CS11-711 Advanced NLP

Language Model Pre-training

Graham Neubig and Lucio Dery

Carnegie Mellon University
Language Technologies Institute

Site
https://phontron.com/class/anlp2022/

(w/ slides by Antonis Anastasopoulos)

Multi-task Learning Overview

Terminology

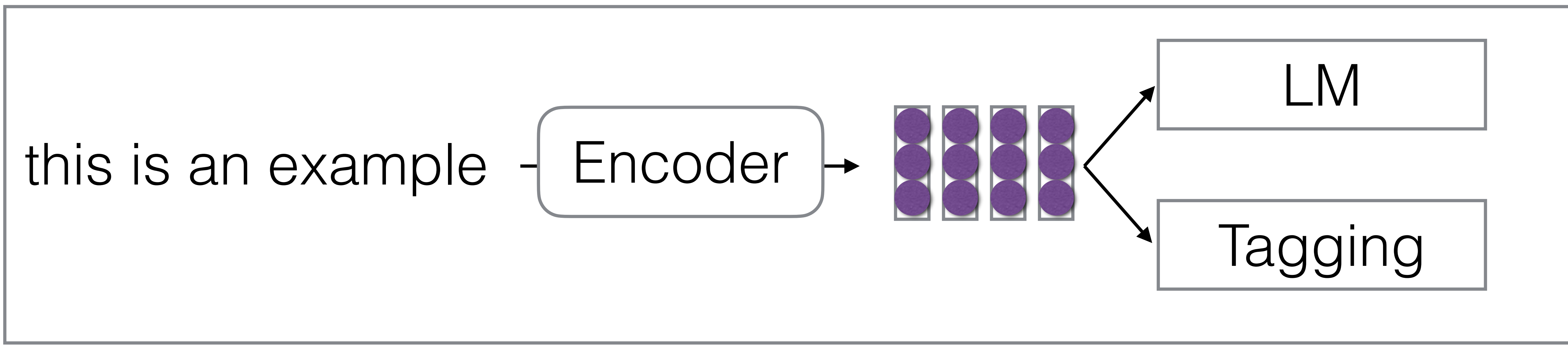
- **Multi-task learning** is a general term for training on multiple tasks
- **Transfer learning** is a type of multi-task learning where we only really care about one of the tasks
- **Pre-training** is a type of transfer learning where one objective is used first
- **Few-shot, zero-shot learning** indicates learning to perform a task with very few, or zero labeled examples for that task

Plethora of Tasks in NLP

- In NLP, there are a plethora of tasks, each requiring different varieties of data
  - **Only text:** e.g. language modeling
  - **Naturally occurring data:** e.g. machine translation
  - **Hand-labeled data:** e.g. most analysis tasks
- And each in many languages, many domains!

Standard Multi-task Learning

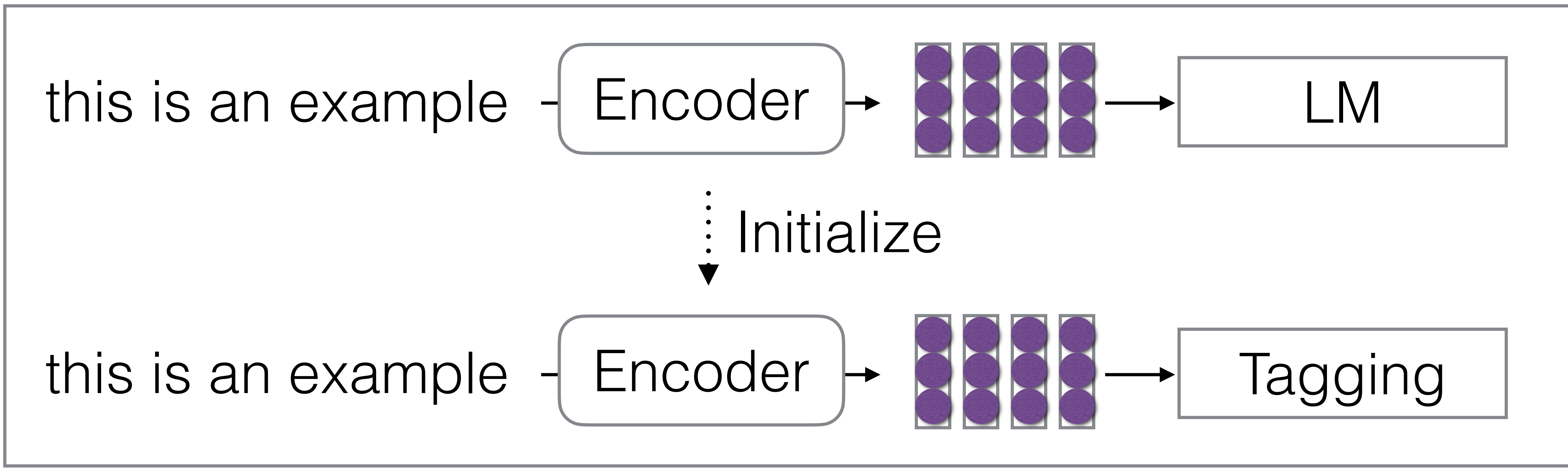
- Train representations to do well on multiple tasks at once



- Often as simple as randomly choosing minibatch from one of multiple tasks

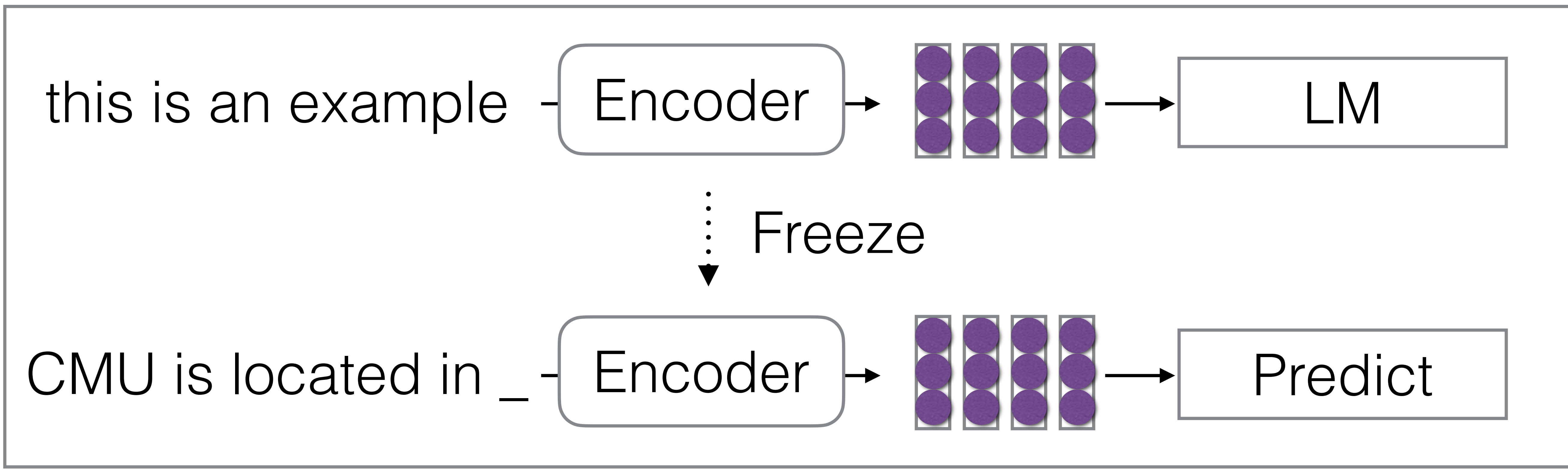
Pre-train and Fine-Tune

- First train on one task, then train on another



Prompting

- Train on LM task, make predictions in textualized tasks



Thinking about Pre-trained Models

Thinking about Pre-trained LMs

- Many pre-trained LMs have names like BERT, RoBERTa, GPT-3, PaLM
- These often refer to a combination of
  - **Model:** The underlying neural network architecture
  - **Training objective:** What objective is used to pre-train
  - **Data:** What data the authors chose to use to train the model
- They papers presenting the models are also often notable for **experimental results**

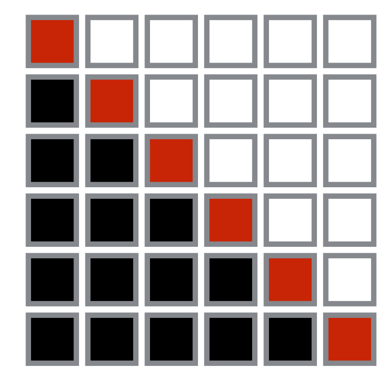
Which Model?

- Usually **Transformer**, although the details vary
- **Size** is an all-important parameter, bigger is usually more performant
- **Model details** sometimes vary (or are underspecified)

Which Objective?

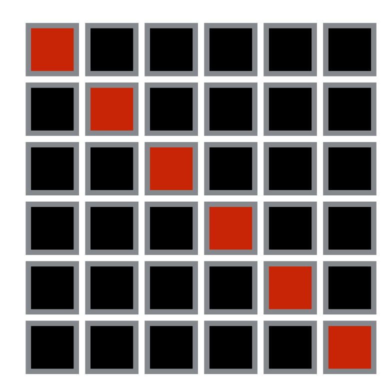
- Two most common varieties
  - **Auto-regressive language modeling** -> used more for prompting/text generation

$$P(X) = \prod_{i=1}^{|X|} P(x_i | x_1, \ldots, x_{i-1})$$



  - **Masked language modeling** -> used more for pre-train + fine-tune

$$P(X) \neq \prod_{i=1}^{|X|} P(x_i | x_{\neq i})$$



Which Data?

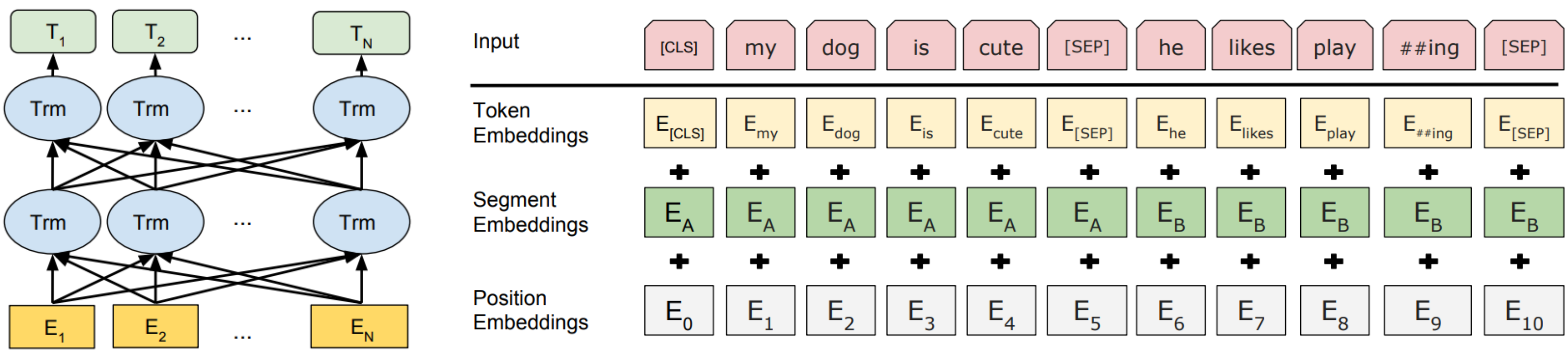
- Data is extremely important, common sources
  - **Books corpus** - a large corpus of books
  - **Wikipedia**
  - **Common crawl** - data from the whole internet

Representation Learning through LMs

Masked Language Modeling (BERT)

(Devlin et al. 2018)

- **Model:** Multi-layer self-attention. Input sentence or pair, w/ [CLS] token, subword representation



- **Objective:** Masked word prediction + next-sentence prediction
- **Data:** BooksCorpus + English Wikipedia

Masked Word Prediction
(Devlin et al. 2018)

1. predict a masked word
   - 80%: substitute input word with [MASK]
   - 10%: substitute input word with random word
   - 10%: no change

- Like context2vec, but **better suited for multi-layer self attention**

Consecutive Sentence Prediction
(Devlin et al. 2018)

1. classify two sentences as consecutive or not:
    - 50% of training data (from OpenBooks) is "consecutive"

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

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

Hyperparameter Optimization/Data (RoBERTa)

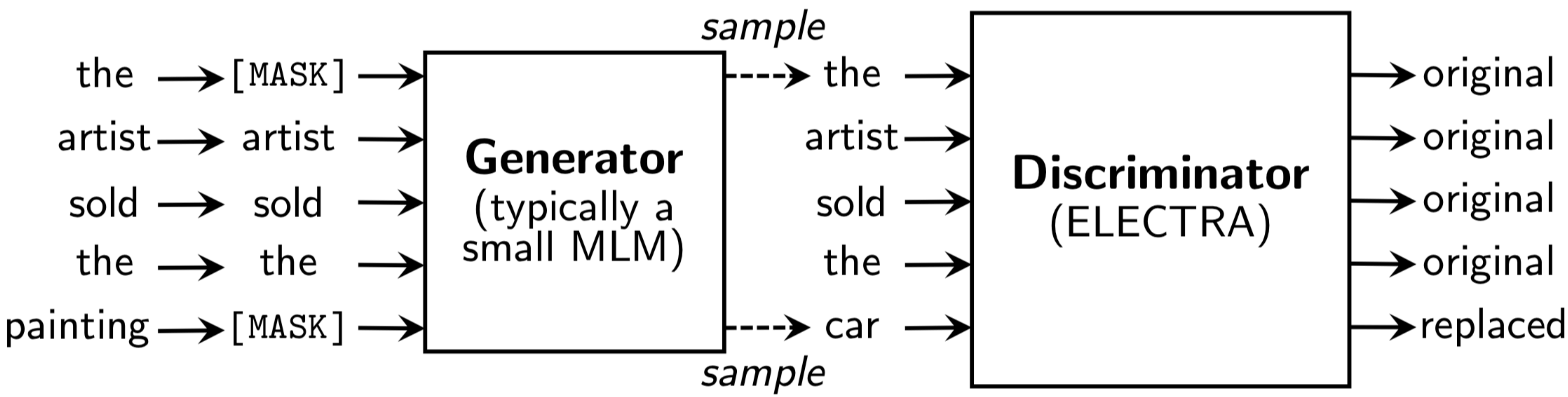
(Liu et al. 2019)

- **Model:** Same as BERT
- **Objective:** Same as BERT, but *train longer* and *drop sentence prediction* objective
- **Data:** BooksCorpus + English Wikipedia
- **Results:** are empirically much better than BERT

Distribution Discrimination (ELECTRA)

(Clark et al. 2020)

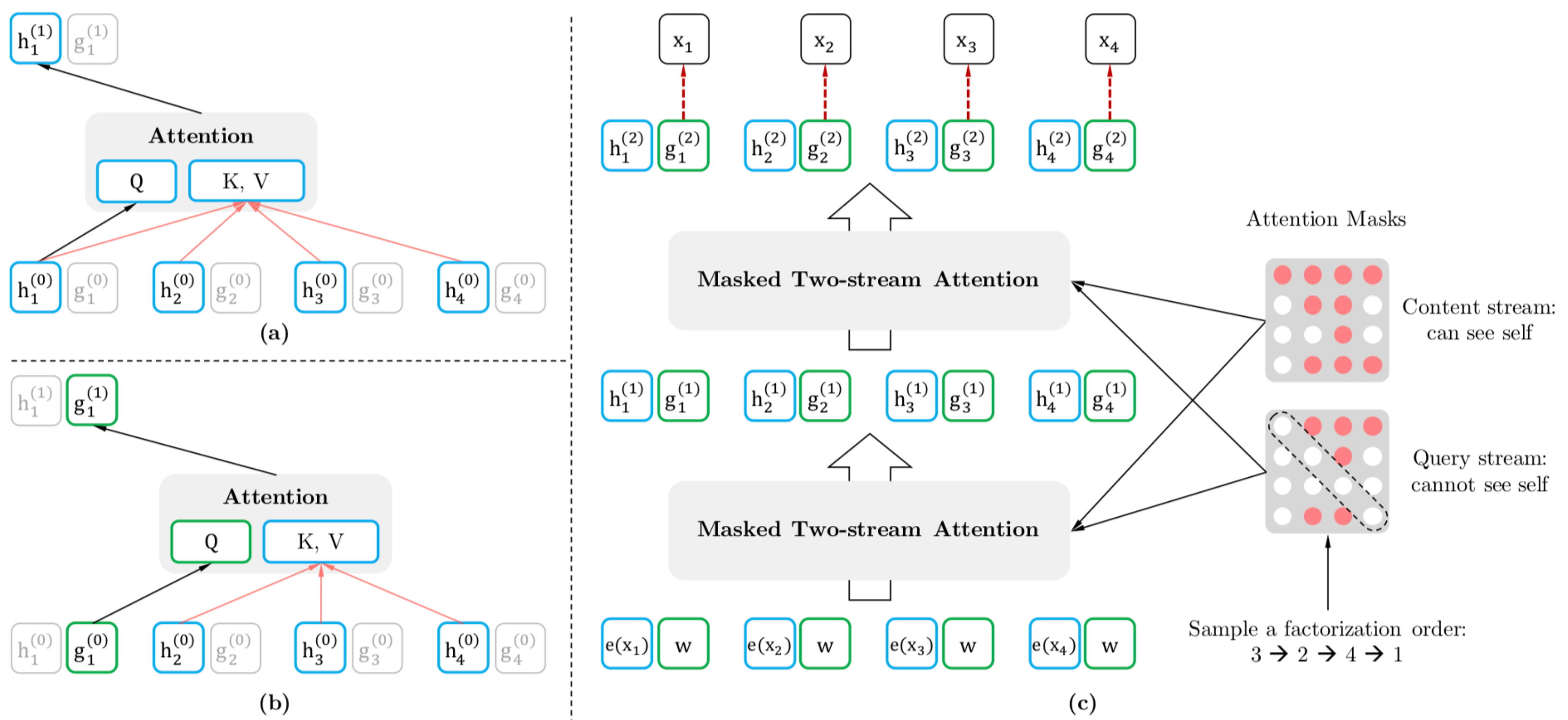
- **Model:** Same as BERT
- **Objective:** Sample words from language model, try to discriminate which words are sampled



- **Data:** Same as BERT, or XL-Net (next) for large models
- **Result:** Training much more efficient!

Permutation-based Auto-regressive Model + Long Context (XL-Net) (Yang et al. 2019)

- **Model:** Same as BERT, but include longer context
- **Objective:** Predict words in order, but different order every time



- **Data:** 39B tokens from Books, Wikipedia and Web

DeBERTa
(He et al. 2021)

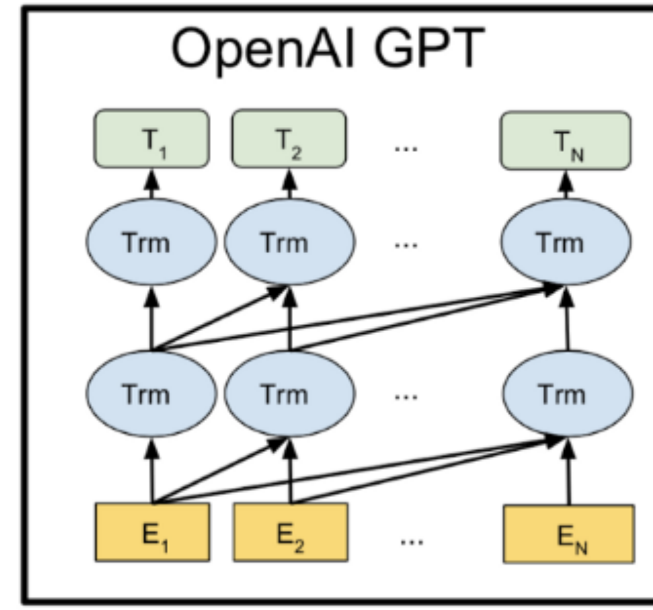
- **Model:** Transformer model with
  - "disentangled attention" treating relative position and content separately
  - absolute positional embeddings added at end of model
- **Objective:** Masked language modeling (w/ regularization by perturbing input embeddings)
- **Data:** 78GB Wikipedia, Reddit, and Subset of Common Crawl
- **Results:** SOTA on SuperGLUE

Compact Pre-trained Models

- Large models are expensive, can we make them smaller?
- **ALBERT (Lan et al. 2019):** Smaller embeddings, and parameter sharing across all layers
- **DistilBERT (Sanh et al. 2019):** Train a model to match the distribution of regular BERT

Auto-regressive LMs for Generation/Prompting

GPT-2



- **Model:** Left-to-right transformer (1.5B)
- **Training Objective:** Standard language modeling
- **Data:** WebText (millions of web pages)
- **Results:** Impressive results in generation of long-form text, and zero shot task completion
- Available open source, easy to use

GPT-3

- **Model:** Left-to-right transformer (175B)
- **Training Objective:** Standard language modeling
- **Data:** CommonCrawl (1T words)
- **Results:** Further impressive results in generation of long-form text, and zero shot task completion

PaLM

- **Model:** Left-to-right transformer (540B)
- **Training Objective:** Standard language modeling
- **Data:** Common crawl (1T words)
- **Results:** Further impressive results in generation of long-form text, and zero shot task completion

OPT/BLOOM

- Open source large language models (up to 175GB)
- **OPT:** https://github.com/facebookresearch/metaseq
    - (see also the experiment log!)
- **BLOOM:** https://huggingface.co/bigscience/bloom

Should we be *Pre*-training?

(Dery et al. 2021, Dery et al. 2022)

Impacts of Transfer Learning

- **Downstream performance:** Improved downstream task performance
- **Faster convergence:** Fewer epochs to reach same level of performance
- **Data-efficiency:** Fewer datapoint required to achieve good performance

Is Pre-train then Fine-tune always appropriate ?

Pros	**Cons**
• One model for all downstream tasks • Amortize compute burden • All the benefits of transfer learning	• Good pre-training performance does not imply good downstream perf • No free lunch - one pre-training objective cannot perform well across all end tasks • No clear way to cross-validate pre-training stage

Pre-training design choices

Lots of pre-training design choices

Objective	Data ($\mathcal{D}$)	Transform ($\mathcal{T}$)	Representation ($\mathcal{R}$)	Output ($\mathcal{O}$)
BERT	Out-of-domain	BERT-Op	Bidirectional	Denoise Token
TAPT	Task data	BERT-Op	Bidirectional	Denoise Token
DAPT	In-domain	BERT-Op	Bidirectional	Denoise Token
ELMO	Out-of-domain	No-Op	Left-to-Right and Right-to-Left	Next Token
GPT	Out-of-domain	No-Op	Left-To-Right	Next Token
XLNet	Out-of-domain	No-Op	Random factorized	Next Token
Electra	Neural LM Data	Replace	Bidirectional	Real / Synthetic
...	...	...	...	...

Letting the end-task choose

- We can generate many more objectives by taking this view
- Let the end-task choose which objectives are most useful

Data ($\mathcal{D}$)		Transform ($\mathcal{T}$)		Representation ($\mathcal{R}$)		Output ($\mathcal{O}$)
Out-of-domain		No-Op		Bidirectional		Next Token
In-domain		Replace		Left-to-Right		Real / Synth
Task data	$\times$	Mask	$\times$	Right-to-Left	$\times$	Denoise Token
Neural LM Data		Noising embeds		Rand. factorized		TF-IDF
...		...		...		...

$\downarrow$

TAPT = {Task data → BERT-Op → Bidirectional → Denoise Token}
GPT = {Out-of-domain → No-Op → Left-to-Right → Next Token}
New-Obj$_1$ = {Task data → BERT-Op → Left-to-Right → Denoise Token}
New-Obj$_2$ = {In-domain → No-Op → Random Factorized → TF-IDF}
...

Letting the end-task choose

- Choosing can be hard sampling or soft weighting

Table 2: Our framework and **AANG** on tasks **using only task data**. Without using any external data, we are able to get significant average performance improvement over baselines. Superscripts are p-values from paired t-tests (best multitask versus best single-task).

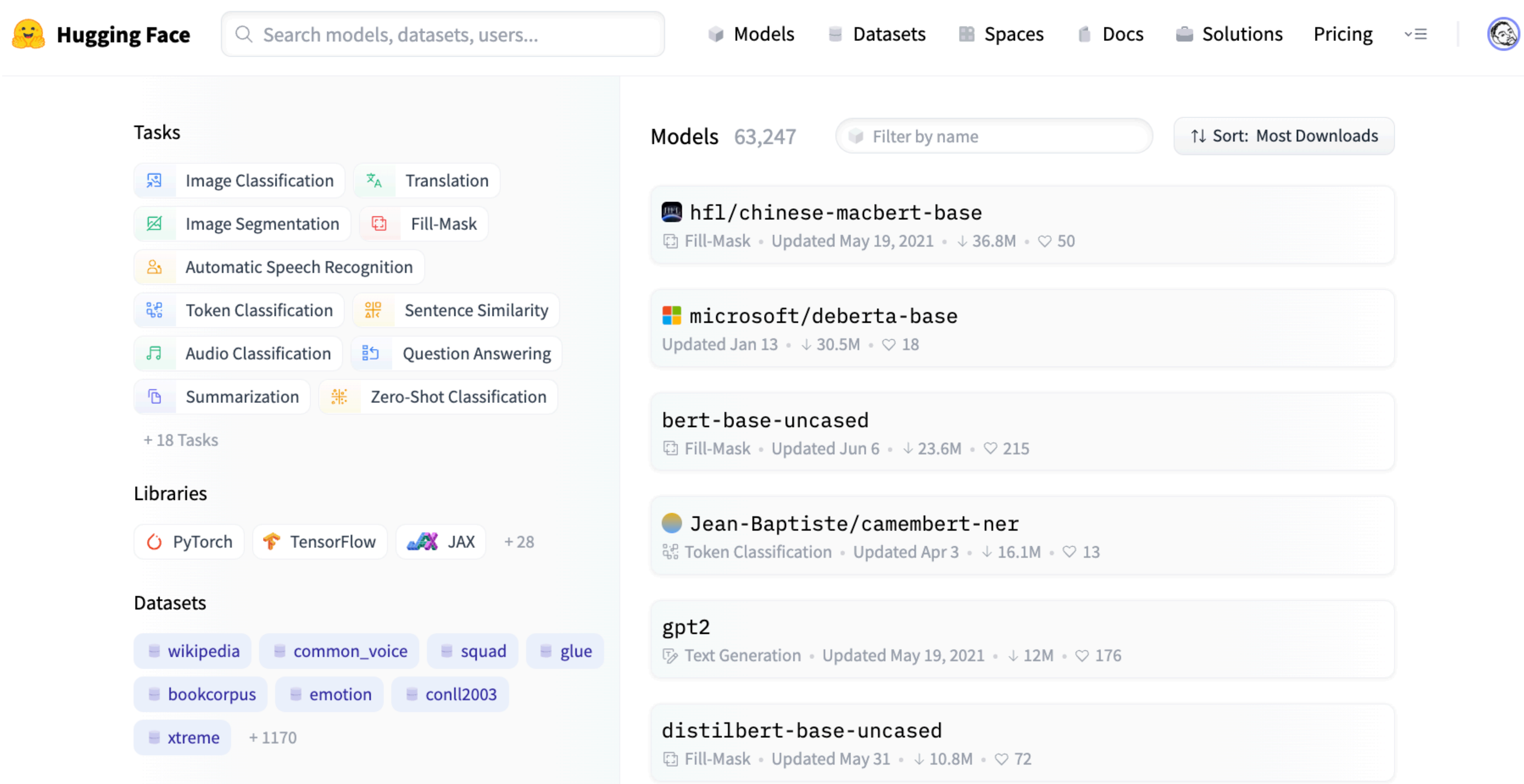
Task Aware	Method	#	CS		BIOMED	NEWS	STANCE	
			ACL-ARC	SCIERC	CHEMPROT	H.PARTISAN	SE-2016-6	AVG
No	RoBERTa	1	$66.03_{3.55}$	$77.96_{2.96}$	$82.10_{0.98}$	$93.39_{2.26}$	$70.37_{1.51}$	77.97
	TAPT (Gururangan et al., 2020)	1	$67.74_{3.68}$	$79.53_{1.93}$	$82.17_{0.65}$	$93.42_{2.87}$	$70.74_{1.21}$	78.72
	[OURS] Multitask-TD	24	$69.60_{3.80}$	$\mathbf{83.37}_{0.58}$	$83.42_{0.26}$	$97.95_{0.73}$	$71.02_{0.43}$	81.07
Yes	X. GPT-style	1	$67.22_{0.44}$	$81.62_{0.84}$	$83.29_{1.21}$	$96.41_{0.73}$	$70.67_{1.46}$	79.84
	Y. XLNET-style	1	$69.76_{2.42}$	$81.81_{0.42}$	$83.39_{0.31}$	$96.41_{1.92}$	$71.18_{0.58}$	80.51
	Z. BERT-style (Dery et al., 2021b)	1	$70.08_{4.70}$	$81.48_{0.82}$	$\mathbf{84.49}^{(0.09)}_{0.50}$	$96.84_{1.72}$	$72.70_{0.60}$	81.12
	[OURS] AANG-[X+Y+Z]	3	$71.51_{3.19}$	$82.89_{0.78}$	$83.68_{0.45}$	$96.92_{1.26}$	$\mathbf{72.75}^{(0.94)}_{0.82}$	81.55
	[OURS] AANG-TD	24	$\mathbf{73.26}^{(0.28)}_{1.32}$	$\mathbf{82.98}^{(0.27)}_{1.52}$	$83.91_{0.32}$	$\mathbf{98.46}^{(0.14)}_{0.0}$	$72.46_{1.65}$	**82.21**

Practicals of large pre-trained models

Huggingface Model Hub

- The hugging face model hub is one go-to source for models

https://huggingface.co/models



Practicals of using large pre-trained models

- Use smaller versions like DistilBert
  - Half the size and very little performance loss
- Gradient accumulation
  - Fitting large batches lead to OOMs - run several smaller batches and back-prop to gather gradients before optimizer step
- Selective finetuning
  - Top few layers -> layer-norm layers -> Every thing else

Scaling Laws
(Kaplan et al. 2020)

- Language models exhibit slow-fast-slow learning pattern
- Larger models improve in fewer steps

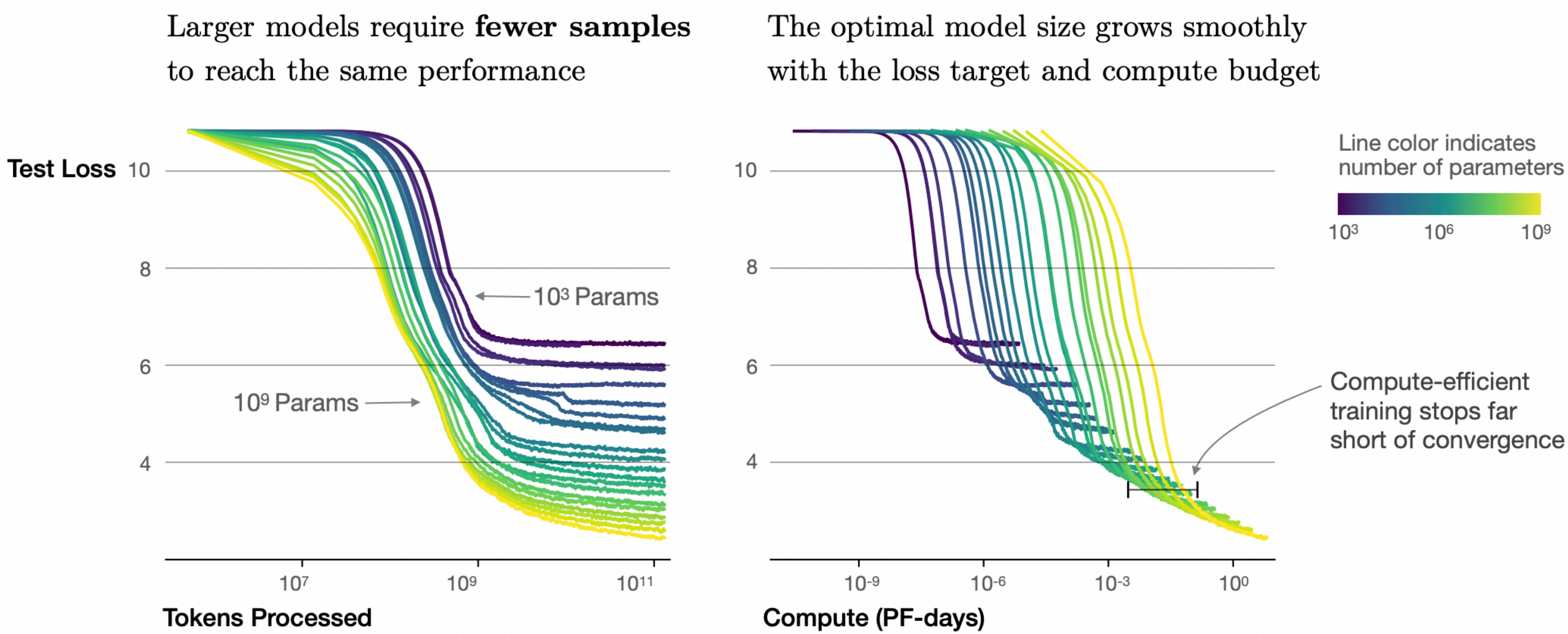


Figure 2 We show a series of language model training runs, with models ranging in size from 10^3 to 10^9 parameters (excluding embeddings).

Questions?