Pretrained Language Models

Jonathan May

September 24, 2024

Preamble

These notes and the notes for parameter efficient/instruction models are possibly the most likely of all notes you'll get in class to go out of date. As of 2024, when I'm editing this preamble, the latest and greatest models are coming out and making big splashes on a daily basis, though possibly not quite as much as in 2023. It's not going to be possible to keep up. So, we will focus on the general idea of pre-trained LMs in the notes and talk about the latest ones in class.

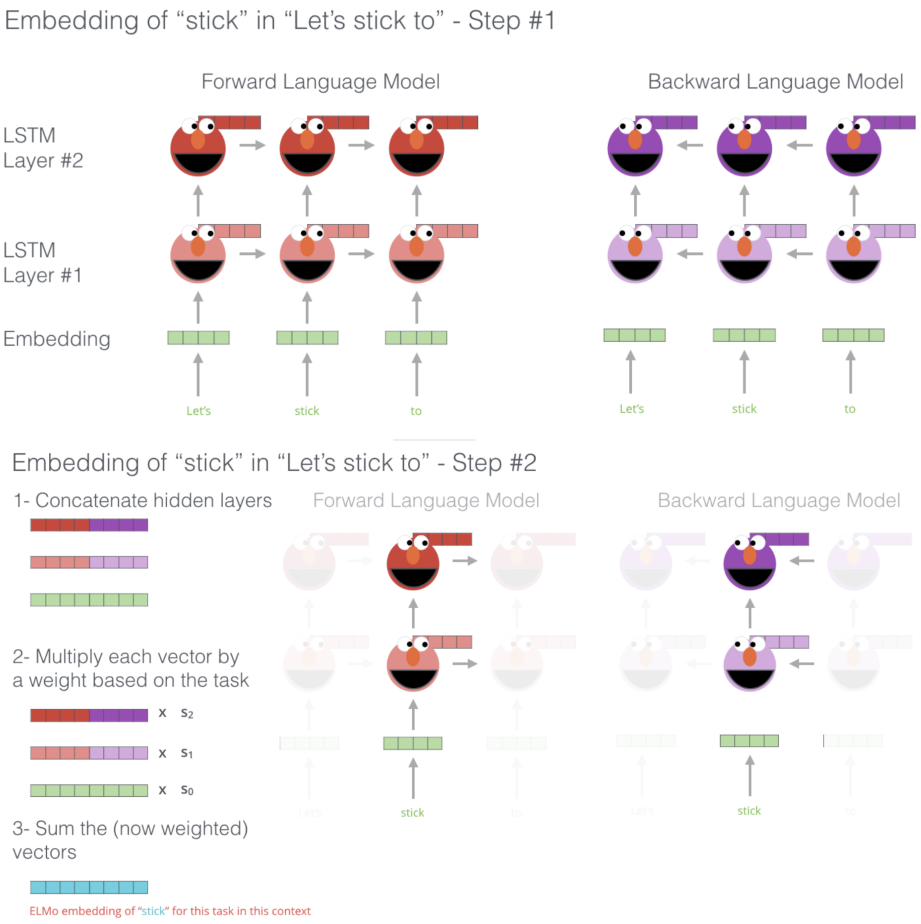
1 ELMo – contextualized word vectors

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/`



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]

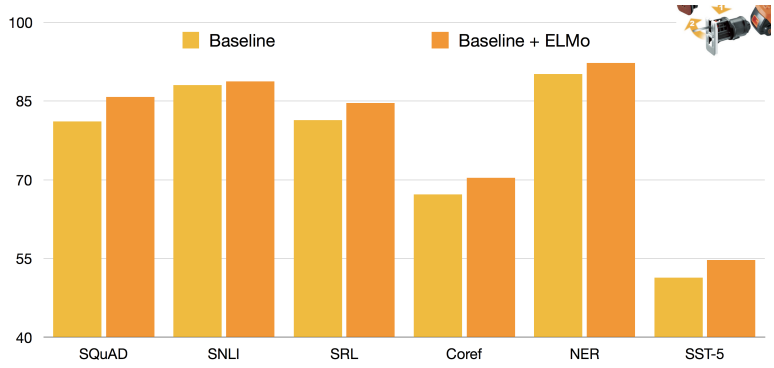
Now, an embedding of a word in its context is obtained by running the context (i.e. the sentence) through the trained bi-LSTM and reading off the hidden state in both directions. Or, as it turns out, you can take some linear interpolation of hidden states at each layer; specifically, how to linearly 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 [11] 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 is they are read in as a character CNN (but I won’t get into the details about this; somewhat also murky details are in [4] – 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 the entity type.
- Sentiment: Classify sentence sentiment in a 5-way label.

2 Fine-Tuning

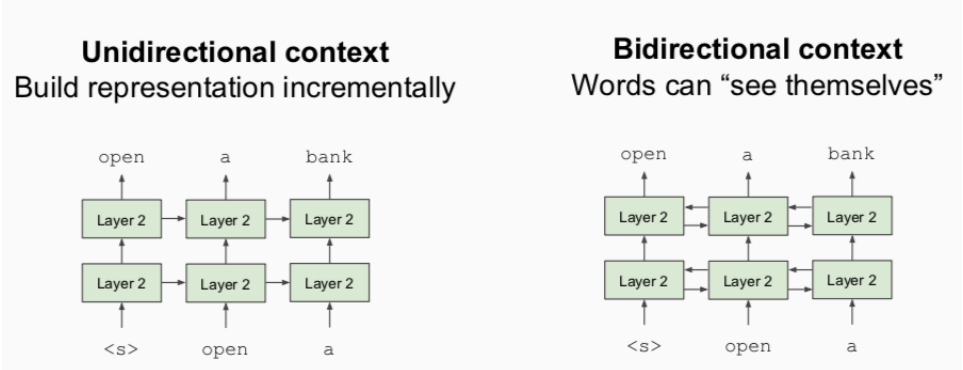
Folks at Google [2] and, about simultaneously, work in our lab [15] had the idea that you could *re-use* neural trained models that were originally trained for one purpose and then altered, or *fine-tuned* on a modification of that task or related task. In our work, we trained a translation model one one language pair with a lot of resources and then continued training it on a new language pair, re-purposing word embeddings, which seemed like it wouldn't work, but did. In the Google work, there was similar use of large-resource tasks fine-tuned on lower-resource related tasks but models were also *pre-trained* with general purpose language modeling tasks, and this was also shown to be helpful.

3 OpenAI GPT

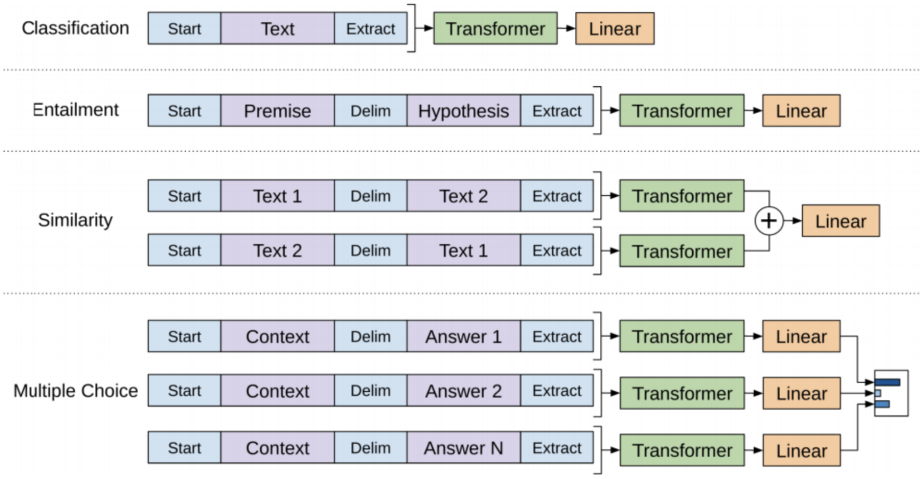
In June 2018, OpenAI improved upon ELMo in a paper that IMO didn't get too much attention [8], maybe because it wasn't even put on ArXiv AFAICT, let alone submitted for publication. It had the following differences from ELMo:

- Transformer architecture instead of biLSTM. Along with that, using BPE. Autoregressive (decoder-only) and used final word hidden state connected to classifier.
- Designed directly for task prediction, with no other architecture, and carried with it, as mentioned above, the 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. Loss includes the task **and** language modeling (detail).
- 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 the 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.

4 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) [3], clearly riffing on the muppet theme.[3] Like GPT, BERT used Transformer and subwords (though it used Google's slightly different WordPiece [12]). 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]).
- Pre-training+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)

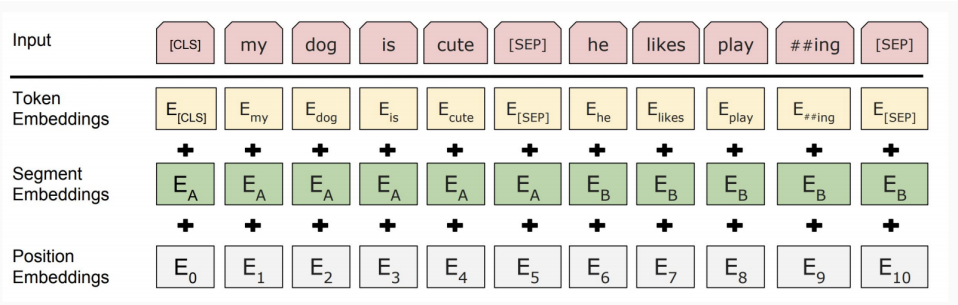
[3]Yes, there were more muppet themed papers: GROVER (Generating aRticles by Only Viewing mEtadata Records.) [13], ERNIE (Enhanced language RepresentatioN with Informative Entities) [14] (I think there were two ERNIEs actually). There was something branded 'big bird' but it wasn't part of the paper name. The trend seems to have eased, thankfully.

In ablation studies, the BERT authors claim the key is in the pretraining tasks: GPT and ELMo just pre-trained on the language model objective (predict the 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):

```
                          store          gallon
                            ↑              ↑
the man went to the [MASK] to buy a [MASK] of milk
```

BERT also structures its input in the following way:

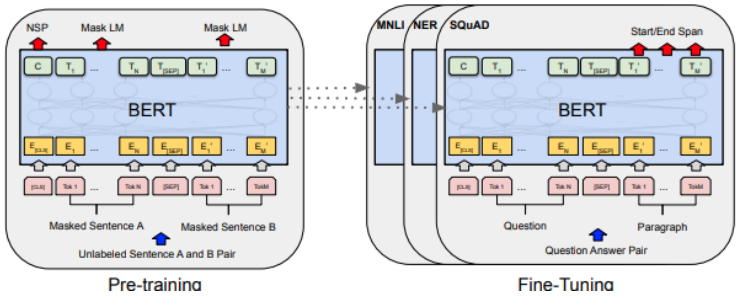


An encoding value is learned (same value on each position) for 'sentence 1' vs 'sentence 2' and added to each embedding. This is how data is then set up (see above). The [CLS] token is used instead of the last word token used in GPT.

Two pre-training 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 pre-training are specified in the training corpus. Note that now self-attention can span the entire sentence.

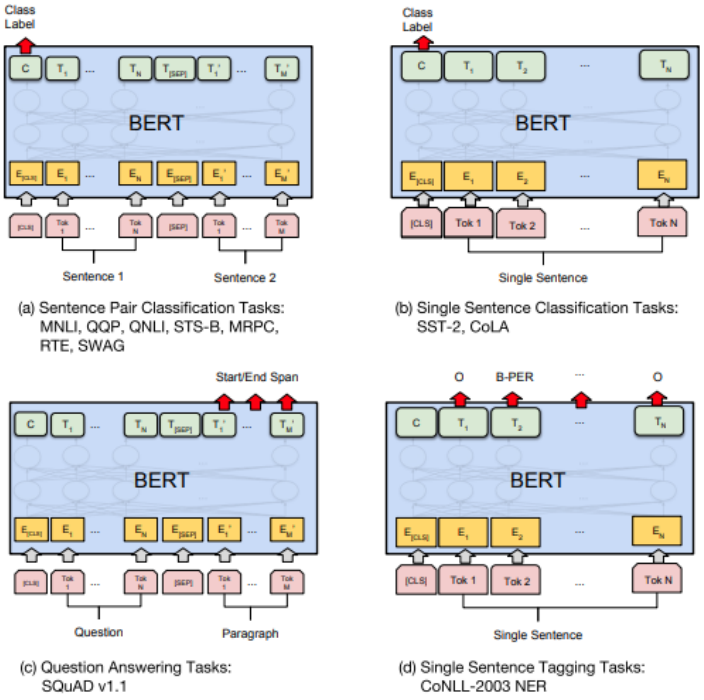
Additionally, a next sentence prediction task is used: Either sentence 2 is the next sentence or it isn't, and this is learned by feeding the top hidden unit for [CLS] into a binary classifier.

Apart from pre-training, BERT uses per-task fine-tuning. Here's a diagram from the BERT paper comparing the two:

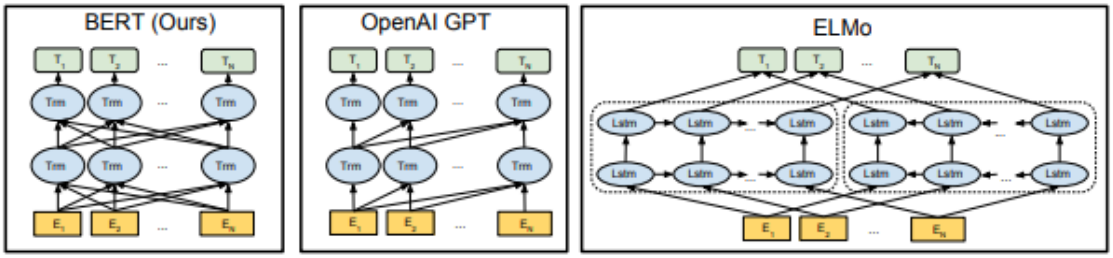


Fine-tuning is the same idea as before, though BERT can be used both in classification and tagging paradigms (so could the other models, presumably). Here are the setups (from

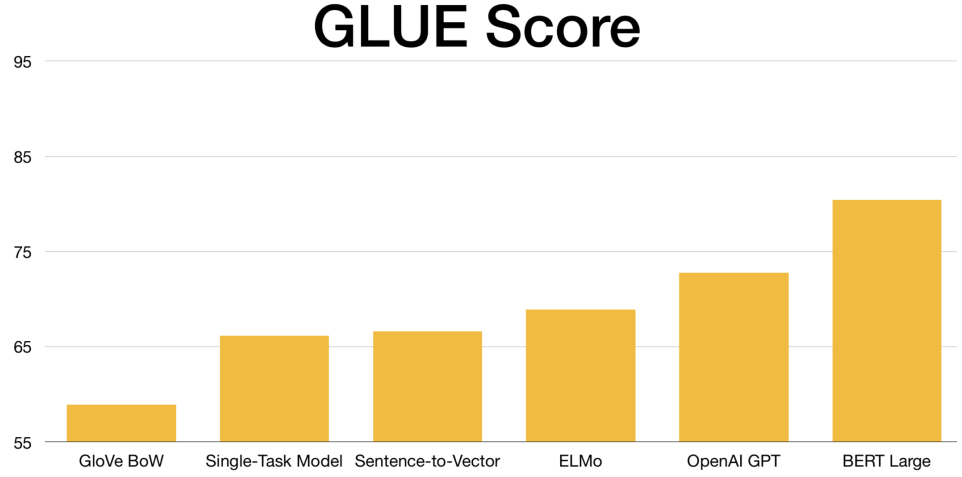
BERT paper):



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



Results were significant:



System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**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 a score from 1 to 5 on semantic similarity
- MRPC: Are two sentences semantically equivalent?
- 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 the majority baseline (65.1). (This has since changed.)

A quick followup from Facebook, RoBERTa (Robustly Optimized BERT pretraining Approach) [7]:

- 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										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	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 [10]: Almost as good as BERT but a lot faster and smaller
- AlBERT [5]: A Lite BERT. Same idea.
- BART [6]: BERT but a sequence-to-sequence model, useful for generation and classification
- T5 [9]: 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; no change in 2024 but there are small changes in SuperGLUE) 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

5 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`.

6 Brief Tutorial of fine-tuning with HuggingFace (see provided notebook as well)

(Subject to change and probably more than one way to do it!)

I will show a notebook. You may wish to look at documentation at `https://huggingface.co/docs/transformers/v4.44.2/en/main_classes/trainer`

References

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

[2] Andrew M Dai and Quoc V Le. "Semi-supervised Sequence Learning". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/7137debd45ae4d0ab9aa953017286b20-Paper.pdf.

[3] 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.

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

[5] 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.

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

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

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

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

[15] Barret Zoph et al. "Transfer Learning for Low-Resource Neural Machine Translation". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Ed. by Jian Su, Kevin Duh, and Xavier Carreras. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1568–1575. DOI: 10.18653/v1/D16-1163. URL: https://aclanthology.org/D16-1163.