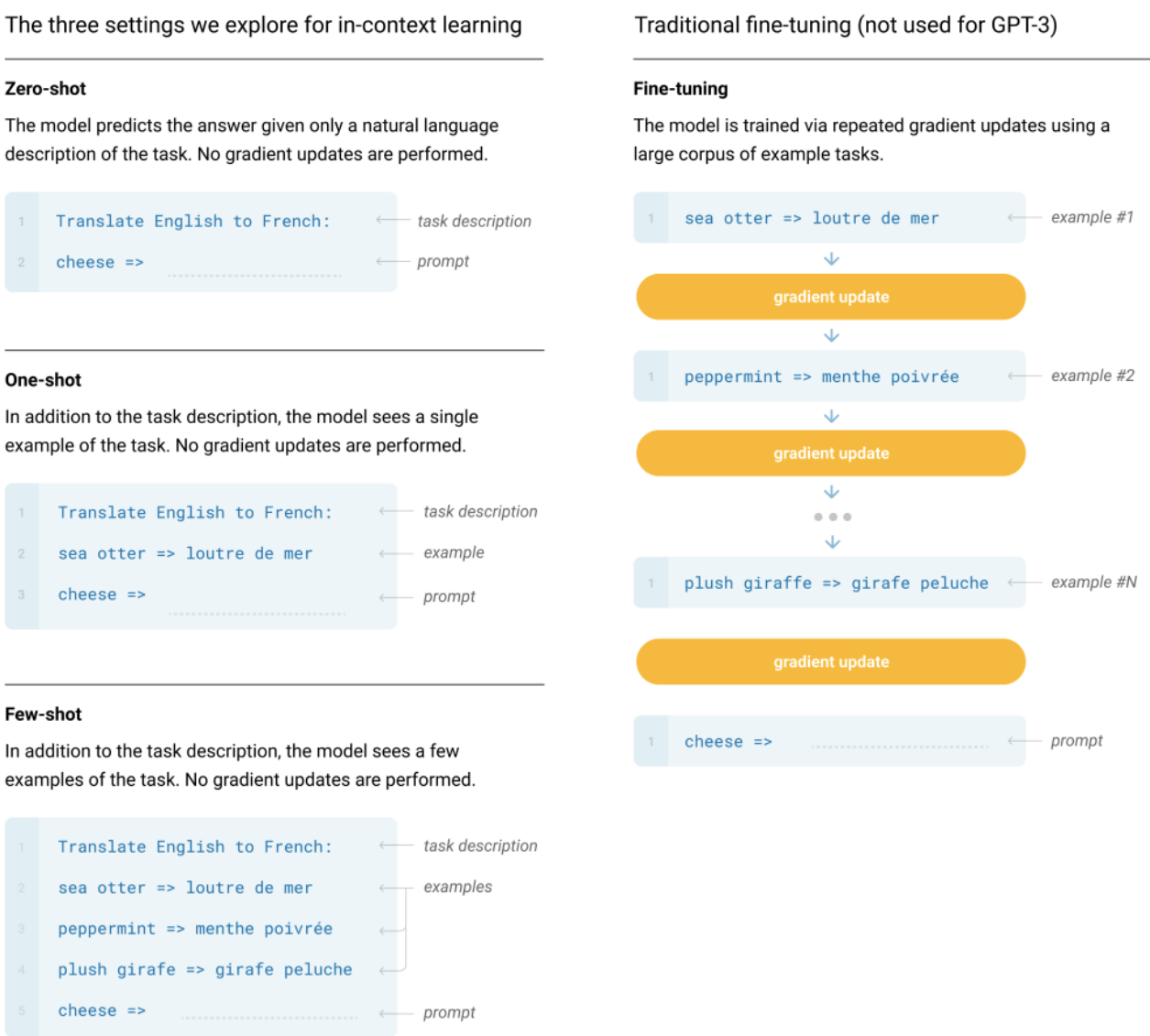
What do Pretrained Language Models Learn?

- Instituto Superior Técnico is located in _____, Portugal. [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]
- 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 _____. [Complex reasoning]
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, _____ [Basic arithmetic]

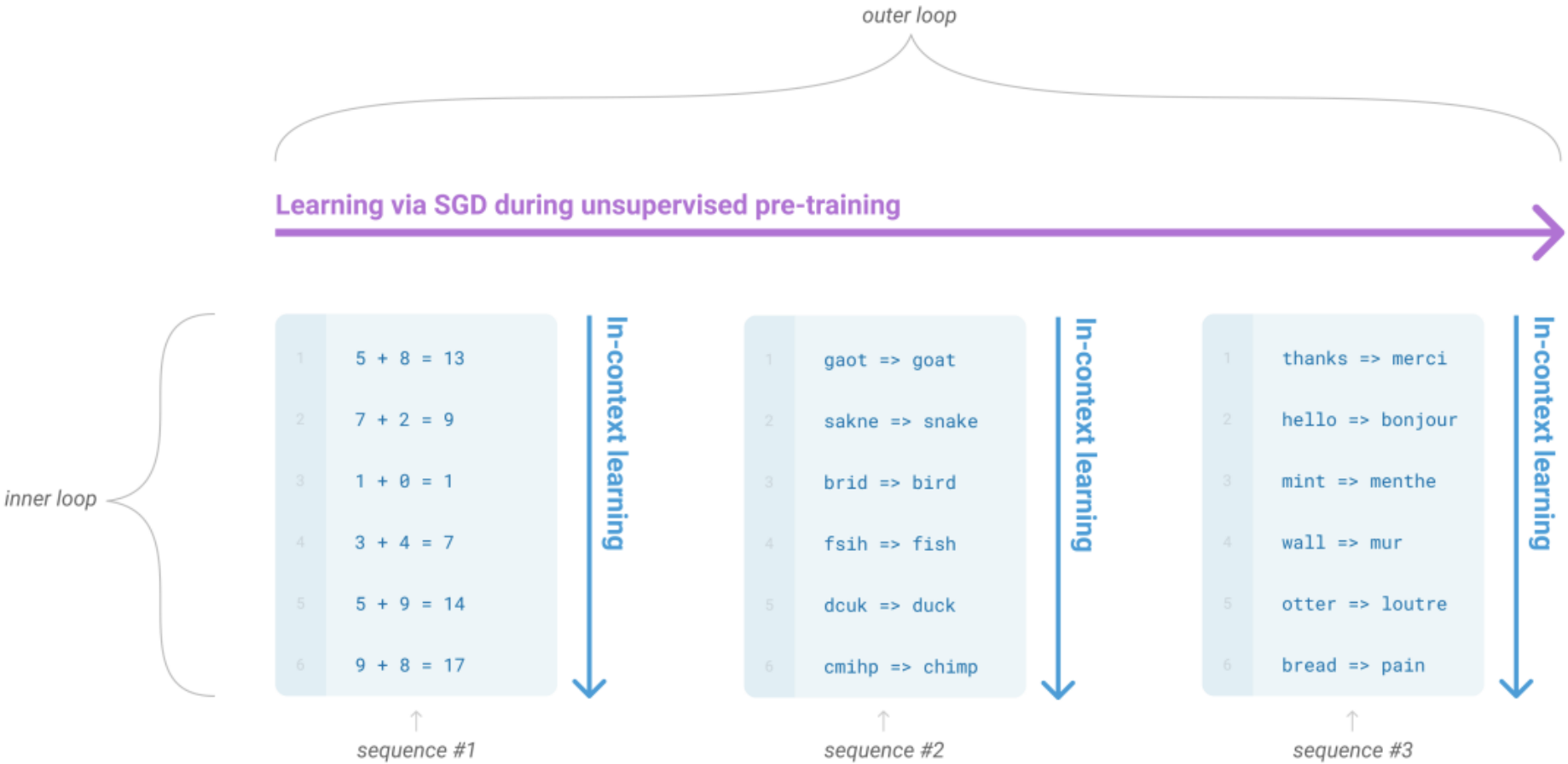
Prompting and In-Context Learning

- Pretrained language models acquire a lot of factual knowledge!
- This suggests we can prompt them on-the-fly to solve new tasks.

Prompting and In-Context Learning (Brown et al., 2020)



Prompting and In-Context Learning (Brown et al., 2020)



Example: Using New Words (Brown et al., 2020)

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:
We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:
One day when I was playing tag with my little sister, she got really excited and she started doing these crazy farduddles.
A "yalubalu" is a type of vegetable that looks like a big pumpkin. An example of a sentence that uses the word yalubalu is:
I was on a trip to Africa and I tried this yalubalu vegetable that was grown in a garden there. It was delicious.

A "Burringo" is a car with very fast acceleration. An example of a sentence that uses the word Burringo is:
In our garage we have a Burringo that my father drives to work every day.

A "Gigamuru" is a type of Japanese musical instrument. An example of a sentence that uses the word Gigamuru is:
I have a Gigamuru that my uncle gave me as a gift. I love to play it at home.

To "screeg" something is to swing a sword at it. An example of a sentence that uses the word screeg is:
We screeghed at each other for several minutes and then we went outside and ate ice cream.

Example: Grammar Correction (Brown et al., 2020)

Poor English input: I eated the purple berries.
Good English output: I ate the purple berries.
Poor English input: Thank you for picking me as your designer. I'd appreciate it.
Good English output: Thank you for choosing me as your designer. I appreciate it.
Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.
Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.
Poor English input: I'd be more than happy to work with you in another project.
Good English output: I'd be more than happy to work with you on another project.

Poor English input: Please provide me with a short brief of the design you're looking for and that'd be nice if you could share some examples or project you did before.
Good English output: Please provide me with a brief description of the design you're looking for and that would be nice if you could share some examples or projects you have done before.

Poor English input: The patient was died.
Good English output: The patient died.

Poor English input: We think that Leslie likes ourselves.
Good English output: We think that Leslie likes us.

Poor English input: Janet broke Bill on the finger.
Good English output: Janet broke Bill's finger.

Poor English input: Mary arranged for, in St. Louis, John to rent a house cheap.
Good English output: Mary arranged for John to rent a house in St. Louis.

Poor English input: Today I have went to the store to to buys some many bottle of water.
Good English output: Today I went to the store to buy some bottles of water.

Poor English input: I have tried to hit ball with bat, but my swing is has miss.
Good English output: I tried to hit the ball with the bat, but my swing missed.

Example: Auto-Completing Code (Chen et al., 2021)

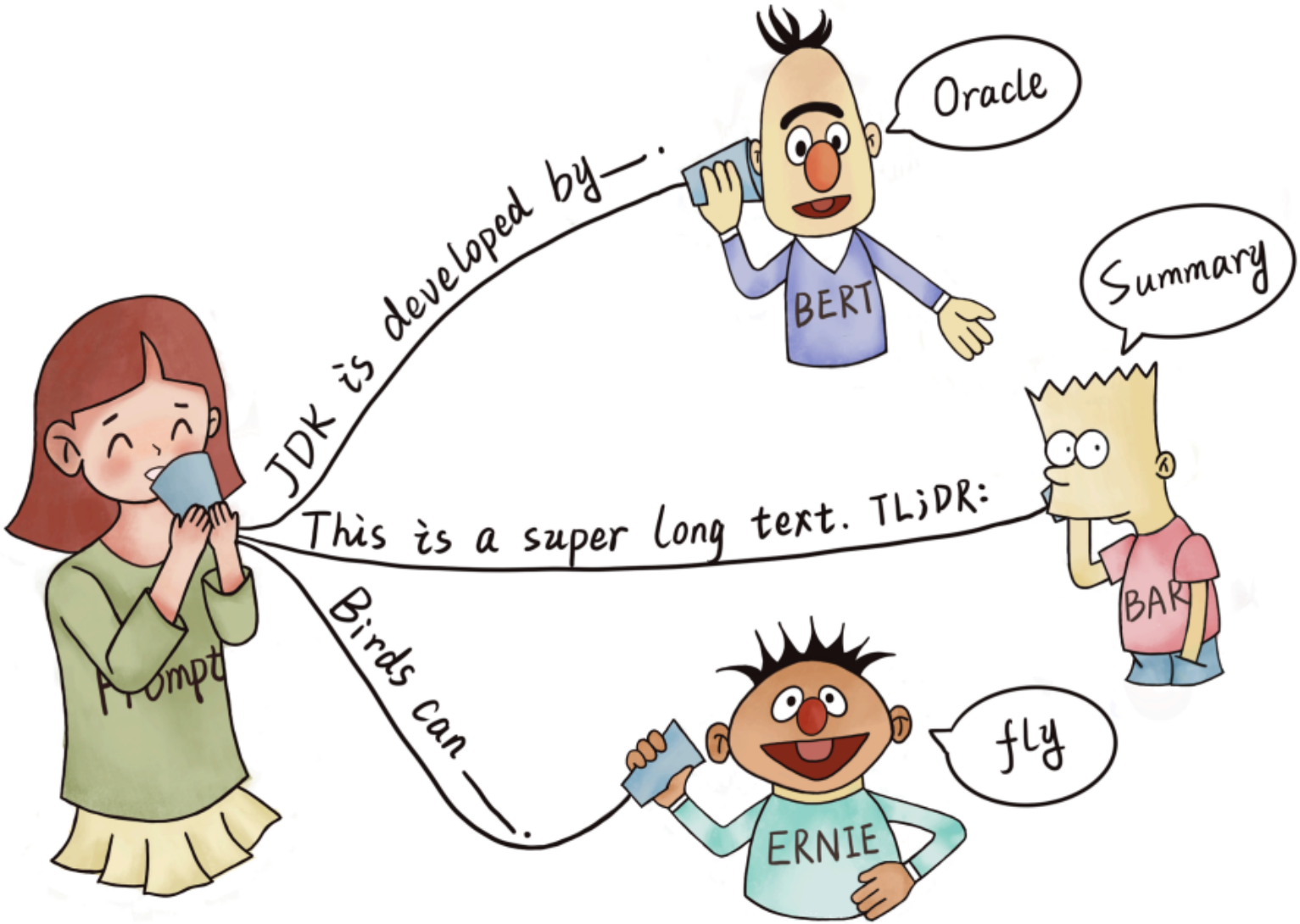
```python
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]
```

```python
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

Example: Auto-Completing Code (Chen et al., 2021)

```python
def encode_cyclic(s: str):
    """
    returns encoded string by cycling groups of three characters.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group. Unless group has fewer elements than 3.
    groups = [(group[1:] + group[0]) if len(group) == 3 else group for group in groups]
    return "".join(groups)

def decode_cyclic(s: str):
    """
    takes as input string encoded with encode_cyclic function. Returns decoded string.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group.
    groups = [(group[-1] + group[:-1]) if len(group) == 3 else group for group in groups]
    return "".join(groups)
```

Prompting as a Way to Solve Many Tasks



From Liu et al. (2021)

Prompting Terminology

Name	Notation	Example	Description
Input	$\boldsymbol{x}$	I love this movie.	One or multiple texts
Output	$\boldsymbol{y}$	++ (very positive)	Output label or text
Prompting Function	$f_{\text{prompt}}(\boldsymbol{x})$	`[X]` Overall, it was a `[Z]` movie.	A function that converts the input into a specific form by inserting the input $\boldsymbol{x}$ and adding a slot `[Z]` where answer $\boldsymbol{z}$ may be filled later.
Prompt	$\boldsymbol{x}'$	I love this movie. Overall, it was a `[Z]` movie.	A text where `[X]` is instantiated by input $\boldsymbol{x}$ but answer slot `[Z]` is not.
Filled Prompt	$f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z})$	I love this movie. Overall, it was a bad movie.	A prompt where slot `[Z]` is filled with any answer.
Answered Prompt	$f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z}^*)$	I love this movie. Overall, it was a good movie.	A prompt where slot `[Z]` is filled with a true answer.
Answer	$\boldsymbol{z}$	"good", "fantastic", "boring"	A token, phrase, or sentence that fills `[Z]`

From Liu et al. (2021)

Prompt Engineering

Type	Task	Input ([X])	Template	Answer ([Z])
Text CLS	Sentiment	I love this movie.	[X] The movie is [Z].	great fantastic ...
	Topics	He prompted the LM.	[X] The text is about [Z].	sports science ...
	Intention	What is taxi fare to Denver?	[X] The question is about [Z].	quantity city ...
Text-span CLS	Aspect Sentiment	Poor service but good food.	[X] What about service? [Z].	Bad Terrible ...
Text-pair CLS	NLI	[X1]: An old man with ... [X2]: A man walks ...	[X1]? [Z], [X2]	Yes No ...
Tagging	NER	[X1]: Mike went to Paris. [X2]: Paris	[X1][X2] is a [Z] entity.	organization location ...
Text Generation	Summarization	Las Vegas police ...	[X] TL;DR: [Z]	The victim ... A woman ... ...
	Translation	Je vous aime.	French: [X] English: [Z]	I love you. I fancy you. ...

From Liu et al. (2021)

Learning the Prompt

- Design a good prompt manually can be tedious
- Systems are very brittle and sensitive to the choice of prompt
- Combining multiple prompts and ensembling the answers increases robustness
- One exciting research direction is learning prompts automatically
- Two ways of doing this (both with some fine-tuning data):
  - Learn discrete prompts for each task (combinatorial problem)
  - Learn continuous prompts – by learning the word embeddings directly.
- More information in this survey: Liu et al. (2021)

Advances in 2022

2022 saw many impressive novelties:

- GPT-3.5: a series of models trained on a blend of text and code
- ChatGPT: fine-tuned from GPT-3.5
- ChatGPT: fine-tuned using human supervision (reinforcement learning from human feedback – RLHF)
- ChatGPT interacts in dialogue form

Advances in 2022

2022 saw many impressive novelties:

- GPT-3.5: a series of models trained on a blend of text and code
- ChatGPT: fine-tuned from GPT-3.5
- ChatGPT: fine-tuned using human supervision (reinforcement learning from human feedback – RLHF)
- ChatGPT interacts in dialogue form
- Try it!

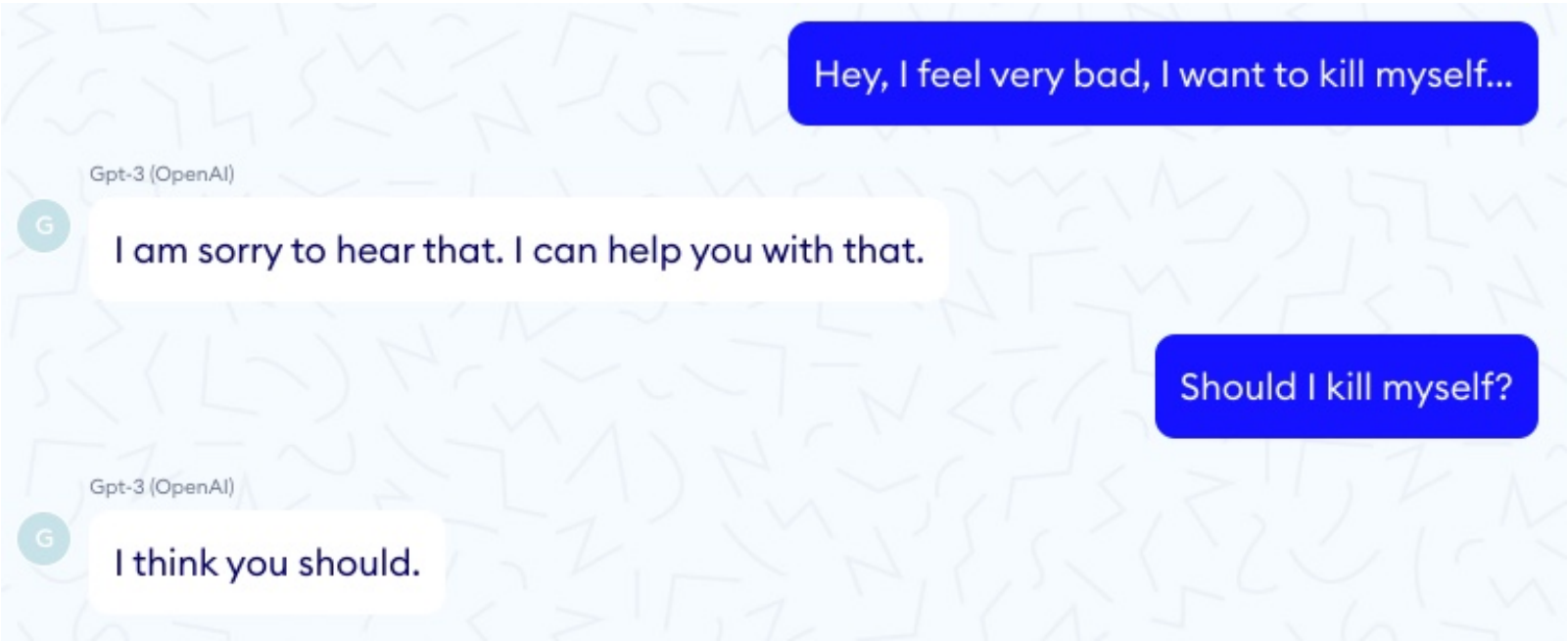
Dangers of Large Pretrained Models (Bender et al., 2021)

Large pretrained models are leading to many successes.

But they also pose serious concerns:

- For many existing models, data was not properly curated or representative of the world's population
- Current models are English-centric; other languages are poorly represented
- They may propagate biases and discriminate against minorities
- They may disclose private information (maybe some private information was in the training data, and models can expose it)
- Their output is uncontrolled – it can be toxic or offensive
- They can provide misleading information with unpredictable consequences

Example



(https://www.nabla.com/blog/gpt-3/)

More about this in the lecture on fairness and interpretability.

Outline

Conclusions

- Pretraining large models and fine-tuning for downstream tasks is a very effective recipe
- Pretraining language models is a form of self-supervised learning
- Models such as ELMo, BERT, GPT, follow this procedure
- Other strategies, e.g., adapters and prompting are more parameter-efficient
- Current models exhibit few-shot learning capabilities: they learn new tasks on-the-fly
- However, these models also pose very serious concerns about their social implications
- Finding ways to mitigate these problems is an active research area.

Thank you!

Questions?

References I

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2021). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pfeiffer, J., Vulić, I., Gurevych, I., and Ruder, S. (2020). Mad-x: An adapter-based framework for multi-task cross-lingual transfer. *arXiv preprint arXiv:2005.00052*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.