CS6101 / WING.NUS Reading Group - Scribe Notes from Weeks 06 and Recess - Domain Adaptation / Adapting Pretrained Models. This document is public

Domain Adaptation / Adapting Pretrained Models Lecture

Lecturers: Abhinav Ramesh Kashyap, Devamanyu Hazarika, Alexandre Gravier, Rabiul Awal
Authors: Yisong Miao, Zhang Ruixi, Ding Xu, Xu Lin
Slide Link:
https://docs.google.com/presentation/d/1JQWRXKjh-nlu1vU4CKEJKOBbOhZ8JTYYuPH0iSN2hvE/edit?usp=sharing
<ps: still not complete>
This document:
https://docs.google.com/document/d/1mqw2kWiW2diG913IFip3EO22Nu_TfyFgxM_m_caDoxg
Other documents in this series at http://bit.ly/cs6101-2010-notes

Week 6 Materials

<Domain Adversarial Training of Neural Networks>

Reference: Ganin, Yaroslav, et al. "Domain-adversarial training of neural networks." *The Journal of Machine Learning Research* 17.1 (2016): 2096-2030.

Two kinds of domain adaptation: Unsupervised (with no labelled data in target domain) and semi-supervised.

Abhinav's Talk on: how can we probabilistically define a domain?

1. Quantify the difference between domain;
2. Quantify the bad error in target domain;
3. Theory basis for the representations we need to learn.

Domain defined in the probabilistic way: a distribution of a/some random variables

Learning a discriminative classifier or other predictor in the presence of a *shift* between training and test distributions is known as **domain adaptation (DA)**

Definition:

A domain $\mathscr{D}$ *is a pair* $(\mathscr{X}, f)$

Input variable: X **f:** project X to [0,1]. We have *source domain* $\mathcal{D}_{\mathrm{S}}$ and the *target domain* $\mathcal{D}_{\mathrm{T}}$

Both source domain and target domain are from the same X .

$$h : \underbrace{\mathcal{X} \rightarrow}_{\text{Binary Classifier (Holds for multi-class as well)}} [0,1]$$

$$\epsilon_S(h,f) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_S} [\underbrace{|h(x) - f(x)|}_{\text{Error in classification for x}}]$$

$$\epsilon_T(h,f) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_T} [|h(x) - f(x)|]$$

Bound between the target error and source error:
Target error < source error + **distribution difference** + difference in labelling function

Assuming difference in labelling function goes to zeros:

$$min\{ \mathop{\mathbb{E}}_{x\sim\mathcal{D}_S} [|f_S(x) - f_T(x)|], \mathop{\mathbb{E}}_{x\sim\mathcal{D}_T} [|f_S(x) - f_T(x)|]\}$$

Here, the **distribution difference** is the focus and could be represented with

- **L1-divergence**

$$d(\mathcal{D}_S, \mathcal{D}_T) = 2 \mathop{sup}_{B\in\mathcal{B}} |P_{D_S}(B) - P_{D_T}(B)|$$

$$\mathcal{B} \text{ All subsets of } \mathcal{D}_S, \mathcal{D}_T$$

  Problems:
  Hard to calculate over all possible events with finite samples for arbitrary distributions.

- **H-divergence**

$$d_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) = 2 \sup_{h\in\mathcal{H}} |\mathbb{E}_{x\sim\mathcal{D}_s}(h(x) = 1) - \mathbb{E}_{x\sim\mathcal{D}_T}(h(x) = 1)|$$

  An empirical H-divergence given symmetric hypothesis: (Ben-David et al. 2006, 2010)

$$\begin{aligned} d_{\mathcal{H}}(\mathcal{D}_S, \mathcal{D}_T) &\simeq 2 \sup_{h\in\mathcal{H}} \left| \frac{1}{|D_S|} \sum_{x\in D_S} \llbracket h(x) = 1 \rrbracket - \frac{1}{|D_T|} \sum_{x\in D_T} \llbracket h(x) = 1 \rrbracket \right| \\ &= 2 \sup_{h\in\mathcal{H}} \left| \frac{1}{|D_S|} \sum_{x\in D_S} 1 - \llbracket h(x) = 0 \rrbracket - \frac{1}{|D_T|} \sum_{x\in D_T} \llbracket h(x) = 1 \rrbracket \right| \\ &= 2 - 2 \min_{h\in\mathcal{H}} \left| \frac{1}{|D_S|} \sum_{x\in D_S} \llbracket h(x) = 0 \rrbracket + \frac{1}{|D_T|} \sum_{x\in D_T} \llbracket h(x) = 1 \rrbracket \right| \quad (2) \end{aligned}$$

In Ben-David et al. 2006, it suggest that if $\hat{d}_{\mathcal{H}}(S,T)$ is generally hard to compute exactly, we could approximate it by running a learning algorithm and learn the discrimination between source and target example. The generalization error between source and target error are ϵ . We could have an approximation to H-divergence as

$$\hat{d}_{\mathcal{A}} \;=\; 2\,(1-2\epsilon)\,.$$

Also named as Proxy A-distance (PAD)

Summary:
H-divergence relies on the capacity of the hypothesis class H to distinguish between examples generated by source domain from examples generated by target domain. It can be estimated from finite samples. The sample estimated will be close to actual value with very high probs.

- **Proxy A distance (Operationalises theory)**

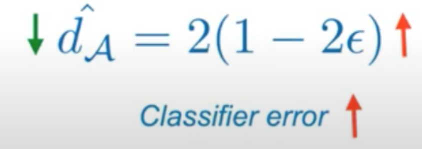
$$d_{\mathcal{A}}(\mathcal{D}_{\mathrm{S}}^{X}, \mathcal{D}_{\mathrm{T}}^{X}) = 2 \sup_{A \in \mathcal{A}} \left| \Pr_{\mathcal{D}_{\mathrm{S}}^{X}}(A) - \Pr_{\mathcal{D}_{\mathrm{T}}^{X}}(A) \right|$$

By choosing $\mathcal{A} = \{A_\eta | \eta \in \mathcal{H}\},$ subset of X, the PAD and H-divergence are identical.

To summarize, the target error could be represented with

$$\mathcal{R}_{D_T}(h) \leq \mathcal{R}_{D_S}(h) + d_{\mathcal{H}}(D_S, D_T) + f\left(\mathrm{VC}(\mathcal{H}), \frac{1}{n}\right) \qquad \text{(upper-bound)}$$

The VC term is the Vapnik-Chervonenkis dimensions and n is the number of samples.



$$\downarrow \hat{d}_{\mathcal{A}} = 2(1-2\epsilon) \uparrow$$

Classifier error ↑

Let classifier not able to distinguish between samples. Such classifier could be trained from Neural network.

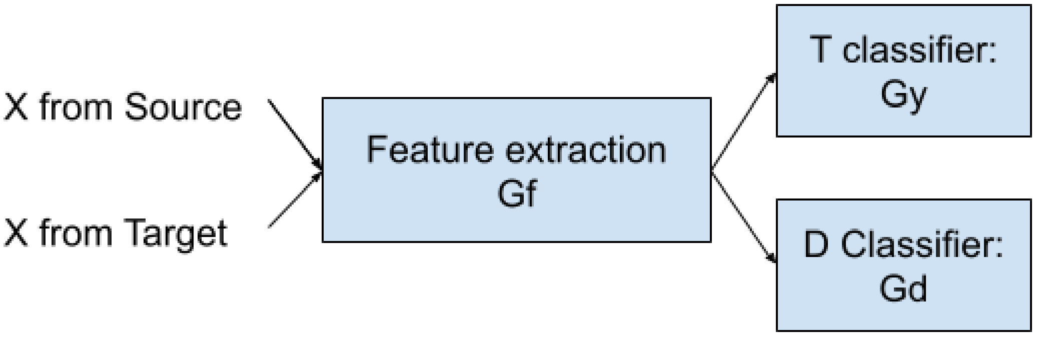
The rest of the paper directly stems from **this intuition:** in order to minimize the target risk the proposed **Domain Adversarial Neural Network (DANN)** aims to build an “internal representation that contains no discriminative information about the origin of the input (source or target), while preserving a low risk on the source (labeled) examples”.

Devamanyu's talks on how can we train models in source domain that works well in a related target domain.

- Indistinguishable between source and target domains.
- Discriminative towards source-domain task.

DANN has two versions: shallow version and deep version.

Shallow DANN:



We want to have DC fail while TC succeeds.

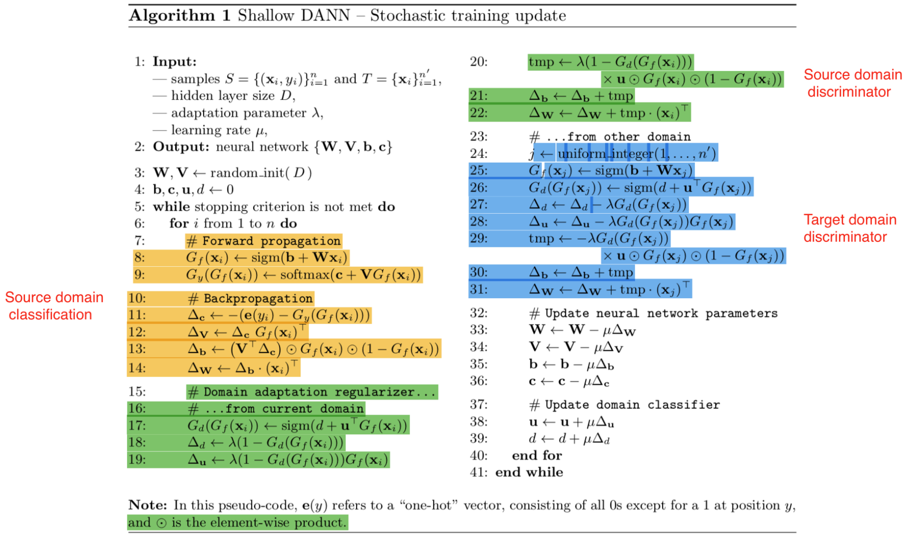
Optimization function:

$$E(\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c},\mathbf{u},z) \tag{9}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}) - \lambda\Big(\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z) + \frac{1}{n'}\sum_{i=n+1}^{N}\mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{u},z)\Big),$$

$$(\hat{\mathbf{W}},\hat{\mathbf{V}},\hat{\mathbf{b}},\hat{\mathbf{c}}) = \operatorname*{argmin}_{\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c}} E(\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c},\hat{\mathbf{u}},\hat{z}),$$

$$(\hat{\mathbf{u}},\hat{z}) = \operatorname*{argmax}_{\mathbf{u},z} E(\hat{\mathbf{W}},\hat{\mathbf{V}},\hat{\mathbf{b}},\hat{\mathbf{c}},\mathbf{u},z).$$



Algorithm 1 Shallow DANN – Stochastic training update

```
Input:
  — samples $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and $T = \{\mathbf{x}_i\}_{i=1}^{n'}$,
  — hidden layer size $D$,
  — adaptation parameter $\lambda$,
  — learning rate $\mu$,
Output: neural network $\{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$

$\mathbf{W}, \mathbf{V} \leftarrow$ random_init( $D$ )
$\mathbf{b}, \mathbf{c}, \mathbf{u}, d \leftarrow 0$
while stopping criterion is not met do
    for $i$ from 1 to $n$ do
        # Forward propagation
        $G_f(\mathbf{x}_i) \leftarrow \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x}_i)$
        $G_y(G_f(\mathbf{x}_i)) \leftarrow \mathrm{softmax}(\mathbf{c} + \mathbf{V}G_f(\mathbf{x}_i))$

        # Backpropagation
        $\Delta_\mathbf{c} \leftarrow -(\mathbf{e}(y_i) - G_y(G_f(\mathbf{x}_i)))$
        $\Delta_\mathbf{V} \leftarrow \Delta_\mathbf{c}\ G_f(\mathbf{x}_i)^\top$
        $\Delta_\mathbf{b} \leftarrow (\mathbf{V}^\top \Delta_\mathbf{c}) \odot G_f(\mathbf{x}_i) \odot (1 - G_f(\mathbf{x}_i))$
        $\Delta_\mathbf{W} \leftarrow \Delta_\mathbf{b} \cdot (\mathbf{x}_i)^\top$

        # Domain adaptation regularizer...
        # ...from current domain
        $G_d(G_f(\mathbf{x}_i)) \leftarrow \mathrm{sigm}(d + \mathbf{u}^\top G_f(\mathbf{x}_i))$
        $\Delta_d \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))$
        $\Delta_\mathbf{u} \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))G_f(\mathbf{x}_i)$
        $\mathrm{tmp} \leftarrow \lambda(1 - G_d(G_f(\mathbf{x}_i)))$
                $\times\ \mathbf{u} \odot G_f(\mathbf{x}_i) \odot (1 - G_f(\mathbf{x}_i))$
        $\Delta_\mathbf{b} \leftarrow \Delta_\mathbf{b} + \mathrm{tmp}$
        $\Delta_\mathbf{W} \leftarrow \Delta_\mathbf{W} + \mathrm{tmp} \cdot (\mathbf{x}_i)^\top$

        # ...from other domain
        $j \leftarrow \mathrm{uniform\_integer}(1, \ldots, n')$
        $G_f(\mathbf{x}_j) \leftarrow \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x}_j)$
        $G_d(G_f(\mathbf{x}_j)) \leftarrow \mathrm{sigm}(d + \mathbf{u}^\top G_f(\mathbf{x}_j))$
        $\Delta_d \leftarrow \Delta_d - \lambda G_d(G_f(\mathbf{x}_j))$
        $\Delta_\mathbf{u} \leftarrow \Delta_\mathbf{u} - \lambda G_d(G_f(\mathbf{x}_j))G_f(\mathbf{x}_j)$
        $\mathrm{tmp} \leftarrow -\lambda G_d(G_f(\mathbf{x}_j))$
                $\times\ \mathbf{u} \odot G_f(\mathbf{x}_j) \odot (1 - G_f(\mathbf{x}_j))$
        $\Delta_\mathbf{b} \leftarrow \Delta_\mathbf{b} + \mathrm{tmp}$
        $\Delta_\mathbf{W} \leftarrow \Delta_\mathbf{W} + \mathrm{tmp} \cdot (\mathbf{x}_j)^\top$

        # Update neural network parameters
        $\mathbf{W} \leftarrow \mathbf{W} - \mu\Delta_\mathbf{W}$
        $\mathbf{V} \leftarrow \mathbf{V} - \mu\Delta_\mathbf{V}$
        $\mathbf{b} \leftarrow \mathbf{b} - \mu\Delta_\mathbf{b}$
        $\mathbf{c} \leftarrow \mathbf{c} - \mu\Delta_\mathbf{c}$

        # Update domain classifier
        $\mathbf{u} \leftarrow \mathbf{u} + \mu\Delta_\mathbf{u}$
        $d \leftarrow d + \mu\Delta_d$
    end for
end while
```

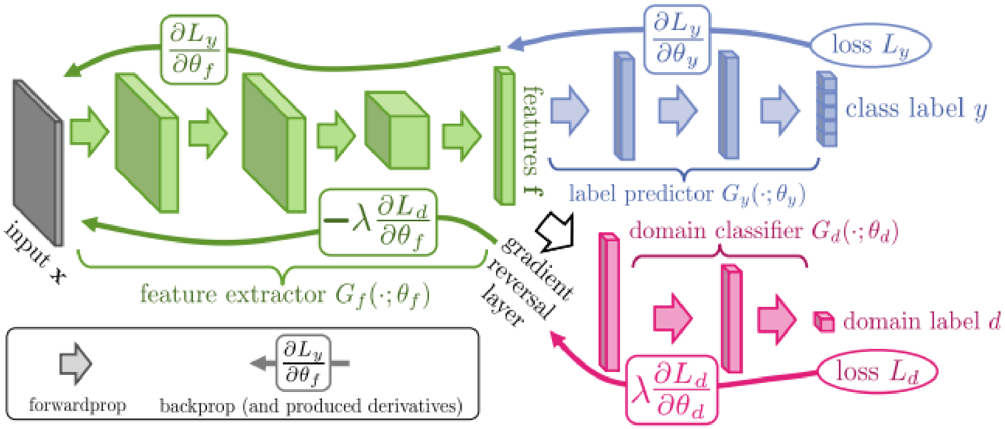
Source domain classification

Source domain discriminator

Target domain discriminator

Note: In this pseudo-code, $\mathbf{e}(y)$ refers to a "one-hot" vector, consisting of all 0s except for a 1 at position y, and $\odot$ is the element-wise product.

Deep DANN:


$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features f
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
domain classifier $G_d(\cdot;\theta_d)$
input x
gradient reversal layer
feature extractor $G_f(\cdot;\theta_f)$
domain label d
loss L_d
$\lambda \frac{\partial L_d}{\partial \theta_d}$
forwardprop
$\frac{\partial L_y}{\partial \theta_f}$
backprop (and produced derivatives)

Change the feature extractor gradient with **-1 with gradient reversal layer.**

Discussion:

1. Should the discriminator have output as half-half 0 and 1 instead of 100% 0 to 1 or 100% 1 to 0?
   The concept of swifting and still able to discrime. In the following work https://people.eecs.berkeley.edu/~jhoffman/papers/Tzeng_ICCV2015.pdf. This work has the domain discriminator enforced to make predictions towards a uniform distribution (to "maximally confuse" it).
2. Other supplementary:
   https://arxiv.org/pdf/1702.05464.pdf.
   This paper provides a good overview of adversarial training methods for domain adaptation.
   Improvement over DANN: Bousmalis, Konstantinos, et al. "Domain separation networks." *Advances in neural information processing systems*. 2016.

Put notes on the topics presented in the lecture here. You may also want to c-n-p notes from other sources (but make sure to attribute them by linking to the original resource).

<Deep Transfer RL for text summarization>

Reference: Keneshloo, Yaser & Ramakrishnan, Naren & Reddy, Chandan. (2018). Deep Transfer Reinforcement Learning for Text Summarization.

Romain first introduced the basic concept of ROUGE.
Two main approach for text summarization: (1) extractive (2) abstractive

Romain then goes through current state-of-the-art of text summarization:

- Basic Seq2Seq model (refer to past week of our reading group [Hyperlink here] Lecture Slides Scribe Notes)
- Pointer Network -- Prof Min comments that it is very related to Copy Network. (switch mechanism… Go check the slides :P)
- Pointer-generator model

Alexandre is online to cover transfer learning.

- Knowledge distillation
- Generalized models. This is not the favorite of Alexandre since he is unclear what is transferred between tasks (individual one can be difficult)
- Copy (some of) the layers

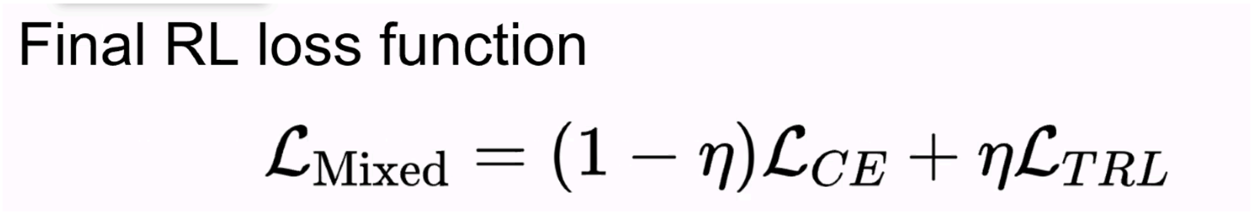
$$\mathcal{L}_{CE} = -\sum_{t=1}^{T} \log p_\theta^*(y_t \mid e(y_{t-1}), s_t, c_{t-1}, \mathbf{X})$$

The similar cross-entropy loss as in seq2seq.
Two problems:

- Exposure bias
- Inflexibility (meaning that the reference is fixed, however such reference might not be the ONLY ground truth. It affects the generalization of the model)

$$\mathcal{L}_{RL} = -\mathbb{E}_{y'_1,\cdots,y'_T \sim p_\theta^*(y'_1,\cdots,y'_T)}\left[r\left(y'_1,\cdots,y'_T\right)\right]$$

"the objective of the reinforcement learning problem here is to maximize the expected ROUGE score when we sample the summary from the proposed hybrid (pointer-generator / seq2seq) network parametrized by Θ (where Θ = all weights and biases in the model)"



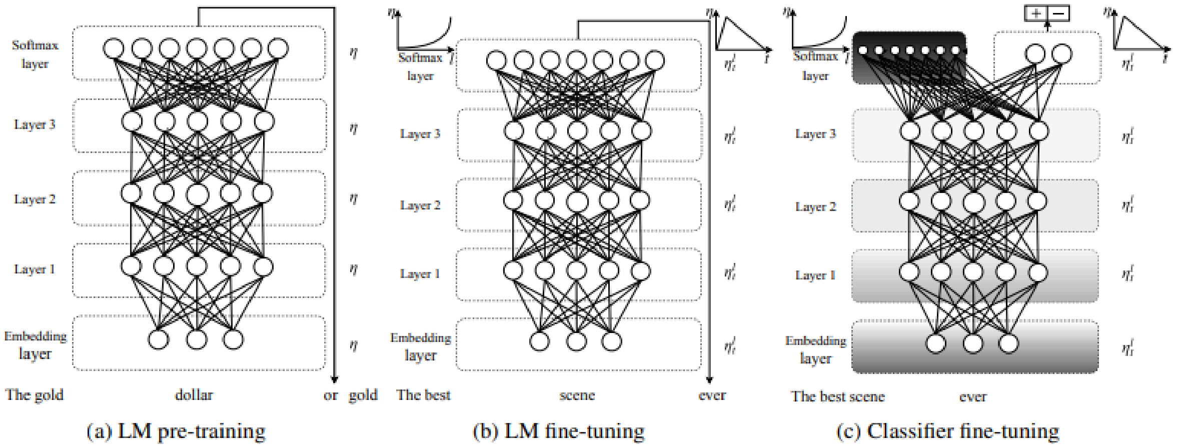
Final RL loss function

$$\mathcal{L}_{\text{Mixed}} = (1-\eta)\mathcal{L}_{CE} + \eta\mathcal{L}_{TRL}$$

Presenters' Question about the paper:

<Universal Language Model Fine-tuning for Text Classification>

- Domain adaptation vs Transfer learning
- ULMFiT is a universal model for transfer learning in any NLP task.
- Review three ways for retaining knowledge and avoiding catastrophic forgetting during fine-tuning:
  - Discriminative fine-tuning,
  - Slanted triangular learning rates
  - Gradual unfreezing
- Framework of the method proposed in this paper ULMFiT
  - General-domain LM pretrained on wikitext-103;
  - Target task LM fine-tuning ,further finetune on task dataset;
    - Discriminative finetuning, finetune each layer with different learning rates. Last layer x, then decrease by a factor of 2.6 for each layer up
    - Slanted triangular learning rates, short increase of initial learning rate, and then long decrease
  - Target task classifier Fine Tuning, gradual unfreeze the model from the last layer.

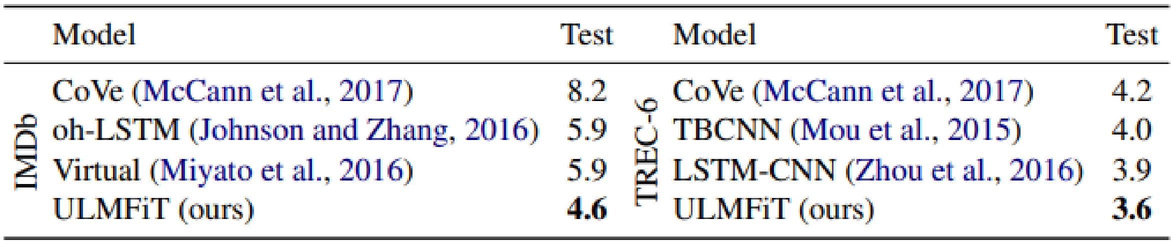


(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

Figure 1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning (‘*Discr*’) and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, ‘*Discr*’, and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

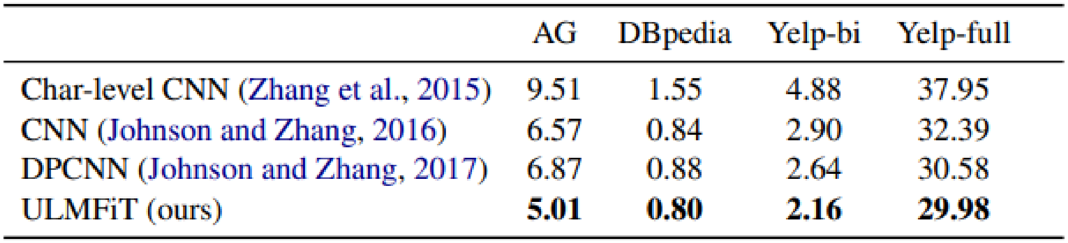
- Experiments

  Experiments shows superiority of this method



	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

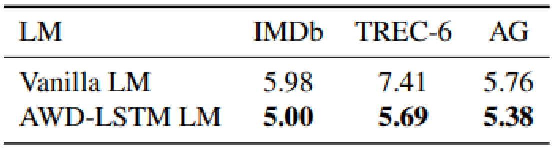
Table 2: Test error rates (%) on two text classification datasets used by McCann et al. (2017).



	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

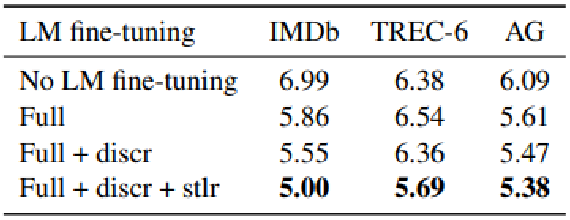
Table 3: Test error rates (%) on text classification datasets used by Johnson and Zhang (2017).

Ablation study shows that each step for transfering can contribute to the last performance.



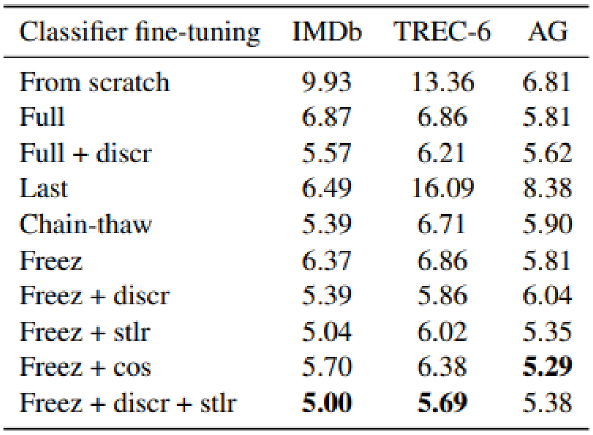
LM	IMDb	TREC-6	AG
Vanilla LM	5.98	7.41	5.76
AWD-LSTM LM	**5.00**	**5.69**	**5.38**

Table 5: Validation error rates for ULMFiT with a vanilla LM and the AWD-LSTM LM.



LM fine-tuning	IMDb	TREC-6	AG
No LM fine-tuning	6.99	6.38	6.09
Full	5.86	6.54	5.61
Full + discr	5.55	6.36	5.47
Full + discr + stlr	**5.00**	**5.69**	**5.38**

Table 6: Validation error rates for ULMFiT with different variations of LM fine-tuning.



Classifier fine-tuning	IMDb	TREC-6	AG
From scratch	9.93	13.36	6.81
Full	6.87	6.86	5.81
Full + discr	5.57	6.21	5.62
Last	6.49	16.09	8.38
Chain-thaw	5.39	6.71	5.90
Freez	6.37	6.86	5.81
Freez + discr	5.39	5.86	6.04
Freez + stlr	5.04	6.02	5.35
Freez + cos	5.70	6.38	**5.29**
Freez + discr + stlr	**5.00**	**5.69**	5.38

Table 7: Validation error rates for ULMFiT with different methods to fine-tune the classifier.

The lecturer said: The most important part for this model is LM-finetuning part, better transfer from large pretrained model to a specific task.

Zoom talk:
From Khushaal JAMMU to Everyone: (2:30 PM)
we'll talk more about this approach in the context of BERT next week! 😋
From Siddhartha Banerjee to Everyone: (2:36 PM)

There are opinions for/against ROUGE being a good metric for summarization. Are there any papers on RL for summarisation that use a different reward function (not with ROUGE)?
From Kan Min-Yen to Everyone: (2:38 PM)
I don't recall seeing other other papers using other metrics aside from ROUGE.Mostly the other metrics require other more cumbersome forms of annotation (e.g., Pyramid), they are harder to utilise as compared the lowest common denominator of ROUGE. Good question though.
From Devamanyu Hazarika to Everyone: (2:41 PM)
https://arxiv.org/pdf/2007.12626.pdf a recent paper discussing summarization evaluation. Might hold some clues.
From Siddhartha Banerjee to Everyone: (2:42 PM)
Xiachong Feng posted a paper on the general chat that uses Distributional semantics. The paper is from Prof. William Wang's group. I remember I had a chat with Prof Wang last year at SSNLP on something related :)Thanks Devamanyu.
From Romain Iehl to Everyone: (3:03 PM)
we can hear, thank you for the presentation

Slack talk
https://people.eecs.berkeley.edu/~jhoffman/papers/Tzeng_ICCV2015.pdf
- Paper where the domain discriminator is enforced to make predictions towards a uniform distribution (to "maximally confuse" it).
https://arxiv.org/pdf/1702.05464.pdf
- Paper providing good overview of adversarial training methods for domain adaptation.
From Devamanyu Hazarika to Everyone 2:24 PM

https://arxiv.org/abs/1909.00141

Deep Reinforcement Learning with Distributional Semantic Rewards...
Deep reinforcement learning (RL) has been a commonly-used strategy for the abstractive summarization task to address both the exposure bias and non-differentiable task issues.
From Xiachong Feng to Siddhartha Banerjee 2:39 PM

One potential issue with distributional semantics might be inclusion of rare entities in the summary. But with a mix of distributional + pointer-net sort of a technique, that can be avoided. ROUGE helps with exact match based optimization.
FromSiddhartha Banerjee to Xiachong Feng 2:39 PM

https://www.aclweb.org/anthology/P18-3015/

Reinforced Extractive Summarization with Question-Focused Rewards
https://www.aclweb.org/anthology/N18-2102/

Multi-Reward Reinforced Summarization with Saliency and Entailment

Ramakanth Pasunuru, Mohit Bansal. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). 2018.
https://arxiv.org/abs/1909.01610

Answers Unite! Unsupervised Metrics for Reinforced Summarization Models
Abstractive summarization approaches based on Reinforcement Learning (RL) have recently been proposed to overcome classical likelihood maximization.
From Xiachong Feng to Everyone 2:40PM

For those unfamiliar with RL and Policy Gradients, watch some of the nice basic lectures on Reinforcement Learning. It's pretty cool as well.
Our CS3244 course does a recap of at least the first lecture (unfortunately not enough to get to REINFORCE and Policy gradients). We had a previous CS6101 iteration on Deep Reinforcement Learning following UCB's Sergey Levine's course.
https://deepmind.com/learning-resources/reinforcement-learning-lectures-series-2018

This lecture series serves as an introduction to reinforcement learning. Comprised of eight lectures, this series covers the fundamentals of learning and planning in sequential decision problems, all the way up to modern deep RL algorithms.

This lecture series serves as an introduction to reinforcement learning. Comprised of eight lectures, this series covers the fundamentals of learning and planning in sequential decision problems, all the way up to modern deep RL algorithms.

From Min-Yen Kan to Everyone 2:43 PM

The support team should assign at least one member to go through the week’s discussion on Slack (if any) and copy the resources into this scribe document. Expect to spend at least 30 minutes post editing notes from Slack into this archival document.

Other Resources

Recess Week Materials

<Introduction to Fine-tuning on BERT>

Key points discussed-

1. BERT has two tasks- a language modelling task and the net sentence prediction task
2. Fine tuning for binary classification is usually done with binary cross entropy
3. Tokenization is important for mapping to BERT's internal vocabulary (which contains approx 30000 words)
4. Other fine tuning methods- Sentence comparison, Question answering and Classify/Tag each token
5. BERT is too large- so we consider DistilBERT (retains 97% of language understanding and is also 60% faster)
6. Q on chat-
    a. To what extent can we use the CLS token as a sentence representation, other than for classification?
    - as a label comparing two sentences
    b. Do you add a test specific output layer for BERT?
    c. Each time, do we use all fine-tuning methods for each downstream task?
    - Should be able to (just structure it accordingly)
    d. How do we compare two sentences? by formulating it as a classification problem?
    - You can formulate it as a classification problem (both SOP and SMP task). This will be an unsupervised approach
    e. What are the differences between classifying one sentence and classifying each tokens?
    - The token is relevant for pre-training, while with the sentence, you try to build coherence
    f. Some fine tuning methods are task specific while other are task agnostic
    g. Albert has a better performance than BERT even with lesser parameters, why?
    - On its own, Albert does not do better, but they compress it first and then scale it up. It has lesser original parameters, while has more or around same number of parameters as BERT.
7. Albert, another simplification of BERT uses SOP (sentence order prediction). Matrix factorization makes the model smaller.

8. BERT- not good with NLG compared to GPT (auto regression model) due to its bidirectional nature
9. Essentially, what's happening in ALBERT is that it splits the embedding parameters into two smaller matrices. Thus, instead of projecting one hot vectors directly into H (hidden layers), one hot vectors are projected into a smaller, lower dimension matrix E (embedding matrix). Then finally they project E into the H hidden space.

   If you tie H and E (what happens in BERT), and with NLP requiring large V (vocabulary), then your E, which is really V*E, must scale with H and thus you end up with models that can have billions of parameters, but most of which are rarely updated in training.

   Thus, untying the two, results in more efficient parameter usage and thus H (context dependent) should always be larger than E (context independent).

<Introduction to Fine-tuning on GPT>

Reference:

1) Brown, et al. "Language Models are Few-Shot Learners."
2) Radford, et al. "Language Models are Unsupervised Multitask Learners."
3) Radford, et al. "Improving Language Understanding by Generative Pre-Training."

2 approaches to pre-training:

Approach 1: When pre-training and fine-tuning objectives are the same. Eg. Pre train on SQuaD and fine tune on custom QA dataset
Approach 2: Different upstream and downstream task. Eg. BERT approach

Downside of both approaches:

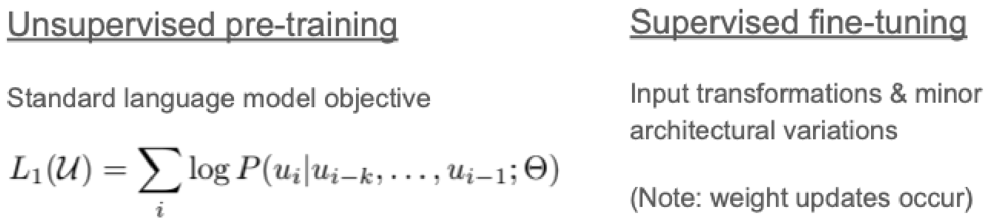
- *Approach 1*
  - Requires a large amount of data for pre training as data used to fine-tune is narrow subset of that task
- *Approach 2*
  - Pretrain-finetune mismatch
- Common to both approaches
  - Need for labelled examples limits their usefulness
  - Poor generalization (post pre training on expansive datasets if you fine tune it on very narrow distribution it generally ends up overfitting)
  - Humans don't need datasets. We're good with 1-2 examples.

GPT (Generative Pre-training) family: Collection of auto regressive language models

GPT-1: Generative pre-training with a transformer works!

- Training can be split in 2 parts:

  Part 1 Part 2

  - 

    Unsupervised pre-training

    Standard language model objective

    $$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

    Supervised fine-tuning

    Input transformations & minor architectural variations

    (Note: weight updates occur)

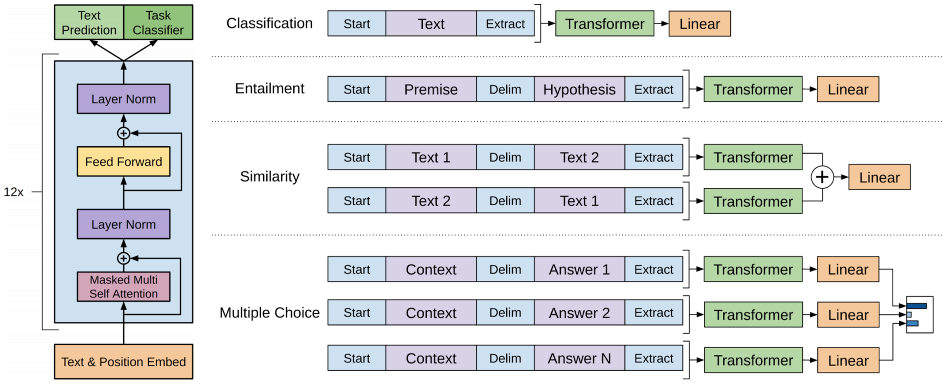
  - 

    Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

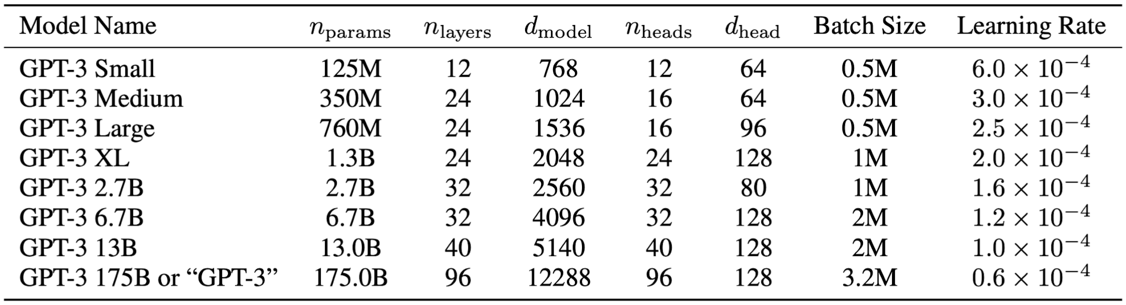
- GPT-1 is largely task agnostic yet it falls short as it doesn't understand very well.

GPT-2: Language models are unsupervised task learners!

- Everything became 'zero-shot' (completely unsupervised)
- Byte-pair encoding. Has a good balance between character and word level language modelling. Can also take into account morphological transformation
- Input transformations- no need for task specific architecture (task agnostic model)
- Much less training data required during fine tuning.
- Instructions are structured with language.

GPT-3: The Scaling Hypothesis is (possibly) true!

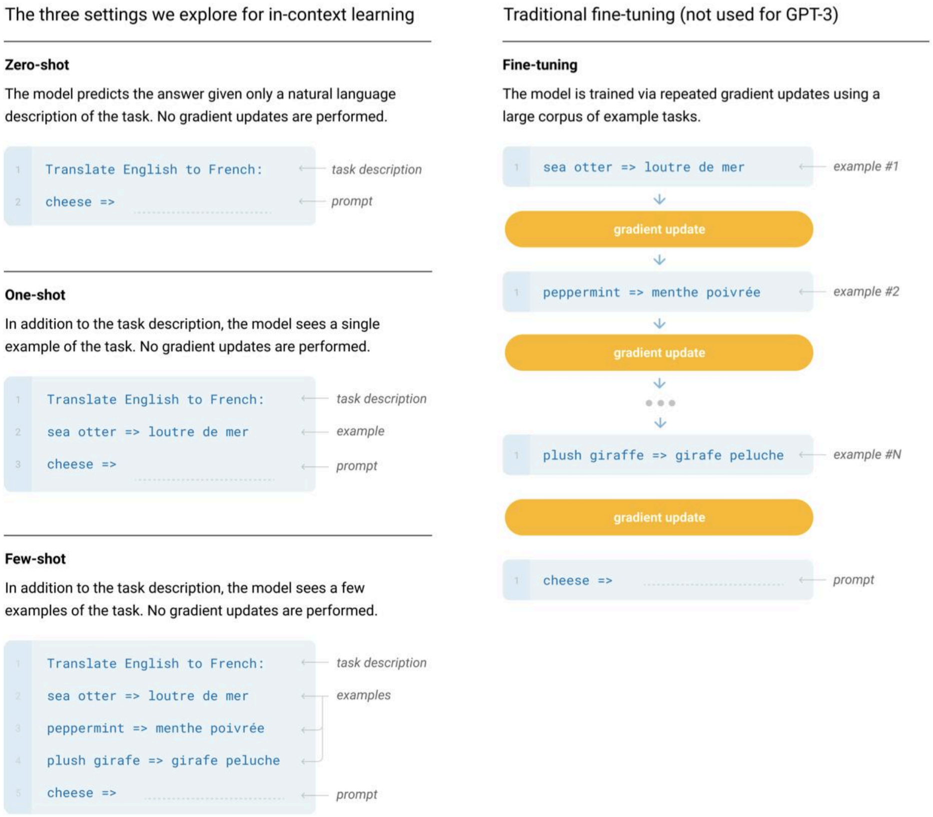
- The model is massive. Check the table below:



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.

- Makes use of in-context learning
  - Using the text input of a pre trained language model as a form of task specification;
  - The model is conditioned on instructions + some set of examples, then asked to complete further instances



The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:
cheese =>
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:
sea otter => loutre de mer
cheese =>
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

Traditional fine-tuning (not used for GPT-3)

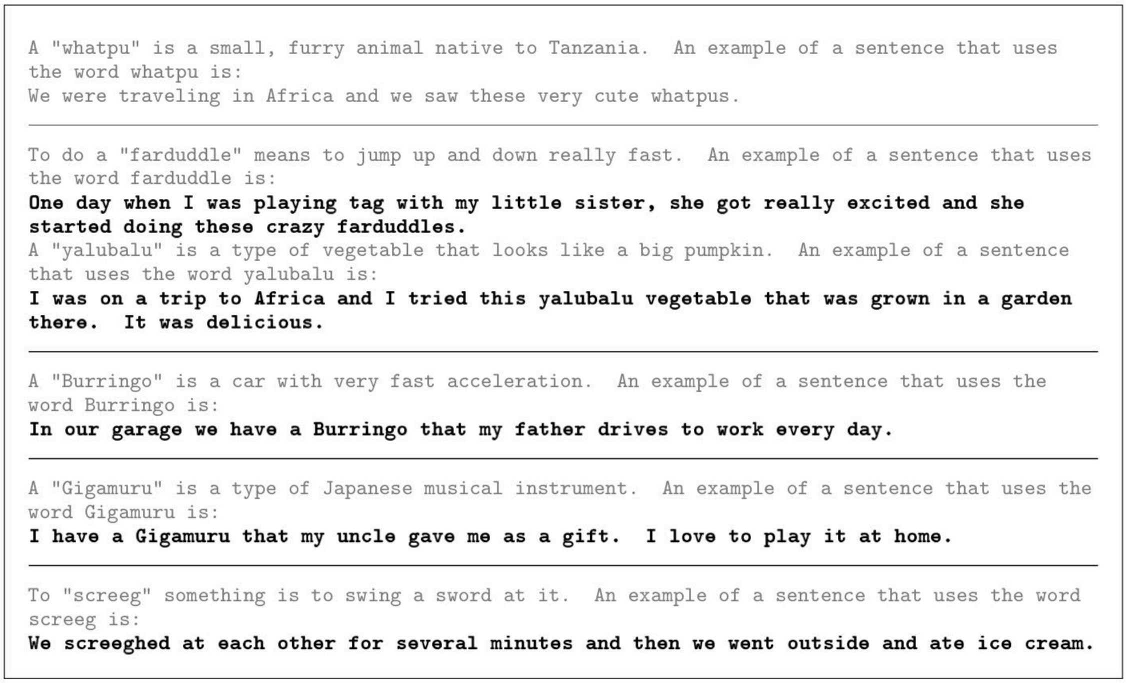
Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

```
sea otter => loutre de mer
peppermint => menthe poivrée
plush giraffe => girafe peluche
cheese =>
```

- GPT-3 tends to do well on these tasks:
  - common-sense reasoning or world-knowledge tasks
  - generative tasks (e.g. writing a news article, completing the end of a story)

    - Winograd-style tasks (certain sentences where pronouns are grammatically ambiguous but for a human they're semantically unambiguous)
    - capturing long-range dependencies (e.g. LAMBADA)
- GPT-3 is not that good in these tasks:
    - tasks that involve comparing two sentences or snippets (e.g. whether word is used in the same way)
    - translation tasks



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.

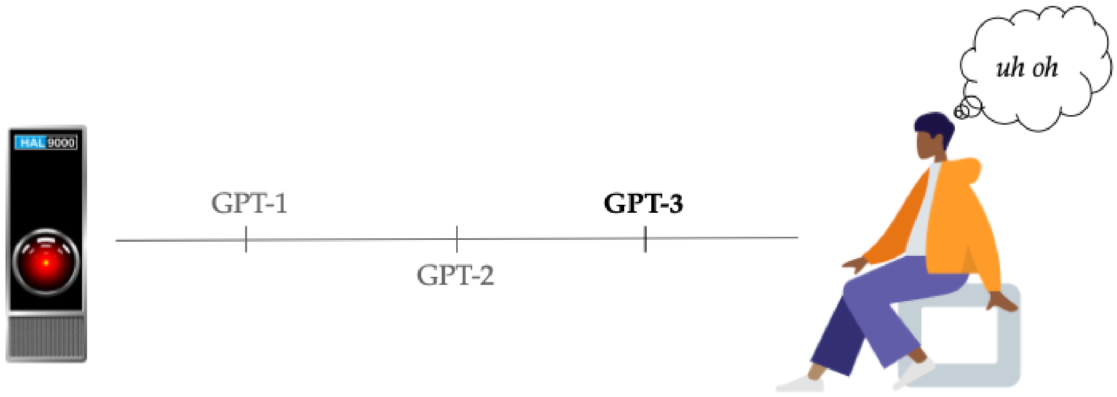
A "Burringo" is a car with very fast acceleration. An example of a sentence that uses the word Burringo is:
In our garage we have a Burringo that my father drives to work every day.

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.

Figure 3.16: Representative GPT-3 completions for the few-shot task of using a new word in a sentence. Boldface is GPT-3's completions, plain text is human prompts. In the first example both the prompt and the completion are provided by a human; this then serves as conditioning for subsequent examples where GPT-3 receives successive additional prompts and provides the completions. Nothing task-specific is provided to GPT-3 other than the conditioning shown here.

Comparing GPT-1 v/s GPT-2 v/s GPT-3



How does GPT-3 perform so well? the model **learns to learn**

The GPT-3 model:

- Develops a broad set of skills and pattern recognition abilities at training time, and then
- Uses those abilities at inference time to rapidly adapt to or recognise the desired task

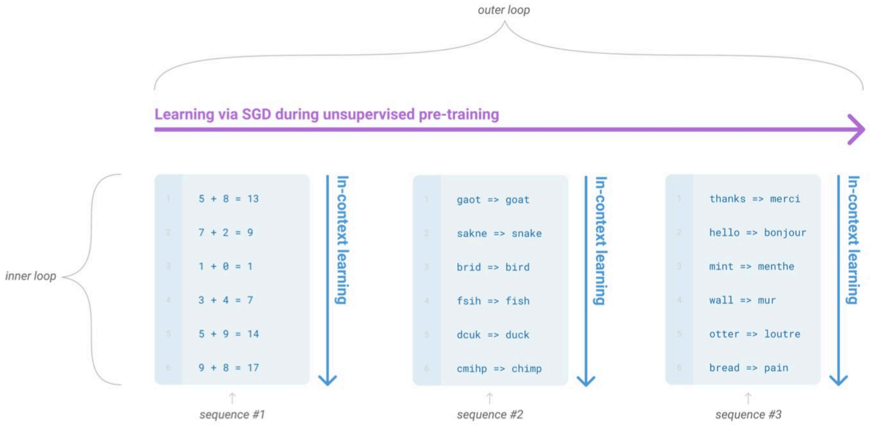


Figure 1.1: Language model meta-learning. During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term "in-context learning" to describe the inner loop of this process, which occurs within the forward-pass upon each sequence. The sequences in this diagram are not intended to be representative of the data a model would see during pre-training, but are intended to show that there are sometimes repeated sub-tasks embedded within a single sequence.

The “in-context learning” approach didn’t work with GPT-2, but it starts to work as you make the model bigger. This essentially boils down to As you make a model larger and train it on large datasets, lots of its problems disappear. Large models become more:

- Powerful
- Generalizable
- Human-like

All this without very complicated architectures.

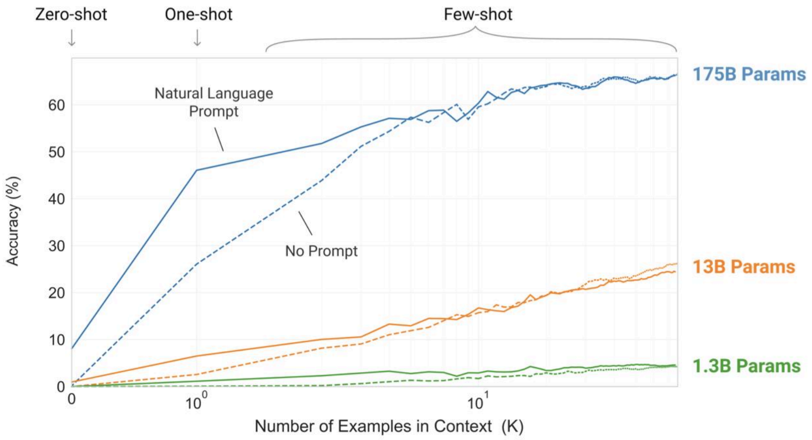


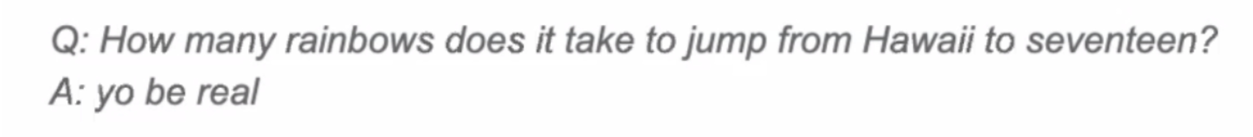
Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper “in-context learning curves” for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

BERT

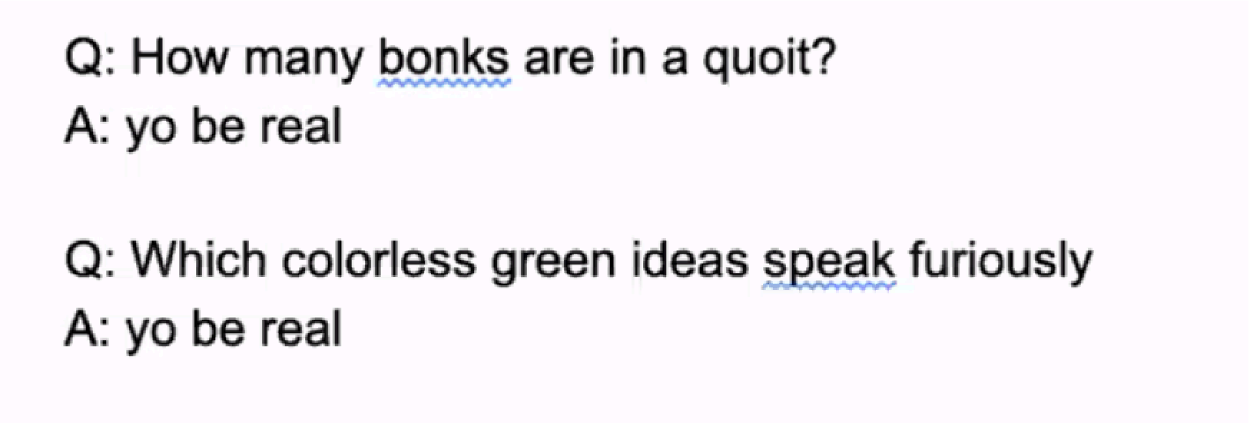
- Only has the next sentence prediction task.

GPT3

- One problem about the GPT is that it tries to decode an answer no matter how well it does on the subject content. So how do we make it more ‘humble’ and accept in some cases that he does not understand the topic/question etc?



Q: How many rainbows does it take to jump from Hawaii to seventeen?
A: yo be real



Q: How many bonks are in a quoit?
A: yo be real

Q: Which colorless green ideas speak furiously
A: yo be real

- Unsatisfying part of the GPT paper is that it misses analysis and is dominated by a lot of experimentational results.

Parameter Eff. TL

Q&A Notes

Questions asked during the Q&A session of Brian's BERT presentation:

- To what extent can we use the CLS token as a sentence representation, other than for classification?
- Do you add a task-specific output layer?
- Each time, do we use all fine-tuning methods for each downstream task?
- How do we compare two sentences? by formulating it as a classification problem?
- What are the differences between classifying one sentence and classifying each tokens?

Questions on Khushhaa's Presentation:

-

Other Resources

- [AlBERT]
- [Natural language generation using BERT]
-