Pretrained Language Models

Jonathan May

September 19, 2023

Preamble

These notes are possibly the most likely of all notes you'll get in class to go out of date. As of 2023, when I'm writing this preamble, the latest and greatest models are coming out and making big splashes on a daily basis. It's not going to be possible to keep up. So we will focus on the general idea of pretrained LMs in the notes and talk about the latest ones in class.
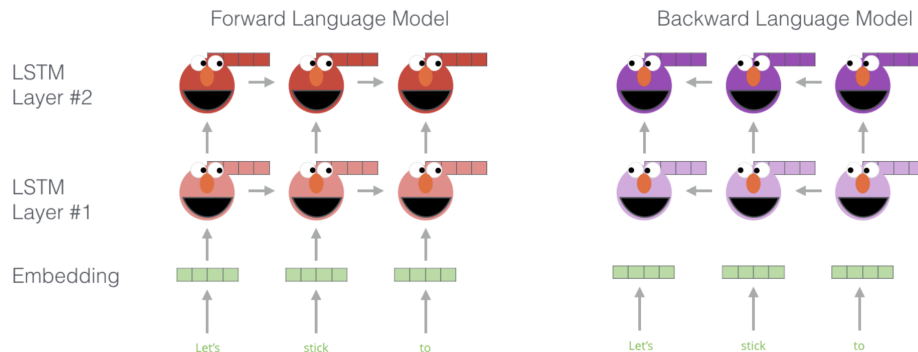
1 ELMo

In 2016, the predominant use of 'neural networks' in NLP was to insert type-based word embeddings like GLoVE or GenSim (implementation of Word2Vec) into existing models. ELMo (Embeddings from Language Models) from AI2 came out in 2018 and introduced what it pitched as better embeddings. It showed across-the-board improvement on a number of diverse NLP tasks and was, not surprisingly, the best paper at NAACL, given that everyone knew about it by the time the conference came around (it was first posted in October 2017). The claim was that this is a set of *contextualized* word embeddings. That is, instead of having one representation for *bank*, the word has a different representation depending on the context (i.e. sentence) it appears in.

How is this done? Well, first a contextual model of text is needed. That's easy, we've already seen several. This predates (sort of) Transformer, so ELMo used the predominant method at the time, bidirectional LSTMs.
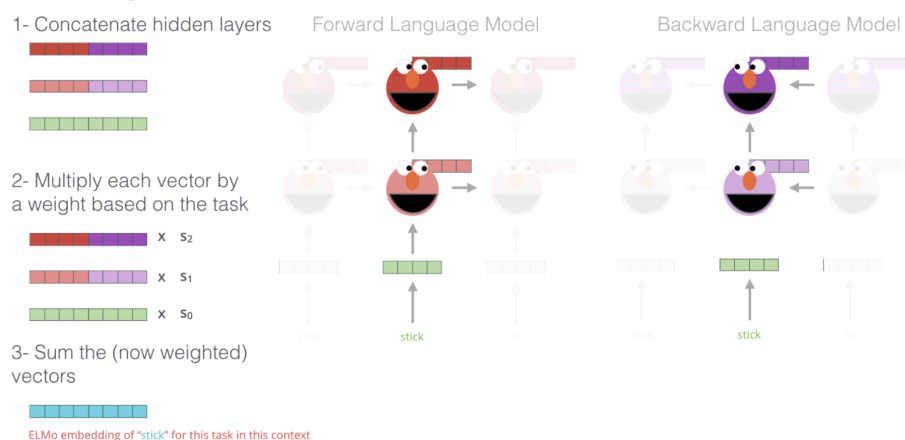
LSTMs are trained on plain text for the language modeling task, i.e. predict the next word (for the forward LSTM) or the previous word (for the backward LSTM). Here's an illustration from The Illustrated BERT: [1]

---

[1] http://jalammar.github.io/illustrated-bert/

Embedding of "stick" in "Let's stick to" - Step #1

Forward Language Model   Backward Language Model

LSTM
Layer #2

LSTM
Layer #1

Embedding

Let's   stick   to   Let's   stick   to

Embedding of "stick" in "Let's stick to" - Step #2

1- Concatenate hidden layers   Forward Language Model   Backward Language Model

2- Multiply each vector by
a weight based on the task

x s_2

x s_1

x s_0

Let's   stick   to   Let's   stick   to

3- Sum the (now weighted)
vectors

ELMo embedding of "stick" for this task in this context

This is trained on the Billion Word Benchmark [1] which is 1B English words from WMT 2011. That probably took a while but AI2 did it so you don't have to![2]

Now an embedding of a word in its context is obtained by running the context (i.e. the sentence) through the trained bi-LSTM and reading off the hidden state in both directions. Or, as it turns out, you can take some linear interpolation of hidden states at each layer; specifically how to linearly interpolate can be chosen by fine tuning interpolation parameters. However the core embeddings aren't fine tuned; they're just produced and used.

What was really cool about ELMo is you could use these embeddings in place of embeddings in your previously built models for various tasks and you pretty much got a gain. The most impressive results presented with the ELMo paper were across-the board lifts in the GLUE [13] tests by taking SOTA models and substituting in ELMo embeddings:

---

[2]Note: how are the words initially embedded? The paper is pretty murky about this! Best I can figure they are read in as a character CNN (but I won't get into the details about this; somewhat also murky details are in [5] – this is what happens when you don't use peer review...which became explicitly a feature by GPT-4.)

Task	Previous SOTA		Our baseline	ELMo + Baseline	Increase (Absolute/Relative)
SQuAD	SAN	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al (2017)	88.6	88.0	88.7 +/- 0.17	0.7 / 5.8%
SRL	He et al (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al (2017)	91.93 +/- 0.19	90.15	92.22 +/- 0.10	2.06 / 21%
Sentiment (5-class)	McCann et al (2017)	53.7	51.4	54.7 +/- 0.5	3.3 / 6.8%

Here's a bar graph (sam bowman slides)
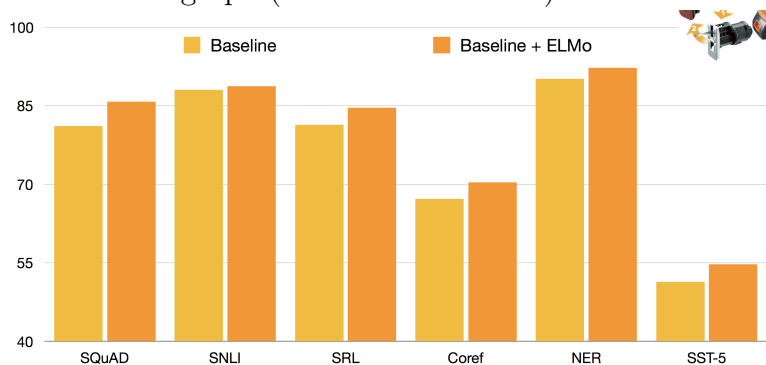
I should probably mention what these tasks are:

- SQuAD: question answering, extractive. Find the span.

- SNLI: natural language inference, aka 'entailment': given a pair of sentences (A, B), does B entail A, contradict A, or is it neutral to A? If A='Three men are standing in a field' and B='People are standing', B entails A. If B='People are sleeping', contradiction. If B='The field is covered in snow', neutral. Classify correctly.

- SRL: determine the semantic roles of text spans as they relate to verbs (e.g in 'Mary sold the book to John', Mary=agent, John=recipient, sold=predicate). Classification.

- Coref: Determine which mentions are of the same entity

- NER: find the spans and label with entity type

- Sentiment: classify sentence sentiment in a 5-way label

2   OpenAI GPT

Not to be out done, in June 2018, OpenAI improved upon ELMo in a paper that IMO didn't get too much attention [10], maybe because it wasn't even put on ArXiv AFAICT, let alone

submitted for publication. It had the following differences from ELMo:

- Transformer architecture instead of biLSTM. Along with that, using BPE.

- Designed directly for task prediction, with no other architecture, and carried with it a notion of *fine-tuning*; a task (e.g. multiple choice question answering) is turned into input sequences (e.g. question, separator token, answer choice). The topmost hidden unit after reading the last word is connected to a feed-forward classifier. Cross entropy on the classifier back-propagated.

- Trained on different data (Books corpus = 800M words)

GPT is essentially a Transformer *decoder* without source attention to an encoder. In other words, it uses masked self-attention that only looks to its left. If it didn't, the topology of Transformer means it could 'cheat':
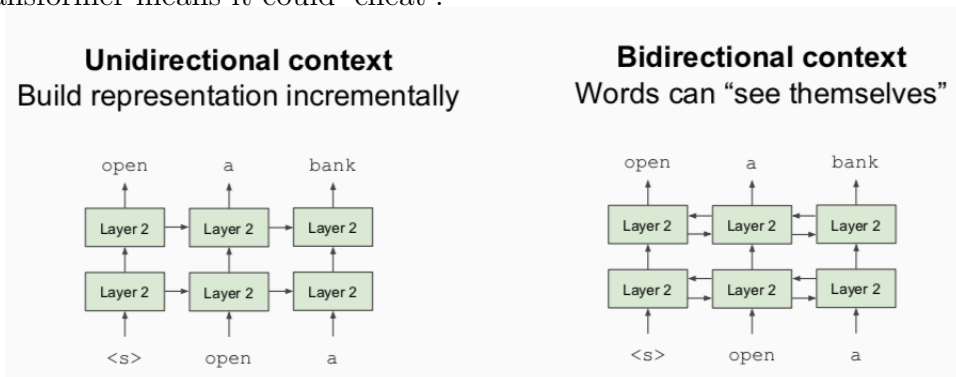
Here's an illustration of how task prediction works. You structure your input data as a series of sentences and then put a feed forward/linear layer on the end to map to classification.
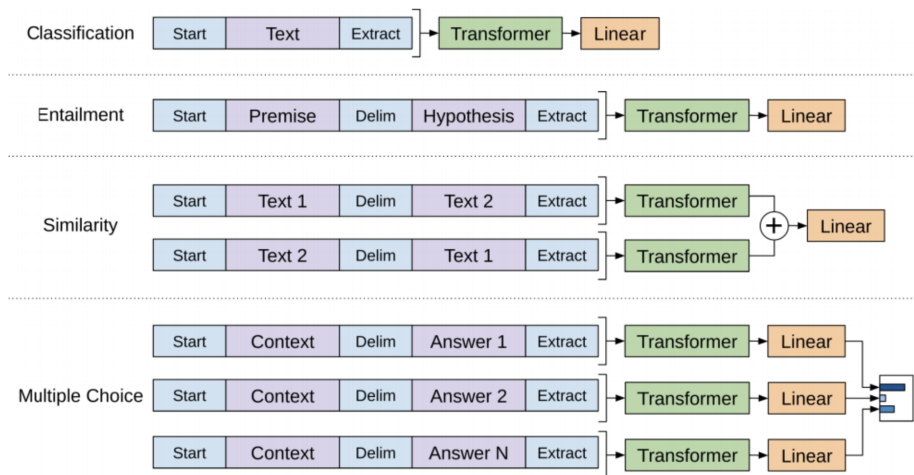
I didn't actually hear GPT until reading the BERT paper...maybe BERT had better marketing. The folks at OpenAI learned their lesson and resolved never to be ignored again.

3 BERT (images from Jacob Devlin slides)

ELMo had a few months of glory (and everyone(?) ignored GPT) until October 2018 when Google struck back with BERT (Bidirectional Encoder Representations from Transformer) [2], clearly riffing on the muppet theme.[3] Like GPT, BERT used Transformer and subwords (though it used google's slightly different WordPiece [14]). BERT also used the fine tuning paradigm. But there were more important differences:

- New objectives: Bidirectional prediction using word masking and next sentence prediction

- More structured two-sentence representation, class token for predictions included during training (first word of every input is the otherwise unused [CLS]).

- Pretraining+Fine Tuning recipe

- Trained on a lot more data (Wikipedia = 2.5B words + Books corpus = 800M words)

- There's a large version of BERT with tons (at the time) of parameters: for L=layers, H=hidden units, A=attention heads, BERT-BASE = (L=12, H=768, A=12, Total Parameters=110M) = same size as GPT; BERT-LARGE = (L=24, H=1024, A=16, Total Parameters=340M)

In ablation studies, the BERT authors claim the key is in the pretraining tasks: GPT and ELMo just pretrained on the language model objective (predict next word).
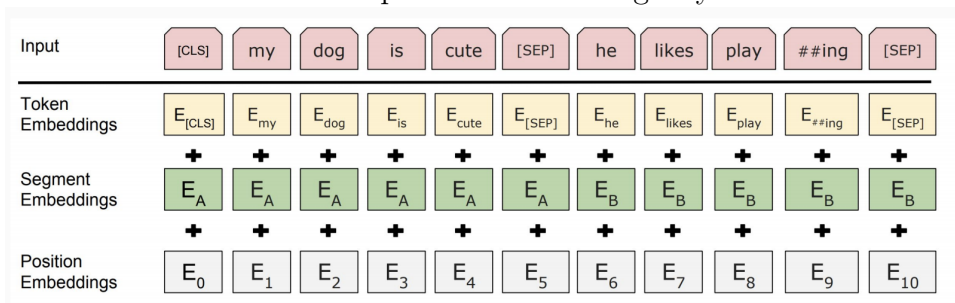
To pretrain, BERT masks out 15% of the words from its training data and then tries to predict them (15 seemed to be the magic number):

BERT also structures its input in the following way:

An encoding value is learned (same value on each position) for 'sentence 1' vs 'sentence 2' and added to each embedding. This is how data is then set up (see above). The [CLS] token is used instead of the last word token used in GPT.

---

[3]Yes, there were more muppet themed papers: GROVER (Generating aRticles by Only Viewing mEtadata Records.) [15], ERNIE (Enhanced language RepresentatioN with Informative Entities) [16] (I think there were two ERNIEs actually). There was something branded 'big bird' but it wasn't part of the paper name. The trend seems to have eased, thankfully.
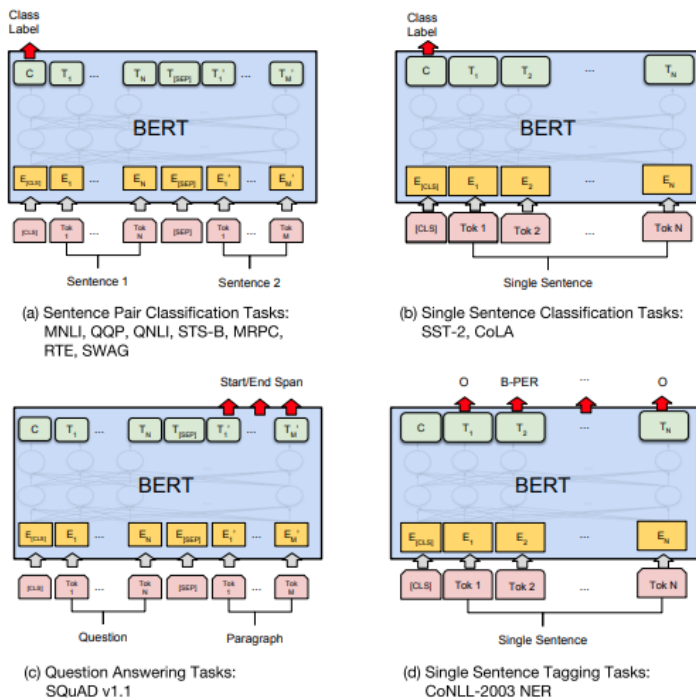
Two pretraining losses are calculated. For each MASK token, the top level hidden unit corresponding to each MASK predicts a word from the vocabulary (well, loss for probability of the correct word is calculated). Only sometimes (10%) a random word is used instead of [MASK] and sometimes (10%) the right word is used, but the 15% of words we need to predict in this pretraining are specified in the training corpus. Note that now self-attention can span the entire sentence.

Additionally, a next sentence prediction task is used: Either sentence 2 is the next sentence or it isn't, and this is learned by feeding the top hidden unit for [CLS] into a binary classifier.
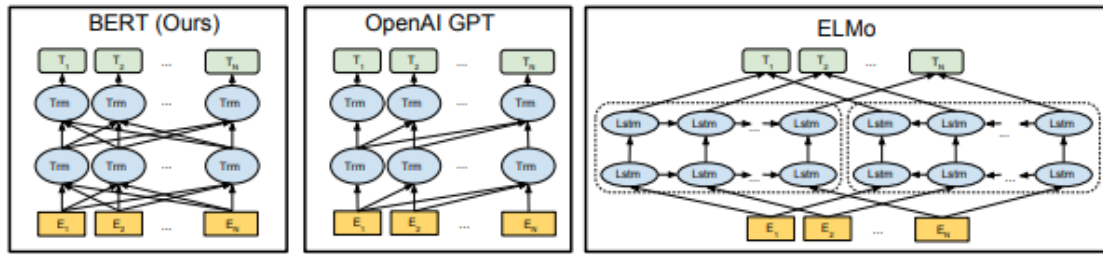
Apart from pre-training, BERT uses per-task fine-tuning. Here's a diagram from the BERT paper comparing the two:
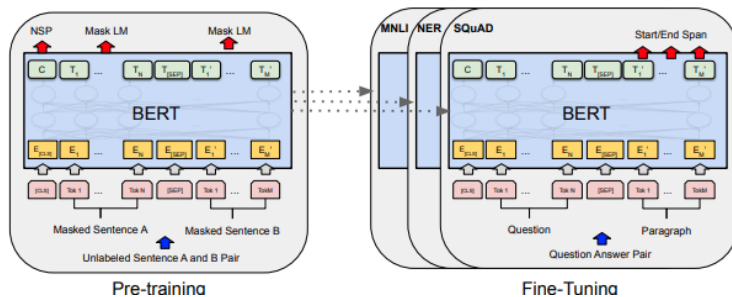
Fine tuning is the same idea as before, though BERT can be used both in classification and tagging paradigms (so could the other models, presumably). Here are the setups (from BERT paper):

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

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

(c) Question Answering Tasks: SQuAD v1.1

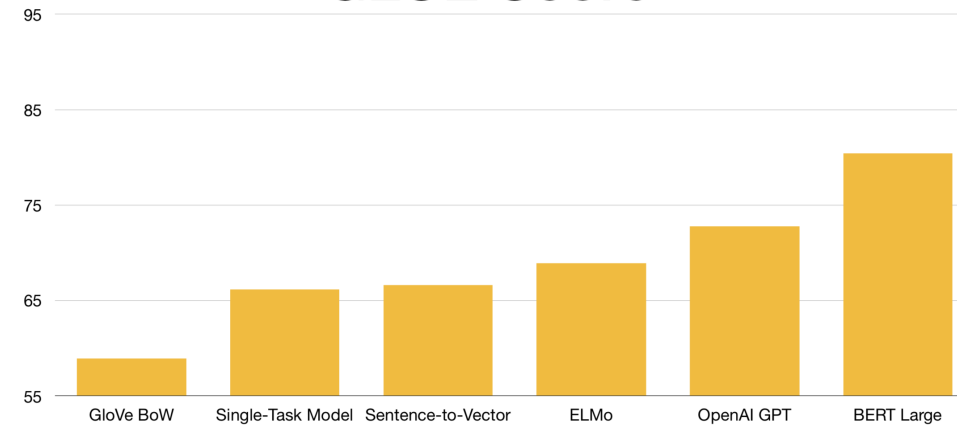(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Here's an overview comparing the topologies of ELMo, GPT, and BERT (from BERT paper):

Results were significant:

GLUE Score

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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT$_{BASE}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT$_{LARGE}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

The tasks:

- MNLI: like NLI but done over many genres, supposed to be less biased (we'll get into that)

- QQP: Quora question pairs: given two questions, are they asking the same thing?

- QNLI: SQUAD converted into a binary NLI task (does this sentence answer the question?)

- SST-2: Binary sentiment analysis

- CoLA: Given an english sentence, is it 'acceptable' to native ears ('Bill's book has a red cover.') or not ('The Bill's book has a red cover.')

- STS-B: Sentence pairs annotated with score from 1 to 5 on semantic similarity

- MRPC: Are two sentences semantically equivalent?

7

- RTE: Like MNLI but much less training data

An additional GLUE task, WNLI, is the Winograd challenge (Resolve 'it' in 'The trophy didn't fit in the suitcase because it was too big/small.') At BERT publication no model, including BERT, outperformed majority baseline (65.1). (This has since changed.)

A quick followup from Facebook, RoBERTa (Robustly Optimized BERT pretraining Approach) [9]:

- Even more data. Everything in BERT (Book Corpus and Wikipedia = 16GB uncompressed) plus Common Crawl news (76 GB after filtering) plus web text data linked to from reddit with 3+ upvotes (38 GB) plus a subset of common crawl filtered to look like winograd stories (31 GB).

- Unlike BERT, masking was done multiple times on sentences.

- Next Sentence Prediction as described in the BERT paper (but possibly not in the implementation) seems to hurt, so it was removed. Just masking is used. (So what happens to the CLS token training? Presumably only fine-tuning is used to make it meaningful but this is somewhat unclear to me.)

- Using the same training settings as BERT, and the same data, RoBERTa was better. When adding more data and training even longer it was even better.

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
BERT$_{LARGE}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet$_{LARGE}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

There have since been many more:

- DistilBERT [12]: Almost as good as BERT but a lot faster and smaller

- AlBERT [6]: A Lite BERT. same idea.

- BART [7] BERT but a sequence-to-sequence model, useful for generation and classification

- T5 [11] really big Transformer trained on Common Crawl filtered for English, then fine-tuned on a lot of tasks all at once

- Multi-language versions of these

- Domain-specific versions of these

Here's a refreshed (as of 2023) leaderboard for GLUE. Human level performance is currently #23 on the list and not shown.

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
1	Microsoft Alexander v-team	Turing ULR v6	↗	91.3	73.3	97.5	94.2/92.3	93.5/93.1	76.4/90.9	92.5	92.1	96.7	93.6	97.9	55.4
2	JDExplore d-team	Vega v1		91.3	73.8	97.9	94.5/92.6	93.5/93.1	76.7/91.1	92.1	91.9	96.7	92.4	97.9	51.4
3	Microsoft Alexander v-team	Turing NLR v5	↗	91.2	72.6	97.6	93.8/91.7	93.7/93.3	76.4/91.1	92.6	92.4	97.9	94.1	95.9	57.0
4	DIRL Team	DeBERTa + CLEVER		91.1	74.7	97.6	93.3/91.1	93.4/93.1	76.5/91.0	92.1	91.8	96.7	93.2	96.6	53.3
5	ERNIE Team - Baidu	ERNIE	↗	91.1	75.5	97.8	93.9/91.8	93.0/92.6	75.2/90.9	92.3	91.7	97.3	92.6	95.9	51.7

4 HuggingFace

A major aid to experimentation is HuggingFace (`https://huggingface.co/`) which has come to prominence by making these models and others not discussed here available as pretrained PyTorch with common user interfaces. They are relatively easy to use and it's easy to compare different models and see which works best on your task. At current writing (this section last edited in 2023) they are practically a *de facto* standard, though such standards change over time and lots of prior work doesn't use their architecture. Nevertheless, I recommend you check them out at `https://github.com/huggingface`.

5 Parameter Efficiency

The models we've so far discussed follow the paradigm that out of the box they don't do too much but when you expose them to some supervised data that is an exemplar of a task and fine-tune their parameters they can do the task when given more input data. One problem with this paradigm is that the base models are quite large, and then when fine-tuned you have another model that is as large as the base. If you have k tasks you have to store k copies of the fine-tuned base model. This is inefficient, so there have been efforts to allow the scaling to many tasks without exploding the number of models that have to be saved. This is an active area of research (as of this 2023 update), but here are a few interesting approaches to parameter efficiency.[4]

5.1 Multi-Task Learning

If you train a model to do more than one thing, then you implicitly are saving parameters.[5] T5 (referenced above) is one example of that approach. The pretrained model is fine-tuned on many tasks all at once, each prepended with an instruction relevant to that task. T5 has since been further fine-tuned on more downstream tasks. A limitation of this approach his that the instructions are 'hard', i.e. you can do the tasks you're trained on but it isn't clear you can do any other tasks.

---

[4]Some of what follows from here: `https://github.com/allenai/acl2022-zerofewshot-tutorial`

[5]Note that the term *multi-task learning* is also applied to a case where you use a single model to do two different kinds of prediction at the same time. I'm abusing the term here.
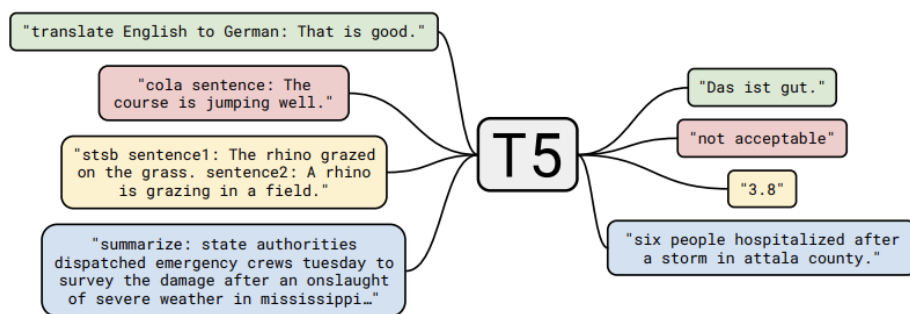
5.2 Continuous Prefix Tuning

The idea behind prefix tuning[6] and a number of other related works is that, rather than imagining or declaring what the task prefix will be as in T5, it might be better to learn the prefix as well. Thus, we can imagine several otherwise unused tokens being learned when a pretrained model is exposed to new task data; the rest of the model isn't modified! Then, instead of preceding your input with 'summarize' you precede it with '[TASK465]' or whatever that you've previously learned. You could do this for any number of tasks independently. There are a few versions of this; the one from [8] is shown below. The results on a few tasks are sometimes better than full fine-tuning but the goal is to reach parity despite only fine-tuning 0.1% of the parameters.
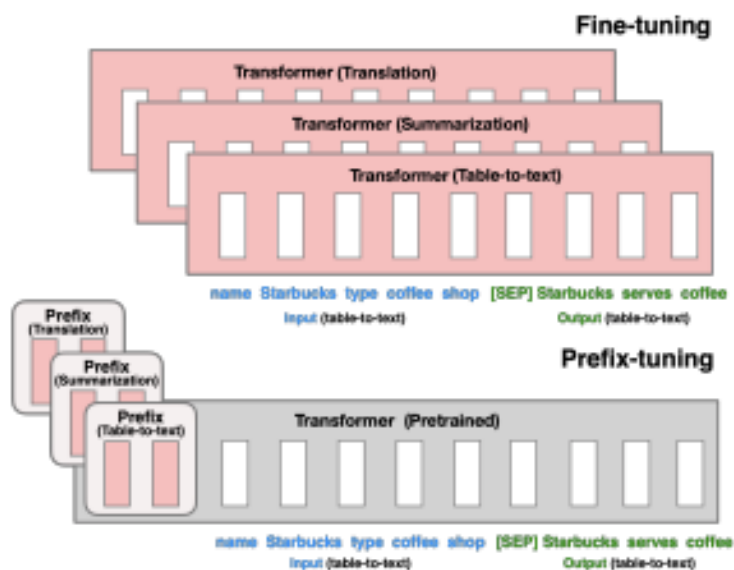

5.3 Adapters

Adapters [3] are kind of the same idea as prefix tuning but instead of adjoining to the left of the stack, they surround each layer of the transformer. The motivation for these came out slightly differently from those of prefixes; before adapters people were playing around with

---

[6]https://aclanthology.org/2021.acl-long.353/

only fine-tuning some of the layers in a PTLM, and adapters were shown to be more efficient than this approach. A slightly larger footprint was claimed in the original adapters paper (3.6%) than by prefixes, but this is of course a hyperparameter. Below is a figure from the paper showing how adapters are added in. A goal of adapters had been to make a 'plug and play' approach to model specialization by using off-the-shelf adapters built for one task and combining them together; I haven't heard too much about this lately, probably because of zero shot models changing the paradigm somewhat.
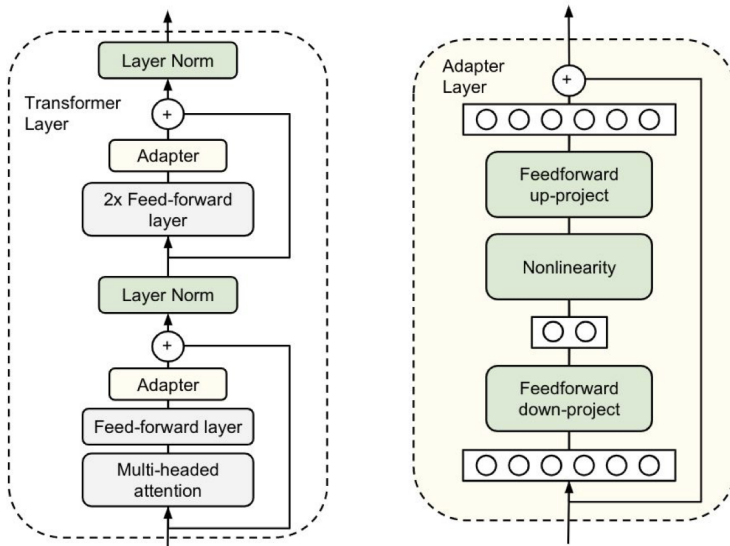
5.4   LoRA

Instead of adding parameters around the current stack why not add them *to* the stack? That is, given $d \times d$ weight matrix W (e.g. a Q, K, V matrix), let $W' = W + W_l$ and let $W_l = W_{l1} \times W_{l2}$ where W_{l1} is $d \times r$ and W_{l2} is $r \times d$, $r << d$. How much lower? Well it varies but values of 4 or 8 were used in the paper [4]. A 'shadow matrix' was learned for the W matrices (sometimes only a subset of them). As few as 0.02 parameters were learned (in the case of GPT-3) with very good results! Deep in the appendix of the paper the authors noted the combination of LoRA and prefix tuning worked very well!
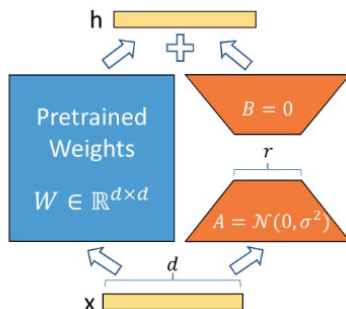
Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 (Adapter^H)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 (Adapter^H)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	91.6	53.4/29.2/45.1

6 Zero-Shot Models

6.1 GPT-2

OpenAI, presumably not happy with being overshadowed by BERT, but not really interested in branding any better, released the paper "Language Models are Unsupervised Multitask Learners"[7] that described GPT-2 in February 2019. It was trained on 8m documents, 40gb of text, sourced from outbound links from Reddit. The biggest model, at 1.5b parameters, exceeded SOTA performance considerably on a variety of LM sets even though it was not adapted to them. It was even able to do well (but not SOTA) on machine translation, question answering, and summarization tasks just by passing in natural language sequence prompts that attempted to elicit the kind of task response desired (e.g. inputting an article, then "TL;DR", then allowing the system to generate. The thing that really made the news was the (cherry-picked) 'unicorn' story. The other thing that made the news was OpenAI's refusal to let nearly anyone actually use the full system for the first year or so, claiming it was 'too dangerous.' So there was a lot of skepticism. But eventually the models were released and the claims were in fact justified.

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	60.12	**93.45**	**88.0**	**19.93**	40.31	0.97	1.02	22.05	44.575
1542M	**8.63**	63.24	**93.30**	**89.05**	**18.34**	35.76	0.93	0.98	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

---

[7]https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

> **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.
>
> Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.
>
> Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.
>
> While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic."
>
> Dr. Pérez believes that the unicorns may have originated in Argentina, where the animals were believed to be descendants of a lost race of people who lived there before the arrival of humans in those parts of South America.
>
> While their origins are still unclear, some believe that perhaps the creatures were created when a human and a unicorn met each other in a time before human civilization. According to Pérez, "In South America, such incidents seem to be quite common."
>
> However, Pérez also pointed out that it is likely that the only way of knowing for sure if unicorns are indeed the descendants of a lost alien race is through DNA. "But they seem to be able to communicate in English quite well, which I believe is a sign of evolution, or at least a change in social organization," said the scientist.

Table 13. Conditional generation on an out-of-distribution context by GPT-2. Cherry pick of 10 samples generated with $k = 40$.

It's worth pausing to describe the two (ish) approaches that are being proposed here and why they are now so cool. The main reason they are cool is that the model parameters aren't ever changed, so manipulation of behavior occurs only at inference time.

1. **In-Context Learning** or demonstration based learning. Several examples of desired input-output behavior are given, then one or several inputs without an output are given and the model produces outputs that follow the pattern.

2. **Instruction Learning**. A natural language description of what is wanted is produced, then an input is given, and the model produces outputs that are responsive to the instruction.

In practice, both are used together. There are also questions about the specific value of each component (see below).

6.2 GPT-3

In 2020 OpenAI released "Language Models are Few-Shot Learners"[8] which heralded the release of GPT-3 in a paper that was 72 pages long (GPT-2 paper was 24 pages long) and that described a model with 175b parameters. It was trained on 570gb of text (500b tokens) from common crawl, their previous data set, and some other 'high quality' data sets. When prompted with only a few examples (and sometimes with none at all) it outperformed SOTA on a wide variety of tasks, including common sense reasoning tasks, machine translation, question answering, and others. This time OpenAI set up an API and web interface so that researchers and others could use the models. The results have been really excellent though sometimes care is needed to prompt appropriately.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

6.3 Chat GPT and beyond

With somewhat less fanfare, in March 2022 OpenAI released 'Training language models to follow instructions with human feedback'[9] and replaced its GPT-3 models with these. Even the smallest (1.3b param) models were shown to be preferred to GPT-3 175b by users. They were fine-tuned to respond better to user prompts using 'Reinforcement Learning with Human Feedback' (which we will cover in a separate lecture). The authors also claim these models are less toxic than previous models.

In late November 2022, OpenAI released Chat GPT, a chatbot interface that wrapped this enhanced GPT-3 in a free-to-use and widely available interface. The promotion of and response to ChatGPT was very very loud. Within a few months seemingly everyone, even beyond the tech world, was aware of the technology, surprised at what it could do, and possibly scared of it. This kicked off an arms race among leading tech companies to build their own models (which started being called 'large language models' despite the term being fairly vapid).

6.4 Why/how does in-context learning work?

(From `https://github.com/allenai/acl2022-zerofewshot-tutorial`)
Some views (this is an area of active study, though):

---

[8]`https://arxiv.org/abs/2005.14165`
[9]`https://arxiv.org/abs/2203.02155`

- Demonstrations do not teach a new task; instead, they allow the 'locating' of an already-learned task during pretraining (Reynolds & McDonell, 2021)

- LMs do not exactly understand the meaning of their prompt (Webson & Pavlick, 2021)

- Demonstrations are about providing a latent concept so that LM generates coherent next tokens (Xie et al. 2022)

- In-context learning performance is highly correlated with term frequencies during pre-training (Razeghi et al. 2022)

- LMs do not need input-label mapping in demonstrations, instead, they use the specification of the input & label distribution separately (Min et al. 2022)

- Data properties lead to the emergence of few-shot learning (burstiness, long-tailedness, many-to-one or one-to-many mappings, a Zipfian distribution) (Chan et al. 2022)
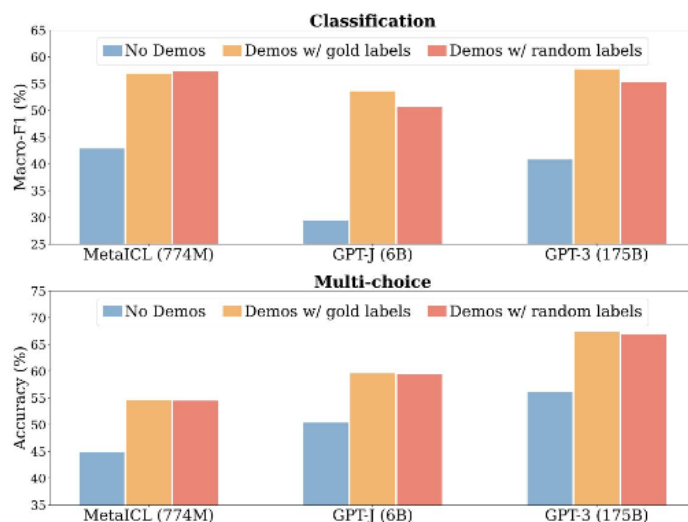
An interesting finding from all of this (from Min et al 22[10]) is that you can prompt with examples but *random* answers and get basically the same results.

---

[10]https://arxiv.org/abs/2202.12837 via the aforementioned tutorial

References

[1] Ciprian Chelba et al. *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling.* 2013. arXiv: 1312.3005 [cs.CL].

[2] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

[3] Neil Houlsby et al. "Parameter-Efficient Transfer Learning for NLP". In: *CoRR* abs/1902.00751 (2019). arXiv: 1902.00751. URL: http://arxiv.org/abs/1902.00751.

[4] Edward J. Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *CoRR* abs/2106.09685 (2021). arXiv: 2106.09685. URL: https://arxiv.org/abs/2106.09685.

[5] Rafal Jozefowicz et al. *Exploring the Limits of Language Modeling.* 2016. arXiv: 1602.02410 [cs.CL].

[6] Zhenzhong Lan et al. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: *CoRR* abs/1909.11942 (2019). arXiv: 1909.11942. URL: http://arxiv.org/abs/1909.11942.

[7] Mike Lewis et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.* 2019. arXiv: 1910.13461 [cs.CL].

[8] Xiang Lisa Li and Percy Liang. "Prefix-Tuning: Optimizing Continuous Prompts for Generation". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers).* Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597. DOI: 10.18653/v1/2021.acl-long.353. URL: https://aclanthology.org/2021.acl-long.353.

[9] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach.* 2019. arXiv: 1907.11692 [cs.CL].

[10] Alec Radford. "Improving Language Understanding by Generative Pre-Training". In: 2018.

[11] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *CoRR* abs/1910.10683 (2019). arXiv: 1910.10683. URL: http://arxiv.org/abs/1910.10683.

[12] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *CoRR* abs/1910.01108 (2019). arXiv: 1910.01108. URL: http://arxiv.org/abs/1910.01108.

[13] Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (2018). DOI: `10.18653/v1/w18-5446`. URL: `http://dx.doi.org/10.18653/v1/w18-5446`.

[14] Yonghui Wu et al. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: `1609.08144 [cs.CL]`.

[15] Rowan Zellers et al. *Defending Against Neural Fake News*. 2019. arXiv: `1905.12616 [cs.CL]`.

[16] Zhengyan Zhang et al. "ERNIE: Enhanced Language Representation with Informative Entities". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019). DOI: `10.18653/v1/p19-1139`. URL: `http://dx.doi.org/10.18653/v1/p19-1139`.