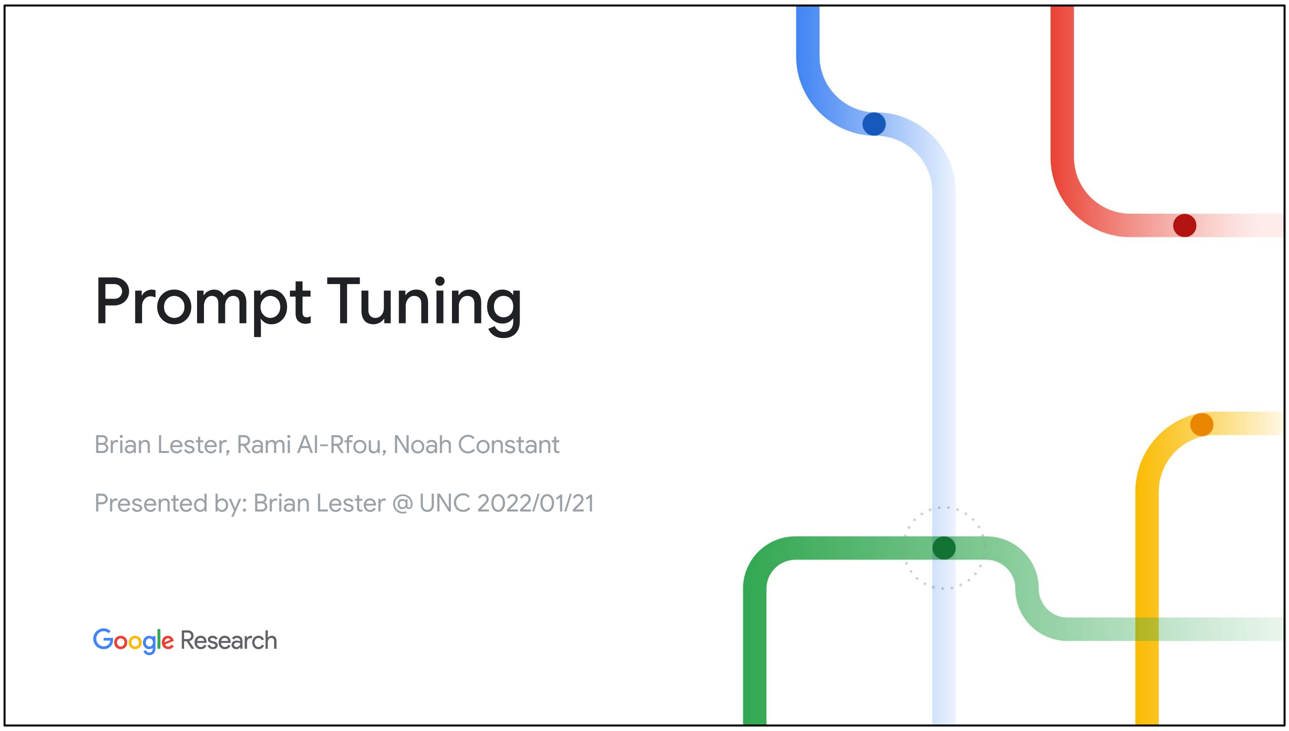


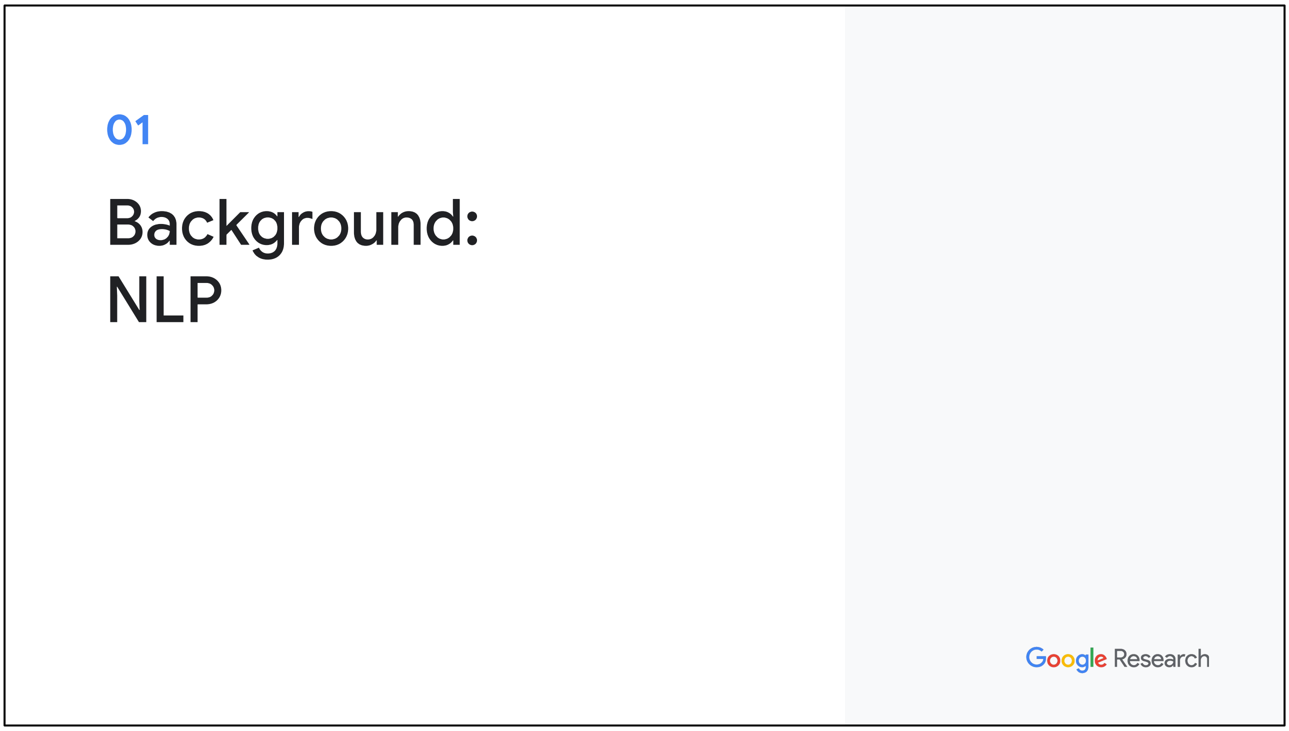
Prompt Tuning

Brian Lester, Rami Al-Rfou, Noah Constant

Presented by: Brian Lester @ UNC 2022/01/21

Google Research

- This presentation is essentially "prompt tuning and such", it covers our EMNLP 2021 paper, our own follow ups, and insights gleaned from others work in the space.
- There is a lot to cover so we'll probably stay at the concept level but I'm happy to dive into details during questions.

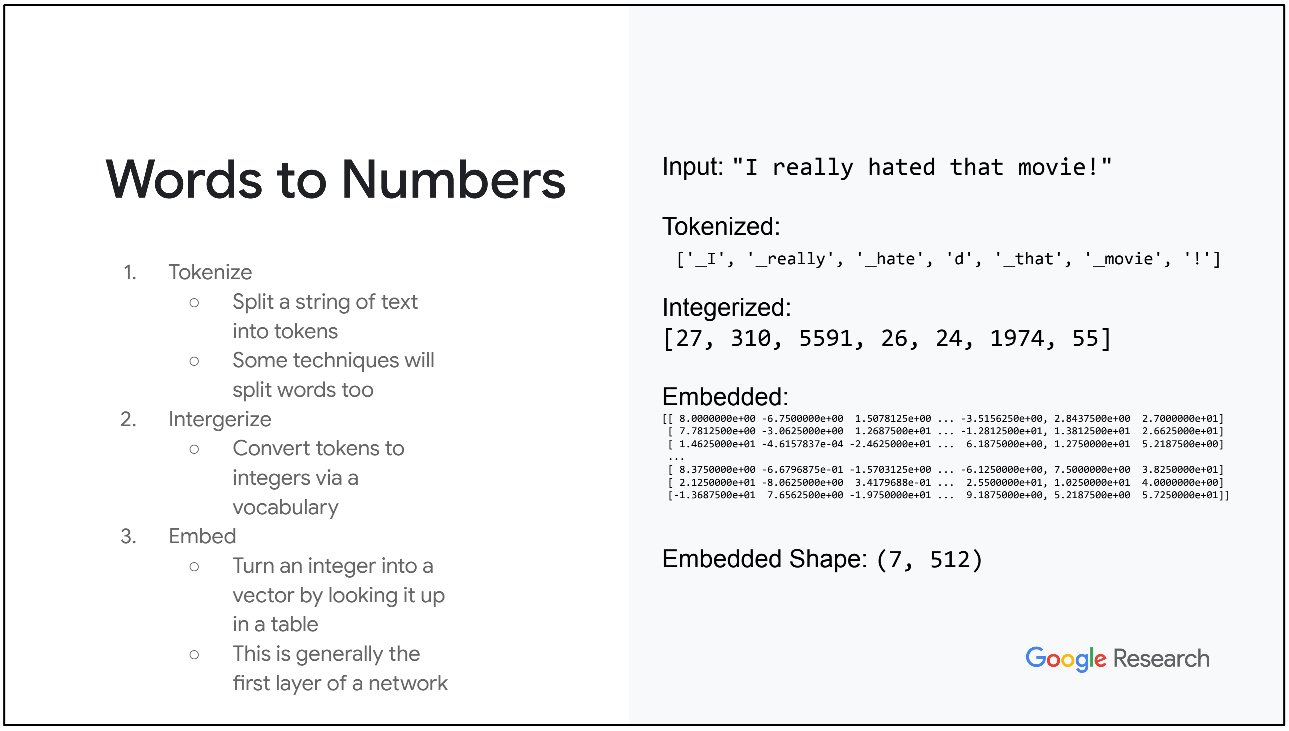


01

Background: NLP

Google Research

I know there is a large mix in the audience, all working on different things, so I'm going to start with a quick crash course in NLP, touching on just the building blocks you'll need for prompt tuning



Words to Numbers

1. Tokenize
   - Split a string of text into tokens
   - Some techniques will split words too
2. Intergerize
   - Convert tokens to integers via a vocabulary
3. Embed
   - Turn an integer into a vector by looking it up in a table
   - This is generally the first layer of a network

Input: `"I really hated that movie!"`

Tokenized:

```
['_I', '_really', '_hate', 'd', '_that', '_movie', '!']
```

Integerized:

```
[27, 310, 5591, 26, 24, 1974, 55]
```

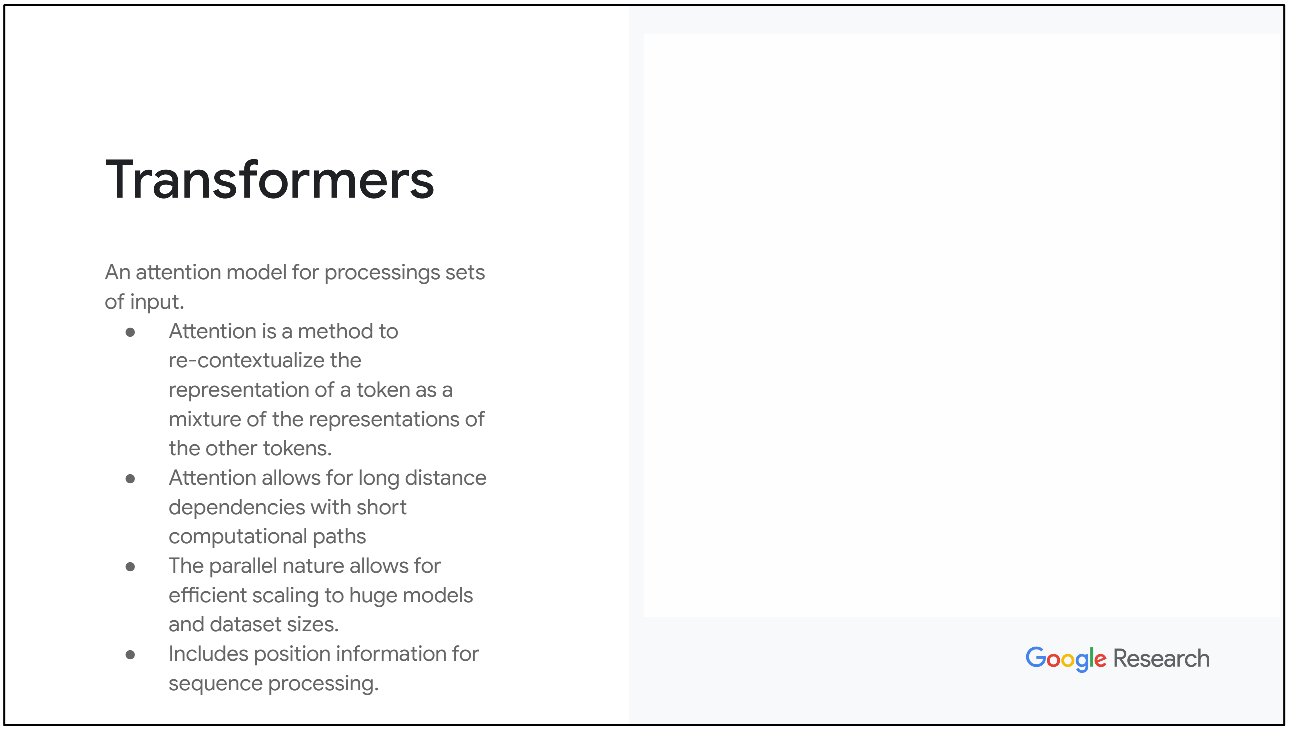
Embedded:

```
[[ 8.0000000e+00 -6.7500000e+00  1.5078125e+00 ... -3.5156250e+00, 2.8437500e+00  2.7000000e+01]
 [ 7.7812500e+00 -3.0625000e+00  1.2687500e+01 ... -1.2812500e+01, 1.3812500e+01  2.6625000e+01]
 [ 1.4625000e+01 -4.6157837e-04 -2.4625000e+01 ...  6.1875000e+00, 1.2750000e+01  5.2187500e+00]
 ...
 [ 8.3750000e+00 -6.6796875e-01 -1.5703125e+00 ... -6.1250000e+00, 7.5000000e+00  3.8250000e+01]
 [ 2.1250000e+01 -8.0625000e+00  3.4179688e-01 ...  2.5500000e+01, 1.0250000e+01  4.0000000e+00]
 [-1.3687500e+01  7.6562500e+00 -1.9750000e+01 ...  9.1875000e+00, 5.2187500e+00  5.7250000e+01]]
```

Embedded Shape: `(7, 512)`

Google Research

- We have text, we can’t just shove that into a model, we need to represent it somehow
- We can’t create a unique representation from a whole string, there are too many possibilities, need to compose smaller representations
- Split the text into individual tokens that we can have representations for.
  - Some split them based on linguistics, into what we think of as words
  - Others split based on the distribution of training text, often creating subword tokens with division that don’t necessarily follow human intuition
- Then we convert each string into integers, this just makes it easier to work with.
- We “embed” each token, that is, look up it’s representation based on it’s integer value.
  - Gives us a vector for each token
  - Our sentence is now a `[Seq-Len, Embed-Size]` matrix
    - This representation of a sentence is important to remember
  - Embedding is generally the first layer of a network and trained with it.



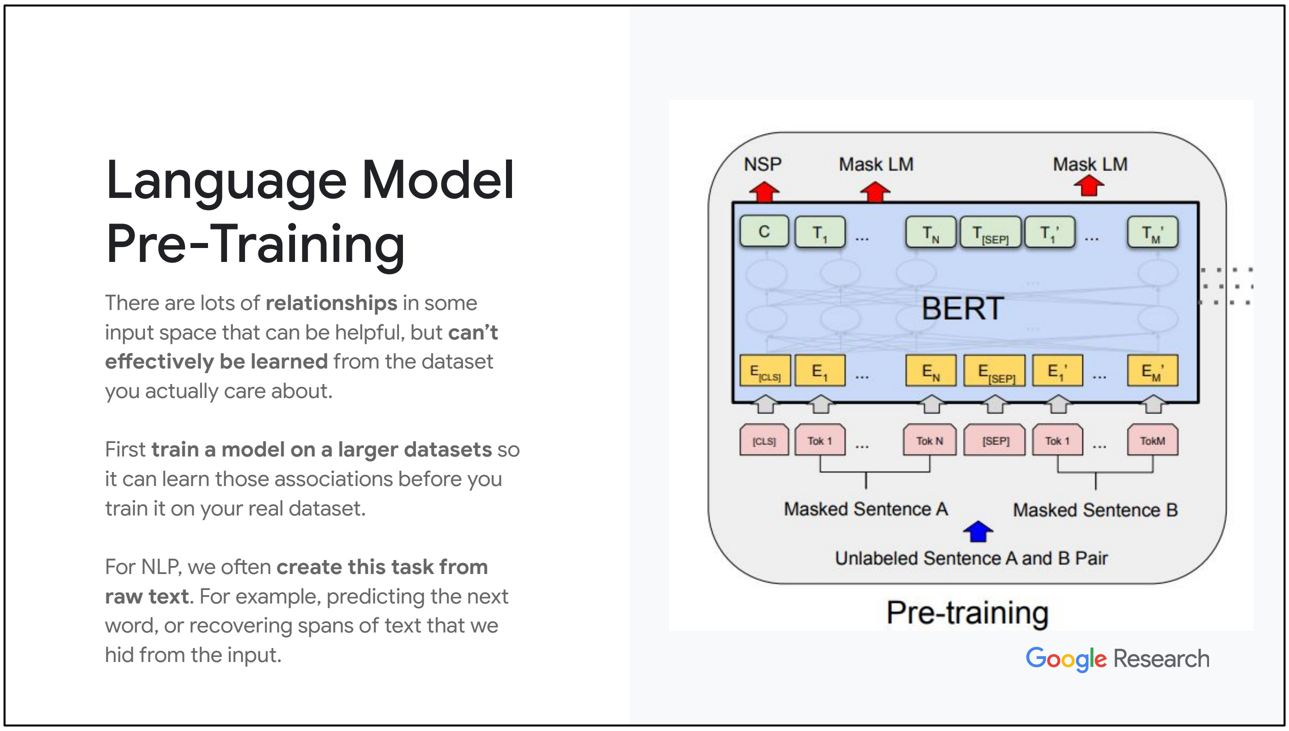
Transformers

An attention model for processings sets of input.

- Attention is a method to re-contextualize the representation of a token as a mixture of the representations of the other tokens.
- Attention allows for long distance dependencies with short computational paths
- The parallel nature allows for efficient scaling to huge models and dataset sizes.
- Includes position information for sequence processing.

Google Research

- Transformers are the current star of NLP.
- They allow use to recontextualized a token based on the rest of the input with attention.
- Attention allows long range dependencies with constant (O(1)) computation, regardless of distance between works.
- Highly parallel allowing for pre-training on massive datasets.
- Attention works on sets, we encode positional information in the tokens to enable sequence processing.

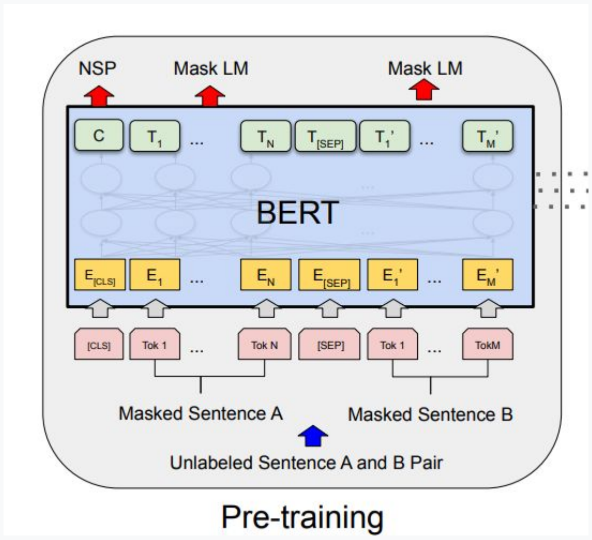


Language Model Pre-Training

There are lots of **relationships** in some input space that can be helpful, but **can't effectively be learned** from the dataset you actually care about.

First **train a model on a larger datasets** so it can learn those associations before you train it on your real dataset.

For NLP, we often **create this task from raw text**. For example, predicting the next word, or recovering spans of text that we hid from the input.

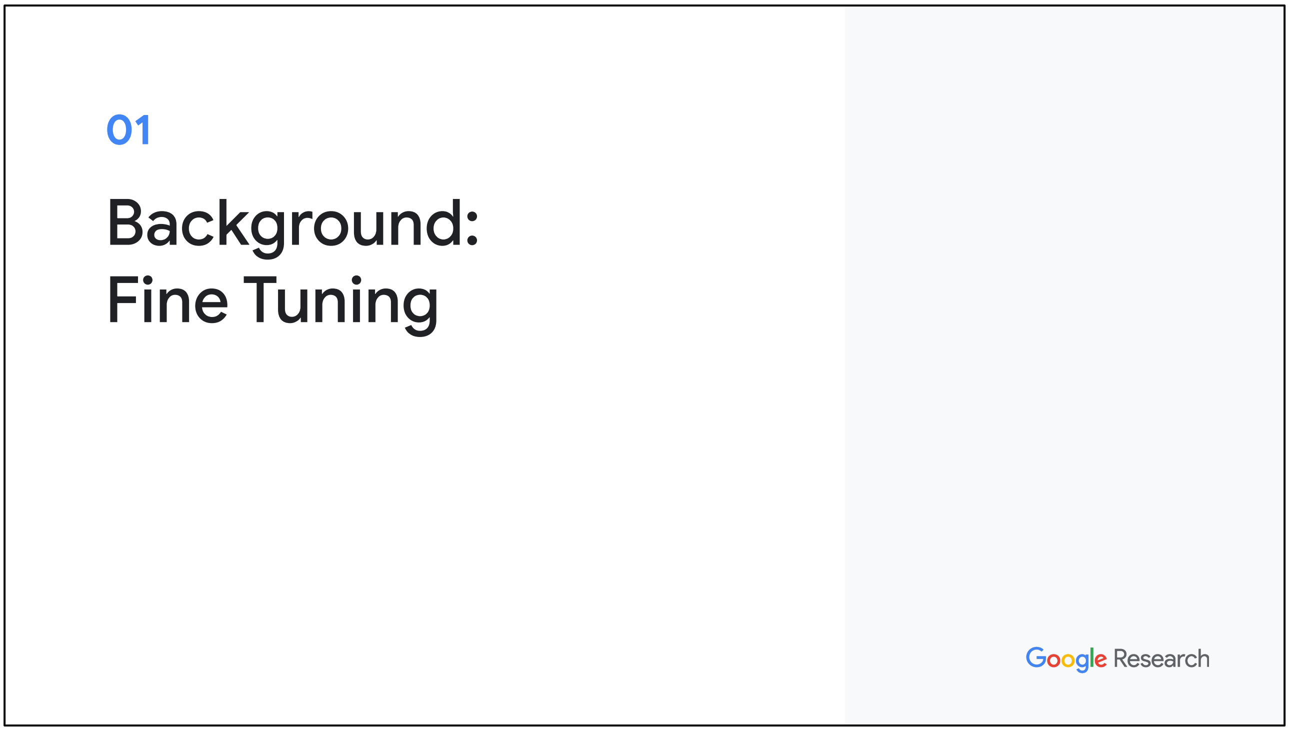


Pre-training

Google Research

- Relations between words, tenses (a dog and a cat are both similar pets, etc) can be helpful in downstream tasks, but maybe can't be inferred from your dataset because it is too small.
- Training on a larger dataset first, called transfer learning, common in many areas, for example starting with a model trained on Image Net in CV of finetuning BERT in NLP
- LM objective, span corruption, MLM

Image from BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (https://arxiv.org/abs/1810.04805)

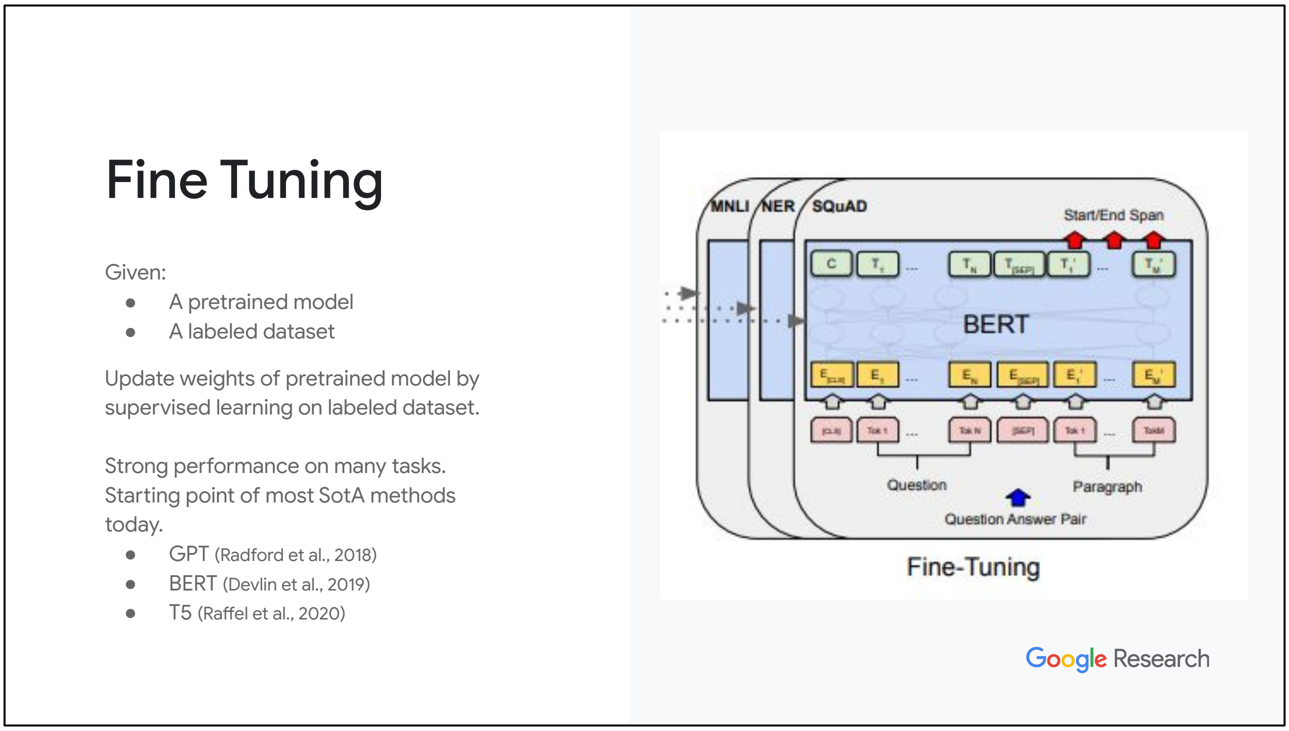


01

Background: Fine Tuning

Google Research

- I’m sure a lot of you will know most of the information in this section but it’s useful for setting the stage for the motivation for a lot of this work
- We are going to talk a little bit about how we use large pre-trained language models in NLP.



Fine Tuning

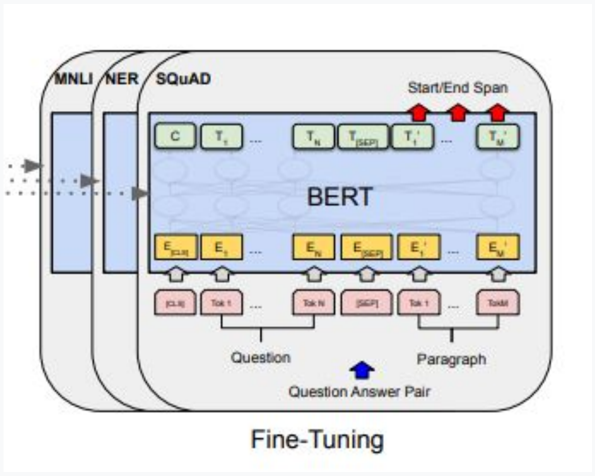
Given:

- A pretrained model
- A labeled dataset

Update weights of pretrained model by supervised learning on labeled dataset.

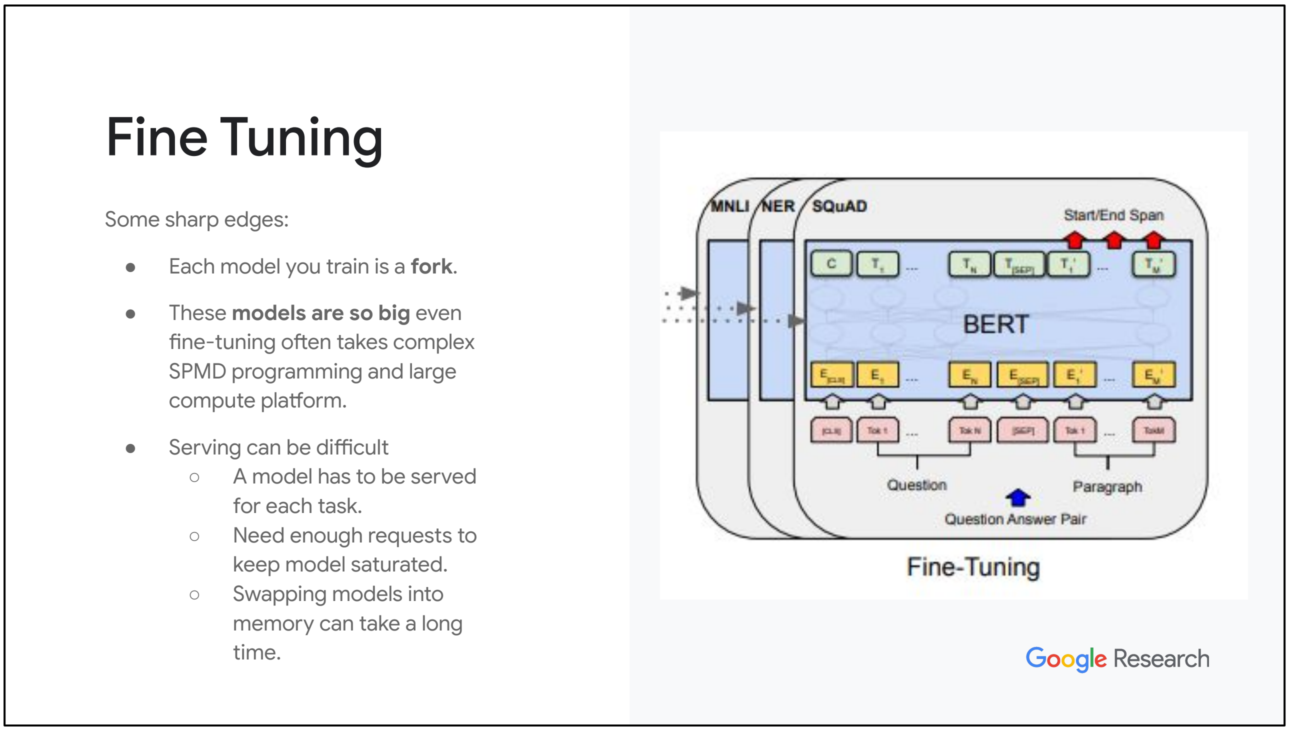
Strong performance on many tasks. Starting point of most SotA methods today.

- GPT (Radford et al., 2018)
- BERT (Devlin et al., 2019)
- T5 (Raffel et al., 2020)



- Fine Tuning a large pre-trained model is the defacto approach for strong performance in NLP
- There really is nothing “fine” about fine-tuning, you update all the parameters, that’s hitting it will a pretty big hammer. We highlight this by calling it model tuning in the paper.
- Lots of famous examples like BERT or T5

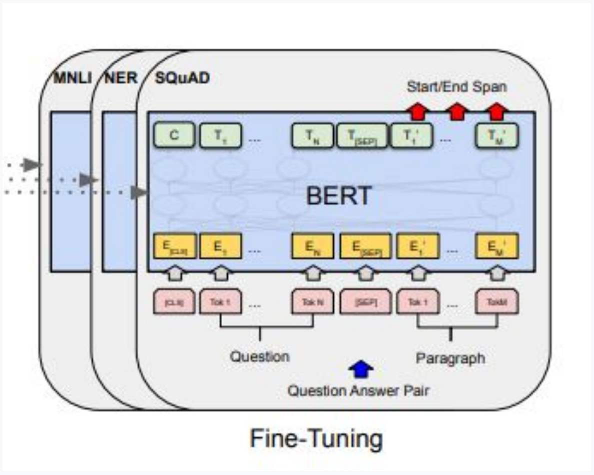
Image from BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (https://arxiv.org/abs/1810.04805)



Fine Tuning

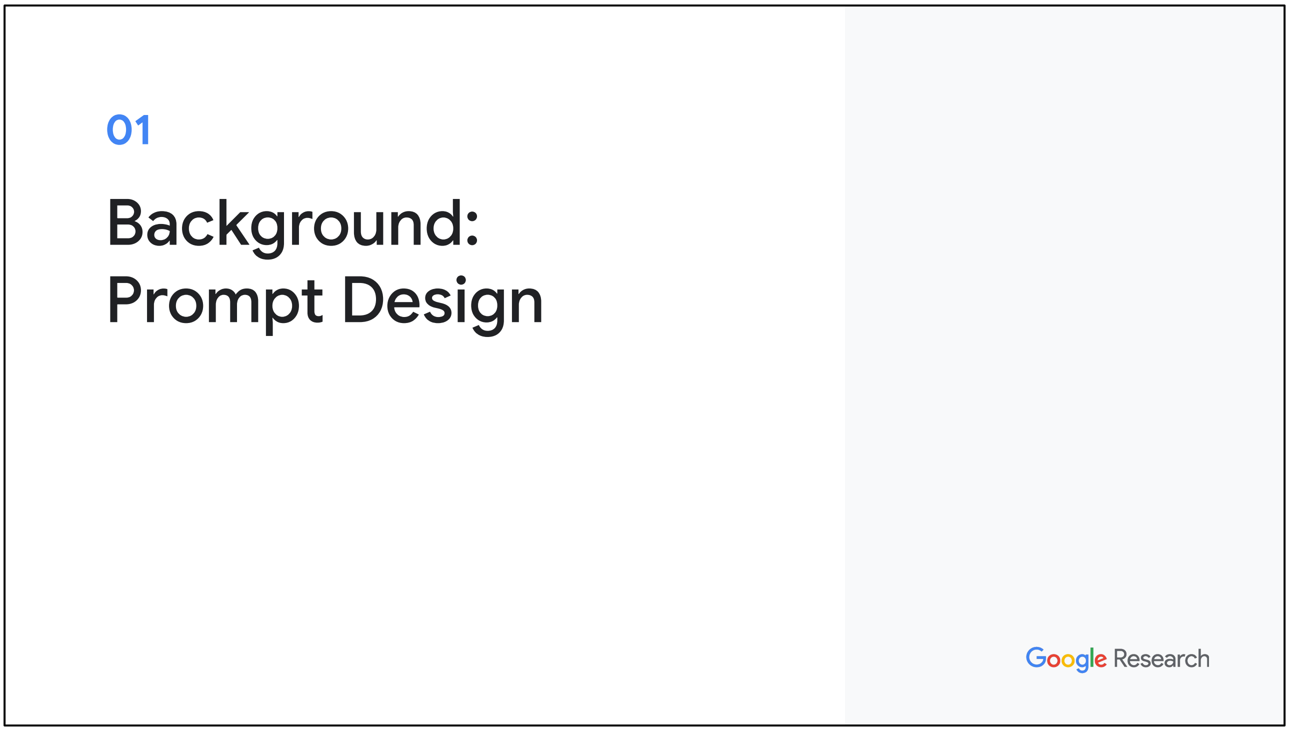
Some sharp edges:

- Each model you train is a **fork**.
- These **models are so big** even fine-tuning often takes complex SPMD programming and large compute platform.
- Serving can be difficult
  - A model has to be served for each task.
  - Need enough requests to keep model saturated.
  - Swapping models into memory can take a long time.



Issues with Fine Tuning:

- Dealing with so many copies of the same huge model
  - Disk space is cheap, but some models, like LaMDA (Thoppilan et al., 2022), takes almost 500 GB on disk
- Even Fine-tuning needs parallel computation.
  - With BERT, the hard part was the pre-training, it was, mostly, reasonable to finetune BERT on a consumer GPU
- Complicates serving, explained later.

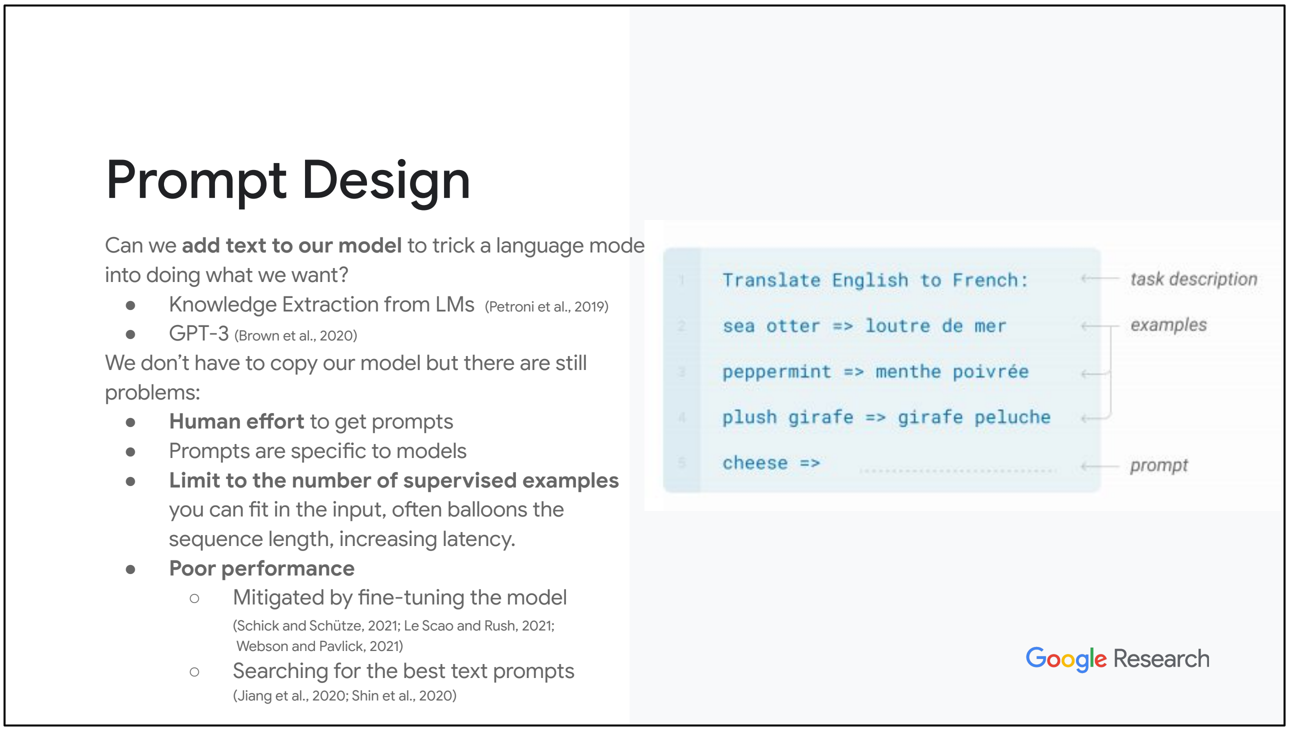


01

Background: Prompt Design

Google Research

Another piece of background is the idea of Prompt Design



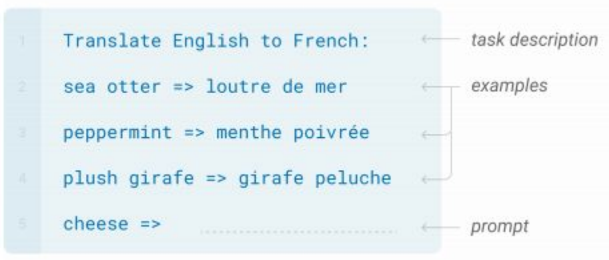
Prompt Design

Can we **add text to our model** to trick a language mode into doing what we want?

- Knowledge Extraction from LMs (Petroni et al., 2019)
- GPT-3 (Brown et al., 2020)

We don't have to copy our model but there are still problems:

- **Human effort** to get prompts
- Prompts are specific to models
- **Limit to the number of supervised examples** you can fit in the input, often balloons the sequence length, increasing latency.
- **Poor performance**
  - Mitigated by fine-tuning the model (Schick and Schütze, 2021; Le Scao and Rush, 2021; Webson and Pavlick, 2021)
  - Searching for the best text prompts (Jiang et al., 2020; Shin et al., 2020)



Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
task description
examples
prompt

Google Research

- Can we trick our model without needing to update it?
- Introduced in the Knowledge extraction from LMs literature
  - Popularized, especially for supervised tasks, by GPT-3
- Explain format here
  - More like import formatting, adding task descriptions before an input, question marks after, etc.
  - Can include supervised examples, people generally refer to this as few-shot or in-context learning
- Pros
  - You don't need to fork your model so you can optimize its serving really well
  - Text prompts feel very interpretable
- Cons
  - You need human designed prompts (search in text space is hard), art not science
  - You need to repeat this human search when the underlying model changes
  - You can only fit a few supervised examples in the model's context, especially for things like reading comprehension with long inputs.
    - This can also increase latency as you have to process longer sequences.
  - Poor Performance
    - GPT-3 175B does 5 points worse on super glue than T5 Base which has 0.1% of the parameters

        - Fine-tuning means you have to copy the model again
        - Searching in the text space is hard

Image from Language Models are Few-Shot Learners (https://arxiv.org/abs/2005.14165)

02

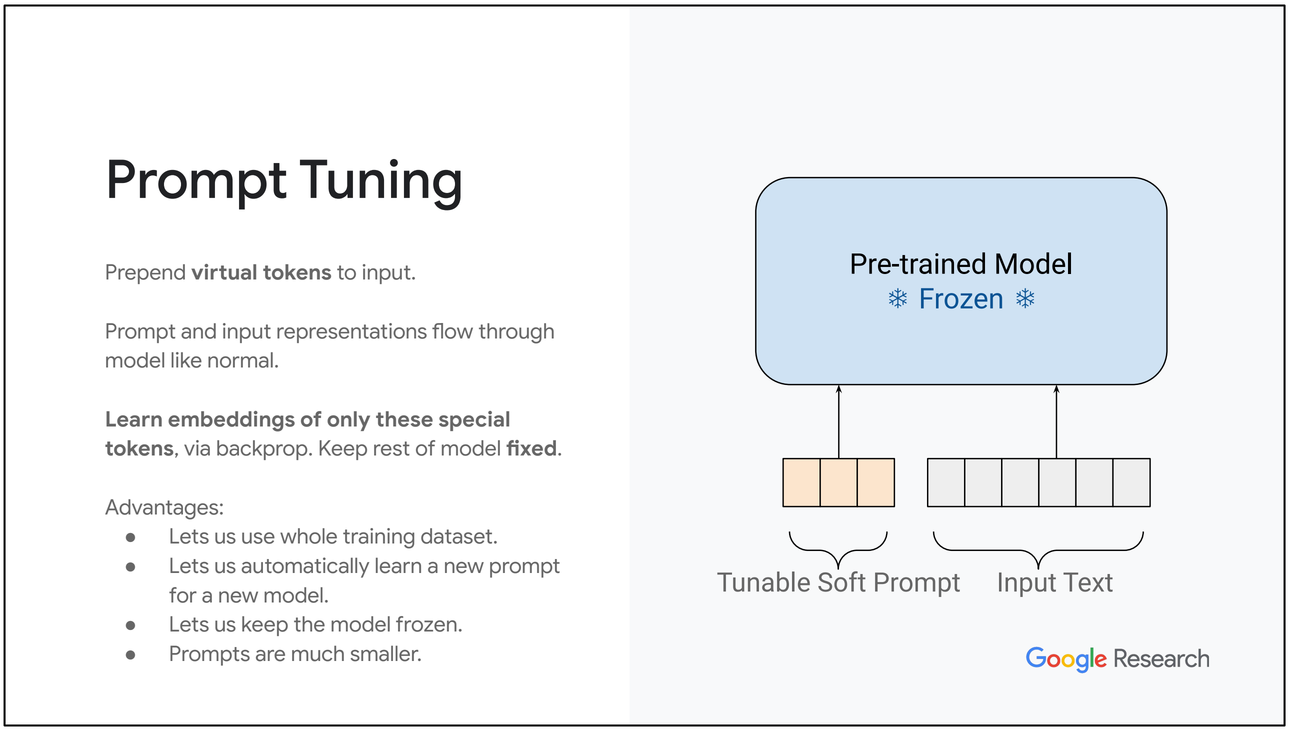
Our Approach: Prompt Tuning (Lester et al., 2021)

- Accepted to EMNLP 2021
- Google AI Blog Post

Google Research

Our approach, Prompt Tuning, splits the difference.

EMNLP 2021



Prompt Tuning

Prepend **virtual tokens** to input.

Prompt and input representations flow through model like normal.

Learn embeddings of only these special tokens, via backprop. Keep rest of model **fixed**.

Advantages:

- Lets us use whole training dataset.
- Lets us automatically learn a new prompt for a new model.
- Lets us keep the model frozen.
- Prompts are much smaller.

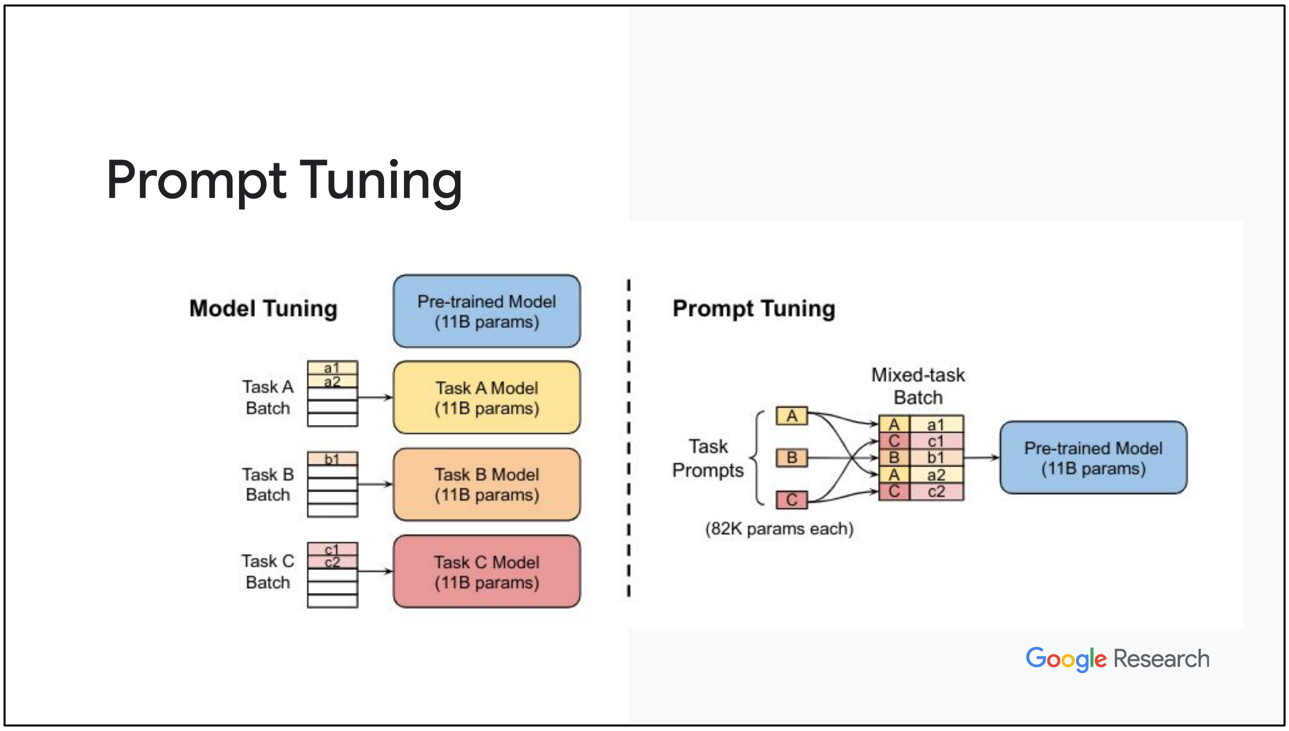
Pre-trained Model
❄ Frozen ❄
Tunable Soft Prompt
Input Text
Google Research

- Prompt Tuning works like this
  - We essentially have a sequence of virtual tokens.
  - These are embedded with a **new** embedding table.
  - We prepend this to the normal embedded input sequence.
  - We run the model forward pass (with this new longer sequence) to get a loss.
  - We run the model backward pass to get gradients.
  - We only apply the gradient updates to the new embedding table, the main model is kept fixed.

Prompt Design can be thought of as searching in the text space over tokens with fixed embeddings
Prompt Tuning can be thought of as searching in the embedding space with a fixed text prompt.

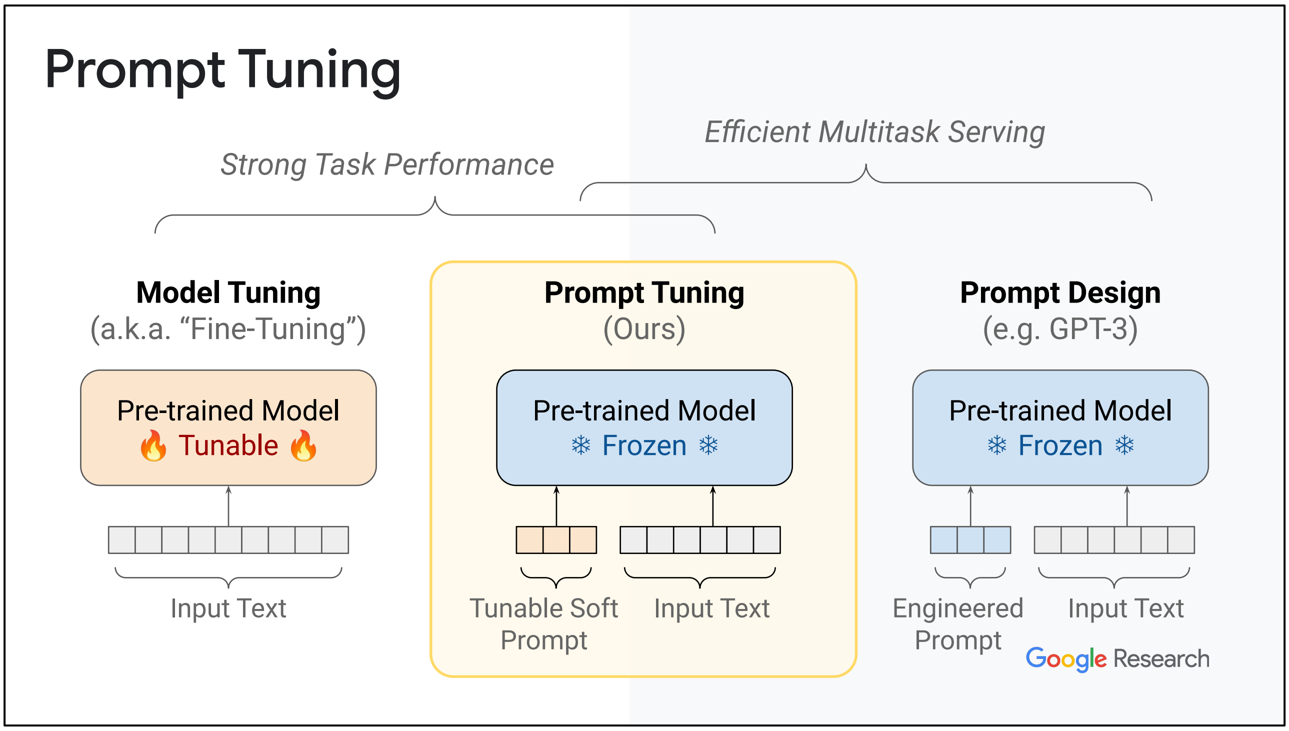
- It has advantages
  - We aren't limited by the number of examples we can fit into the models context, we can learn from a whole dataset.
    - We also don't have as long of a sequence length so we have faster inference
  - We can automatically learn prompts for any model or dataset, it is just backprop, we have been doing that for years.
    - We don't have manual search for prompts on each model.
  - The main model is frozen so we only have to save the very small

- prompts to disk.



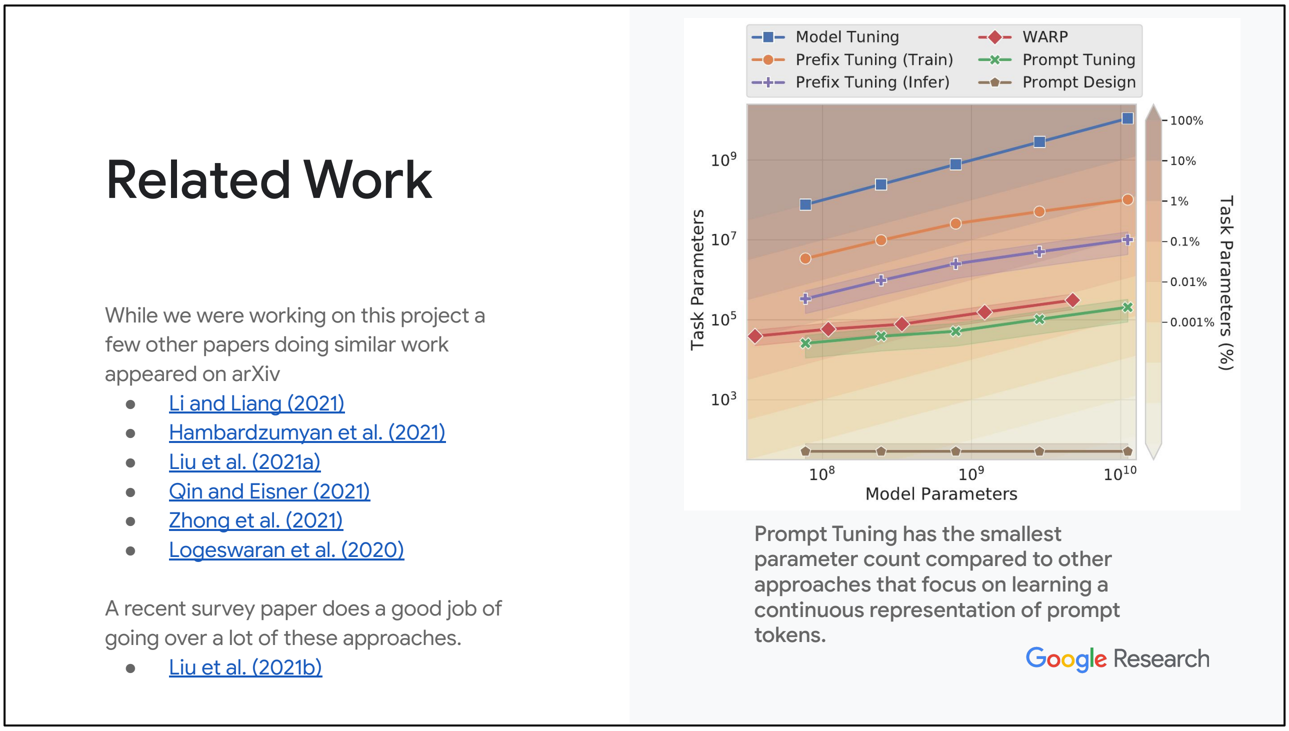
Prompt Tuning

- The main model being frozen gives gives a lot of advantages when serving a model.
- Model Tuning:
  - Each task needs to own model, and that model needs to live in memory
    - You need enough requests to saturate the model to justify it living in memory
    - You need to swap it into memory, slow because so big
    - Either run in O(N) + swap time or O(1) with massive compute resources.
  - Each input only goes to the task model, smaller batches
- Prompt Tuning:
  - Each task is a prompt, much smaller
  - Only need to serve one copy of the frozen model
  - Prompts for various tasks can be applied to different inputs
    - A single, larger mixed task batch of size N
- We have an internal server prototype at the moment where we can pass the input text and a prompt to running model, allowing for a multi-tasking with a single model server.



This diagram reinforces some of the core differences between various approaches.

- Model Tuning gives strong performance but need to fork the model
- Prompt Design can do multitask serving with a single model but has poor performance
- Prompt Tuning has the best of both with strong performance and a frozen core.



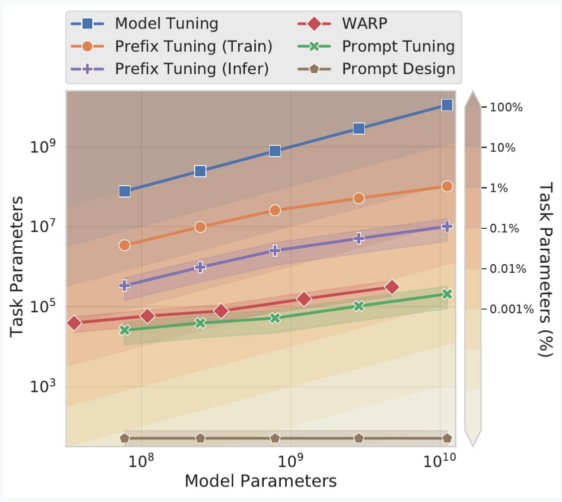
Related Work

While we were working on this project a few other papers doing similar work appeared on arXiv

- Li and Liang (2021)
- Hambardzumyan et al. (2021)
- Liu et al. (2021a)
- Qin and Eisner (2021)
- Zhong et al. (2021)
- Logeswaran et al. (2020)

A recent survey paper does a good job of going over a lot of these approaches.

- Liu et al. (2021b)



Prompt Tuning has the smallest parameter count compared to other approaches that focus on learning a continuous representation of prompt tokens.

Google Research

- While we were working on this, other paper came out :(
- Graph shows the number of updatable parameters each method requires based on if it was applied to T5
- Prompt Tuning is the most parameter efficient method
  - This is because our approach is the simplest, removing things like per-layer variables from while other methods needed.
  - Some methods like adapters (Houlsby, et al., 2019) applied to t5 would result in add about as many parameters as BERT-Large.
- There is more detail about these other methods in our paper and in this Survey from Liu (different Liu) et al. at CMU, and I encourage you to checkout these other works.

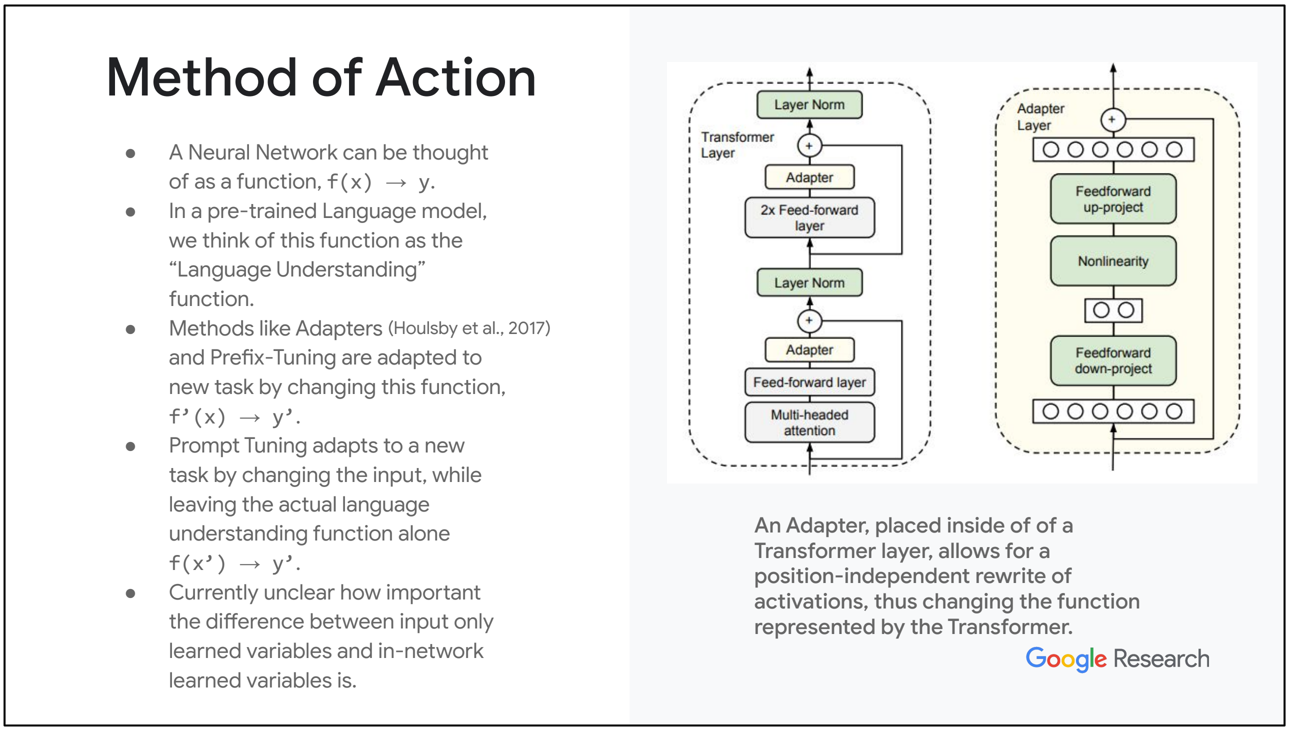
This graph shows the number of task specific (updatable parameters) several different similar works use. Each band represents the mean and stddev of the parameter count as you vary the prompt length from 1 to 100 with T5 as the base model.

Note: The WARP line if offset because they are an Encoder Only architecture.

In addition to being the most parameter efficient, our method is the simplest of all these approaches. It doesn't have:

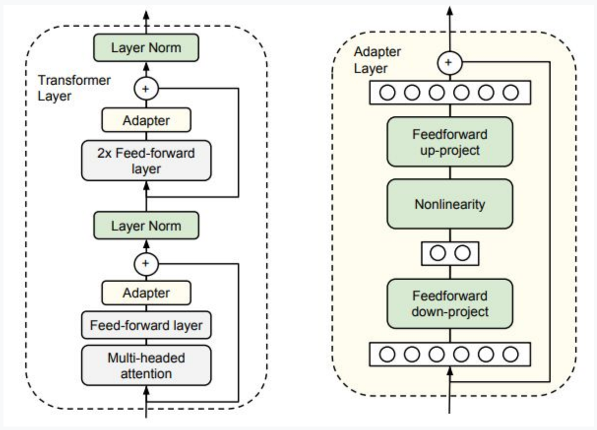
- Variables at every layer
- Text and Learned Prompts woven throughout the text

- Updates to the prompt and the model
- Updates to the text itself
- Trainable output layer



Method of Action

- A Neural Network can be thought of as a function, `f(x) → y`.
- In a pre-trained Language model, we think of this function as the "Language Understanding" function.
- Methods like Adapters (Houlsby et al., 2017) and Prefix-Tuning are adapted to new task by changing this function, `f'(x) → y'`.
- Prompt Tuning adapts to a new task by changing the input, while leaving the actual language understanding function alone `f(x') → y'`.
- Currently unclear how important the difference between input only learned variables and in-network learned variables is.

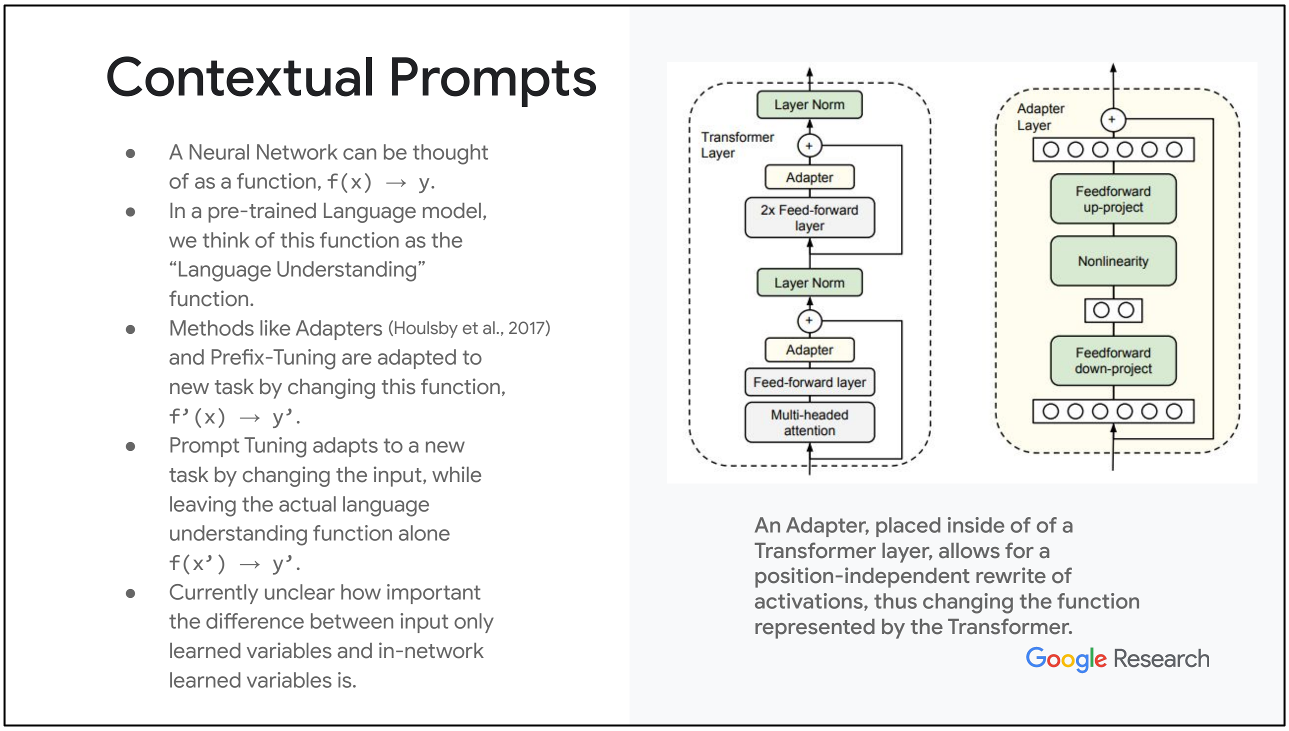


An Adapter, placed inside of of a Transformer layer, allows for a position-independent rewrite of activations, thus changing the function represented by the Transformer.

Google Research

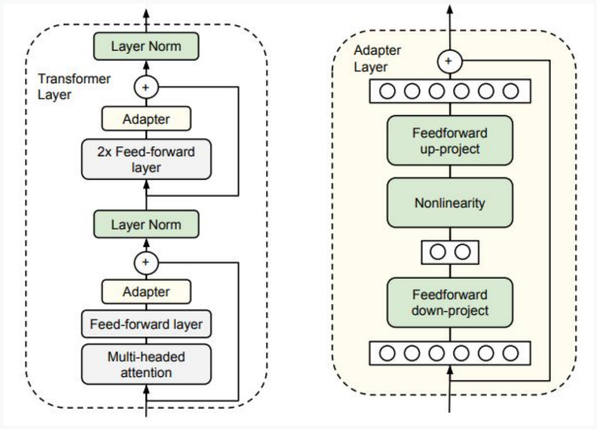
- While some of the differences in methods seem trivial, there are larger conceptual distinctions at a higher level
- Adapters and Prefix Tuning actual overwrite representations inside of the model, this means they are overriding *what* the function does, not just what the function takes *in*
    - In Prefix-Tuning overwriting, you don't have guarantees that you aren't destroying things the pre-trained model might do like storing summary information in the first token.

TODO: Update add LR mention to this?



Contextual Prompts

- A Neural Network can be thought of as a function, `f(x) → y`.
- In a pre-trained Language model, we think of this function as the "Language Understanding" function.
- Methods like Adapters (Houlsby et al., 2017) and Prefix-Tuning are adapted to new task by changing this function, `f'(x) → y'`.
- Prompt Tuning adapts to a new task by changing the input, while leaving the actual language understanding function alone `f(x') → y'`.
- Currently unclear how important the difference between input only learned variables and in-network learned variables is.

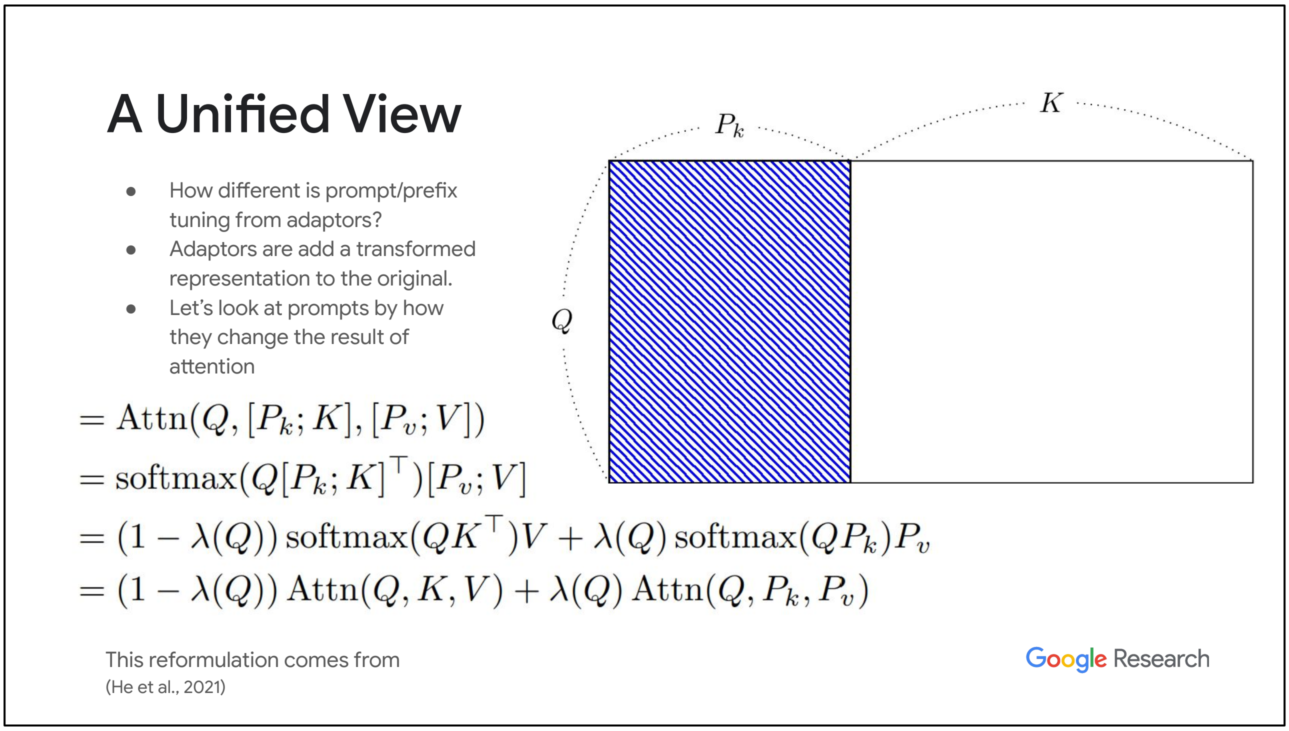


An Adapter, placed inside of of a Transformer layer, allows for a position-independent rewrite of activations, thus changing the function represented by the Transformer.

Google Research

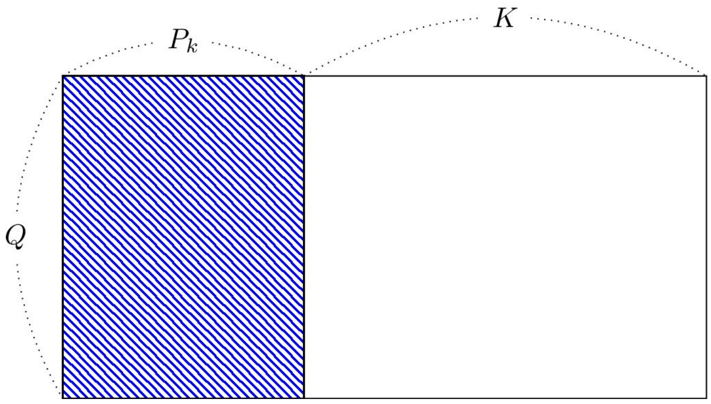
- While some of the differences in methods seem trivial, there are larger conceptual distinctions at a higher level
- Adapters and Prefix Tuning actual overwrite representations inside of the model, this means they are overriding *what* the function does, not just what the function takes *in*
  - In Prefix-Tuning overwriting, you don't have guarantees that you aren't destroying things the pre-trained model might do like storing summary information in the first token.

TODO: Update add LR mention to this?



A Unified View

- How different is prompt/prefix tuning from adaptors?
- Adaptors are add a transformed representation to the original.
- Let's look at prompts by how they change the result of attention



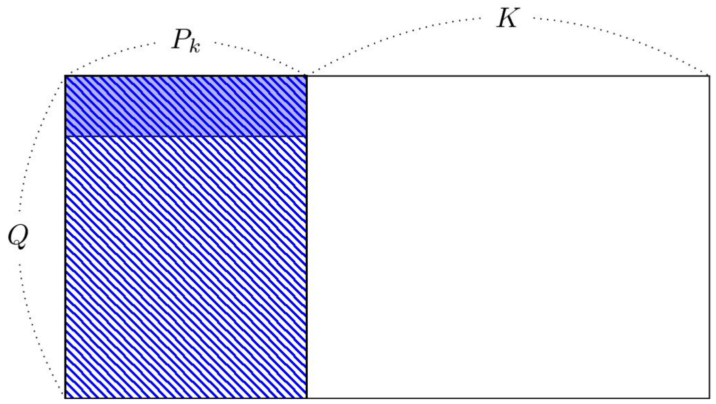
$$
\begin{aligned}
&= \mathrm{Attn}(Q, [P_k; K], [P_v; V]) \\
&= \mathrm{softmax}(Q[P_k; K]^\top)[P_v; V] \\
&= (1 - \lambda(Q))\,\mathrm{softmax}(QK^\top)V + \lambda(Q)\,\mathrm{softmax}(QP_k)P_v \\
&= (1 - \lambda(Q))\,\mathrm{Attn}(Q, K, V) + \lambda(Q)\,\mathrm{Attn}(Q, P_k, P_v)
\end{aligned}
$$

This reformulation comes from (He et al., 2021)

Google Research

This is my first time presenting this aspect so forgive the rough edges and rambling.

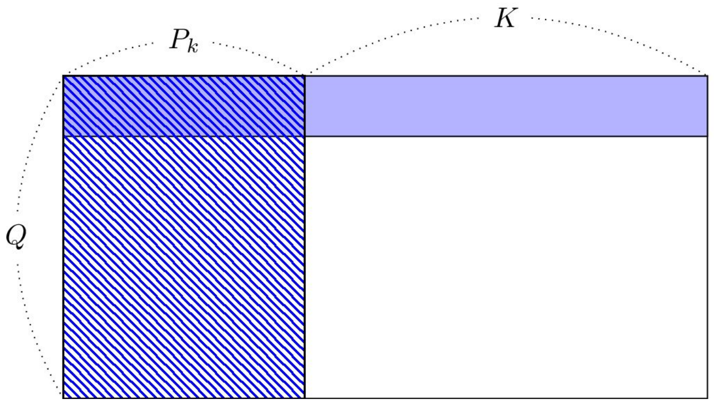
A Unified View



$$\lambda(Q) = \frac{\sum_i \exp(QP_k^\top)_i}{\sum_i \exp(QP_k^\top)_i + \sum_j \exp(QK^\top)_j}$$

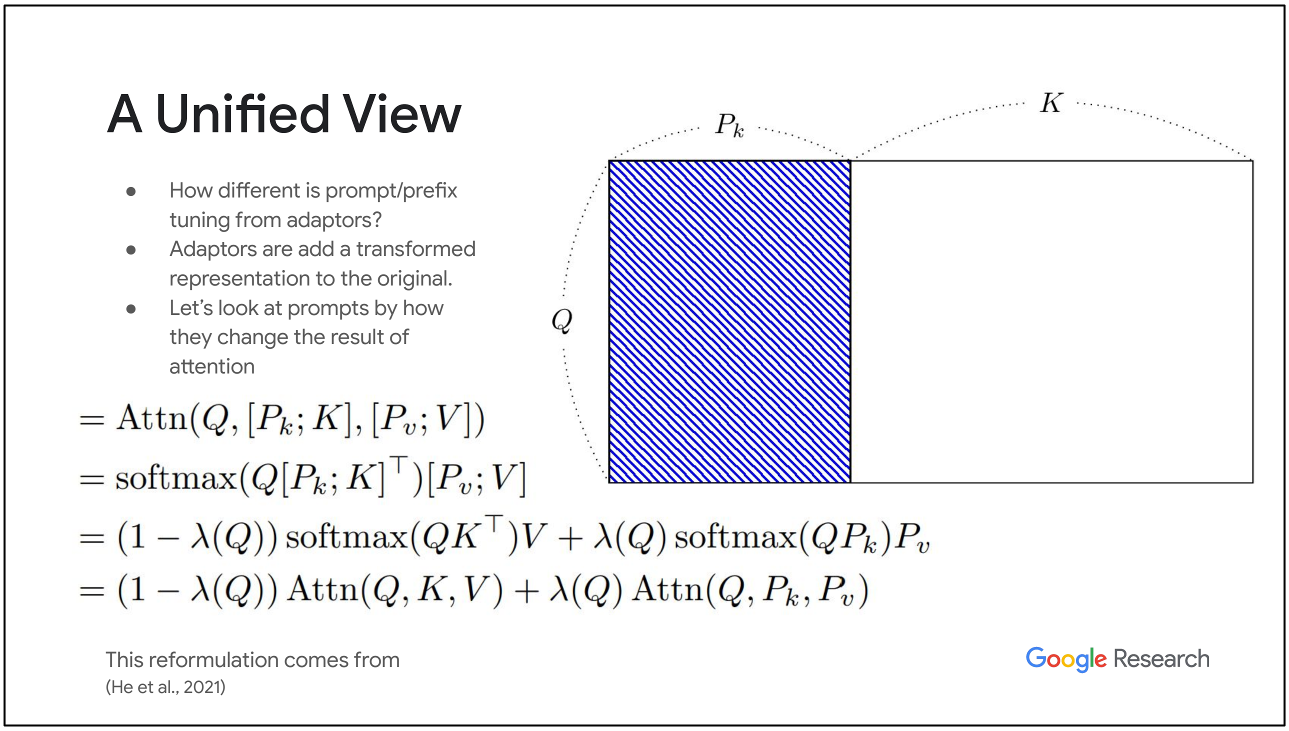
Google Research

A Unified View



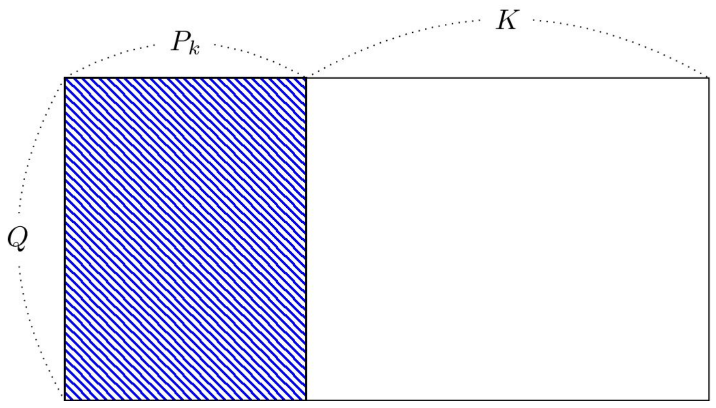
$$\lambda(Q) = \frac{\sum_i \exp(QP_k^\top)_i}{\sum_i \exp(QP_k^\top)_i + \sum_j \exp(QK^\top)_j}$$

Google Research



A Unified View

- How different is prompt/prefix tuning from adaptors?
- Adaptors are add a transformed representation to the original.
- Let's look at prompts by how they change the result of attention



$$\begin{aligned}
&= \mathrm{Attn}(Q, [P_k; K], [P_v; V]) \\
&= \mathrm{softmax}(Q[P_k; K]^\top)[P_v; V] \\
&= (1 - \lambda(Q))\,\mathrm{softmax}(QK^\top)V + \lambda(Q)\,\mathrm{softmax}(QP_k)P_v \\
&= (1 - \lambda(Q))\,\mathrm{Attn}(Q, K, V) + \lambda(Q)\,\mathrm{Attn}(Q, P_k, P_v)
\end{aligned}$$

This reformulation comes from
(He et al., 2021)

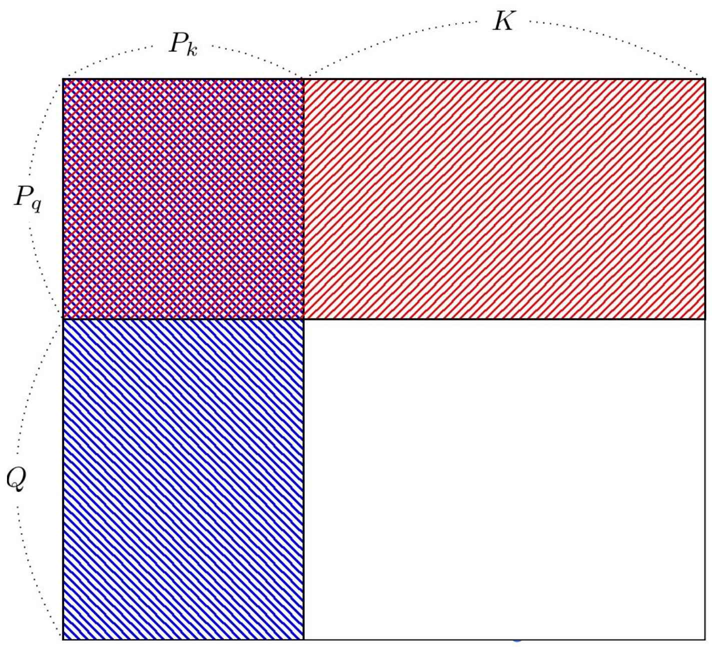
Google Research

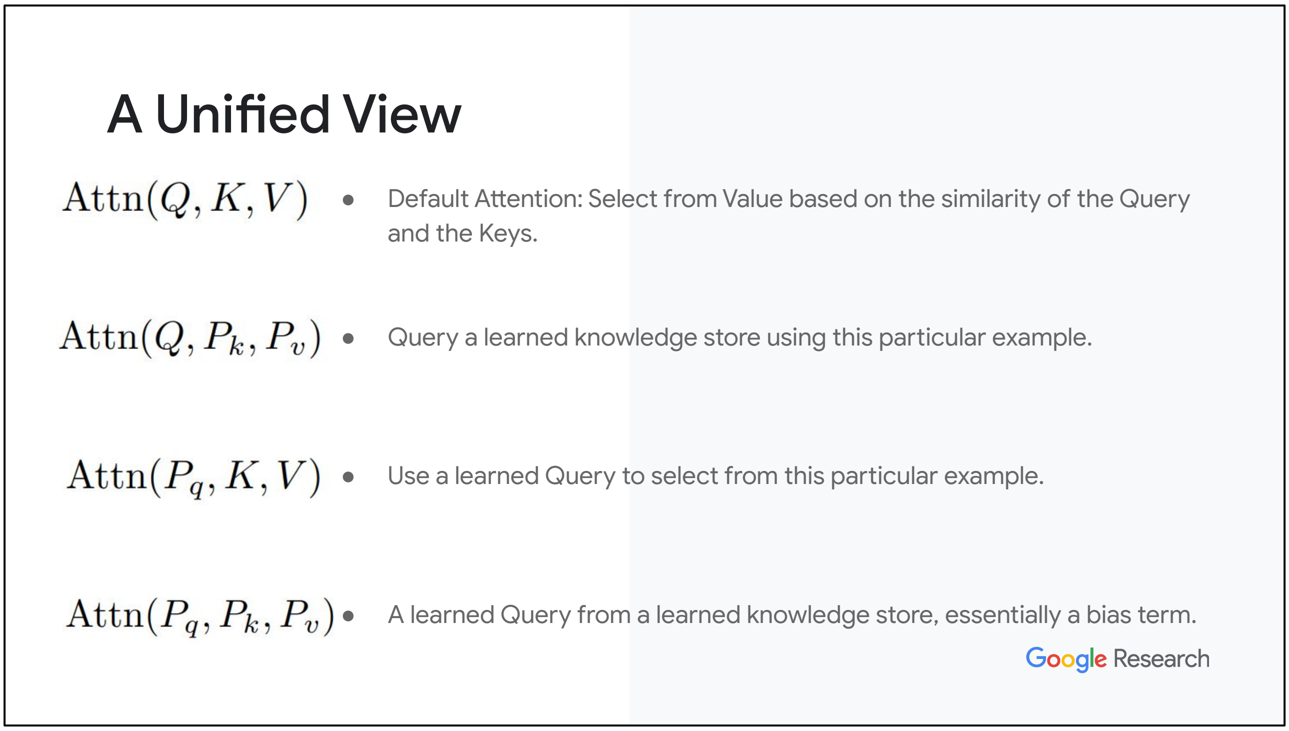
- Prompt tuning as a gated addition to the result of attention.
    - They claim the second term doesn't depend on the input, but the lambda part does, whatever.
    - The addition to attention not depending on the input is actually the weakness of this formulation.
- Their formulation doesn't include positional information on the prompts, but that is often a fixed offset so that doesn't matter much.

A Unified View

- This doesn't account for the Prompt on the Query

$$
\begin{aligned}
&= \operatorname{Attn}([P_q; Q], [P_k; K], [P_v; V]) \\
&= \operatorname{softmax}([P_q; Q][P_k; K]^\top)[P_v; V] \\
&= [(1 - \lambda(P_q)) \operatorname{softmax}(P_q K^\top) V + \lambda(P_q) \operatorname{softmax}(P_q P_k^\top) P_v; \\
&\quad (1 - \lambda(Q)) \operatorname{softmax}(Q K^\top) V + \lambda(Q) \operatorname{softmax}(Q P_k) P_v] \\
&= [(1 - \lambda(P_q)) \operatorname{Attn}(P_q, K, V) + \lambda(P_q) \operatorname{Attn}(P_q, P_k, P_v); \\
&\quad (1 - \lambda(Q)) \operatorname{Attn}(Q, K, V) + \lambda(Q) \operatorname{Attn}(Q, P_k, P_v)]
\end{aligned}
$$





A Unified View

- $\mathrm{Attn}(Q, K, V)$ — Default Attention: Select from Value based on the similarity of the Query and the Keys.
- $\mathrm{Attn}(Q, P_k, P_v)$ — Query a learned knowledge store using this particular example.
- $\mathrm{Attn}(P_q, K, V)$ — Use a learned Query to select from this particular example.
- $\mathrm{Attn}(P_q, P_k, P_v)$ — A learned Query from a learned knowledge store, essentially a bias term.

Google Research

Suggests more design choices to tweak. If P_q is learned at every layer, then Attn(P_q, K, V) would get overridden at each layer so only the last layer would matter. P_k/v at each layer but P_q only at the input layer.

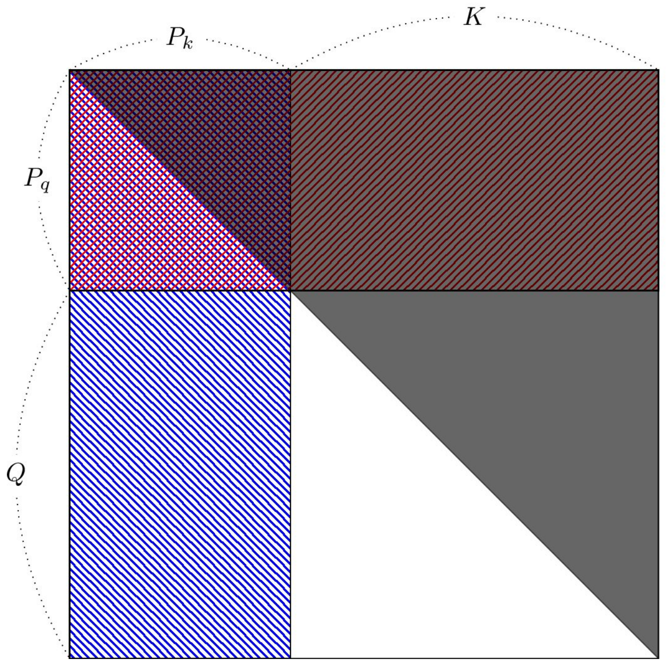
Different sized P_q and P_k/v

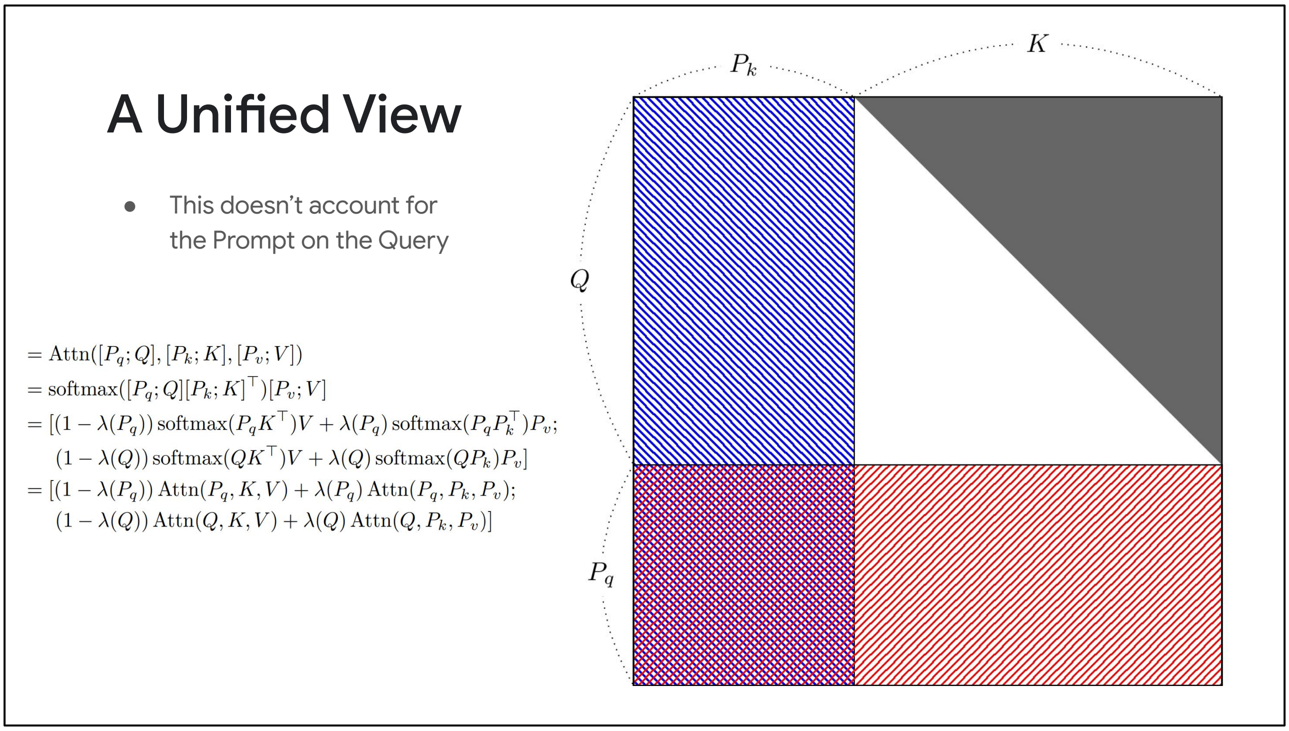
P_q especially important in Encoder/Decoder models?

A Unified View

- This doesn't account for the Prompt on the Query

$$
\begin{aligned}
&= \mathrm{Attn}([P_q; Q], [P_k; K], [P_v; V]) \\
&= \mathrm{softmax}([P_q; Q][P_k; K]^\top)[P_v; V] \\
&= [(1-\lambda(P_q))\,\mathrm{softmax}(P_q K^\top)V + \lambda(P_q)\,\mathrm{softmax}(P_q P_k^\top)P_v; \\
&\quad (1-\lambda(Q))\,\mathrm{softmax}(QK^\top)V + \lambda(Q)\,\mathrm{softmax}(QP_k)P_v] \\
&= [(1-\lambda(P_q))\,\mathrm{Attn}(P_q, K, V) + \lambda(P_q)\,\mathrm{Attn}(P_q, P_k, P_v); \\
&\quad (1-\lambda(Q))\,\mathrm{Attn}(Q, K, V) + \lambda(Q)\,\mathrm{Attn}(Q, P_k, P_v)]
\end{aligned}
$$

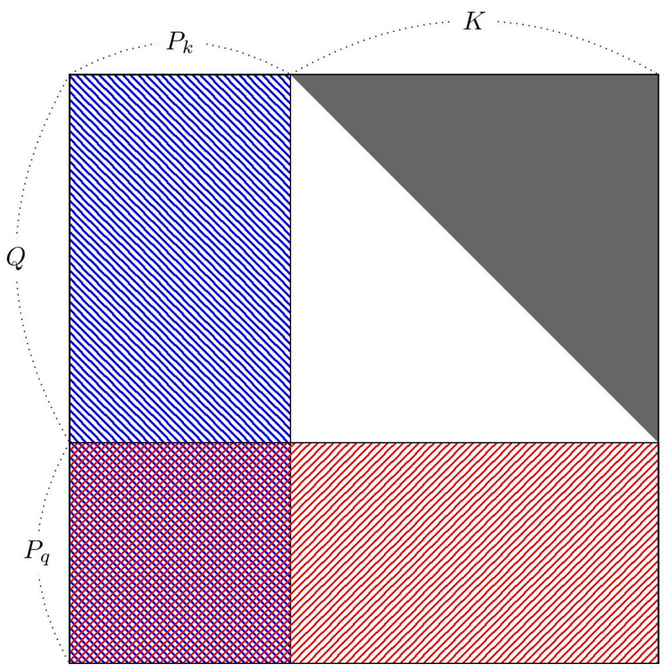


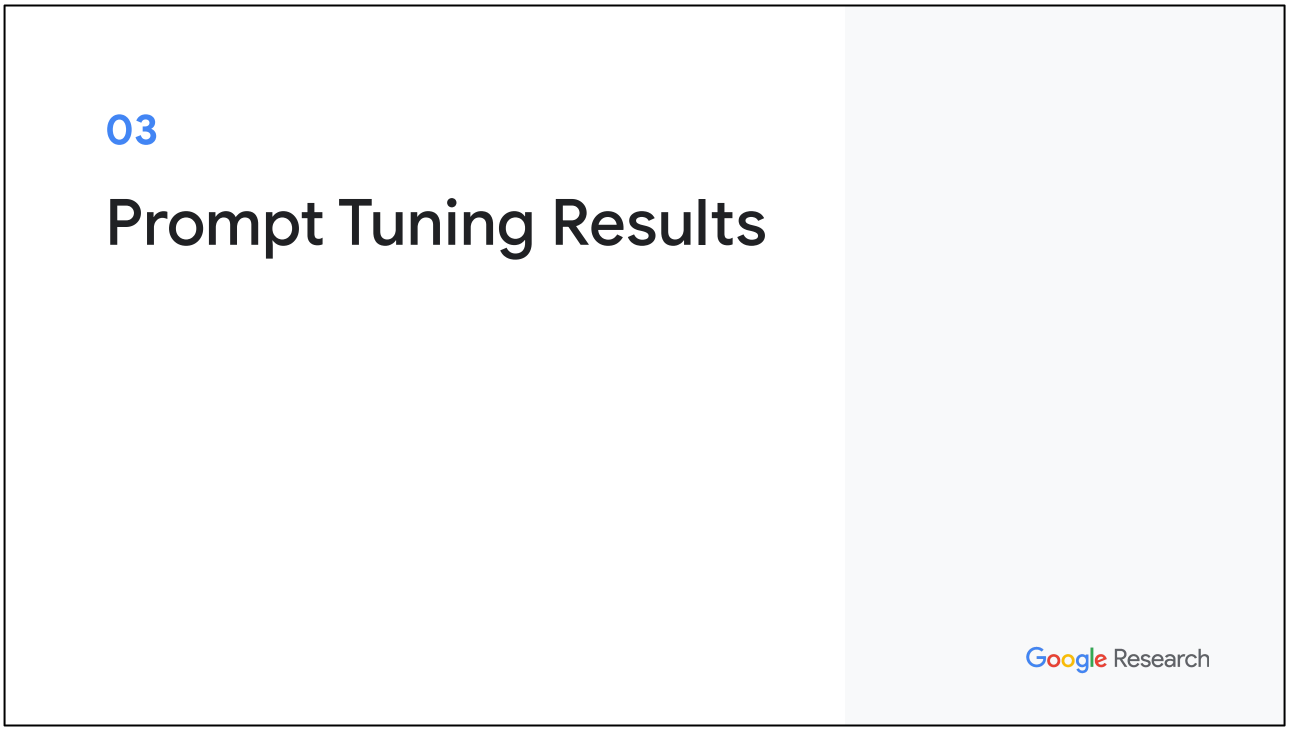


A Unified View

- This doesn't account for the Prompt on the Query

$$
\begin{aligned}
&= \mathrm{Attn}([P_q;Q],[P_k;K],[P_v;V]) \\
&= \mathrm{softmax}([P_q;Q][P_k;K]^\top)[P_v;V] \\
&= [(1-\lambda(P_q))\,\mathrm{softmax}(P_qK^\top)V + \lambda(P_q)\,\mathrm{softmax}(P_qP_k^\top)P_v; \\
&\quad (1-\lambda(Q))\,\mathrm{softmax}(QK^\top)V + \lambda(Q)\,\mathrm{softmax}(QP_k)P_v] \\
&= [(1-\lambda(P_q))\,\mathrm{Attn}(P_q,K,V) + \lambda(P_q)\,\mathrm{Attn}(P_q,P_k,P_v); \\
&\quad (1-\lambda(Q))\,\mathrm{Attn}(Q,K,V) + \lambda(Q)\,\mathrm{Attn}(Q,P_k,P_v)]
\end{aligned}
$$





03

Prompt Tuning Results

Google Research

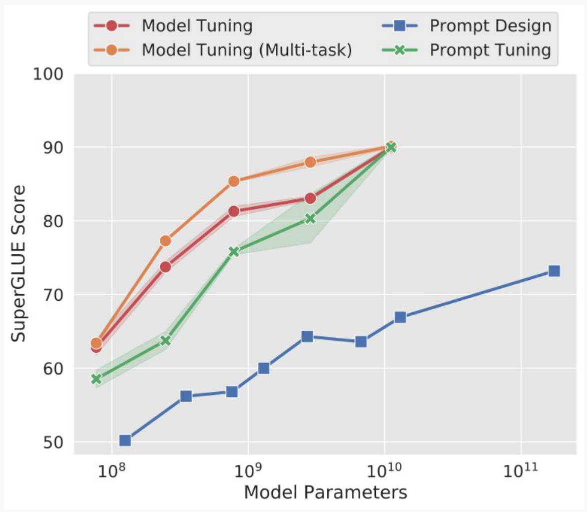
Now the part you were all waiting for, does it work?

Prompt Tuning Results

Prompt Tuning performs very well on SuperGLUE (Wang et al., 2019).

Across all model sizes, Prompt Tuning is far ahead of Prompt Design.

As the model size grows, Prompt Tuning **closes the gap** with Model Tuning.

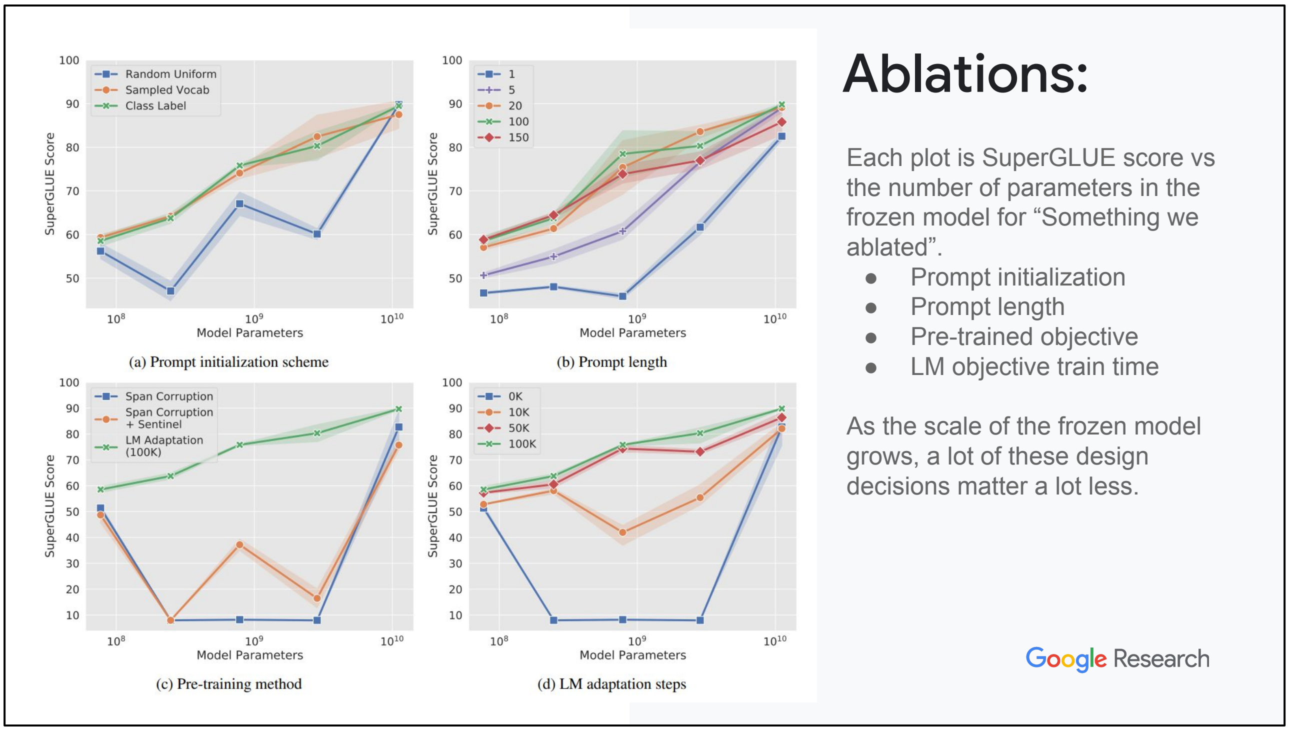


We see that Prompt Tuning is far stronger than prompt design for models of comparable sizes, and that as the size of the LM grows Prompt Tuning catches up to Model Tuning

Google Research

- At a minimum expect it to do better than Prompt Design
- We see it lags behind Modal Tuning at smaller model sizes
- Interestingly, we close the gap as the model scales until we hit XXL where there is virtually no difference in performance

- All work was done with T5 1.1 plus 100k steps of LM adaptation.
- We generate the output label autoregressive (the model needs to actually generate the text) we don't use rank classification.
- We have two model-tuning baselines
  - Single Task model tuning
  - the stronger Multi-Task model tuning.



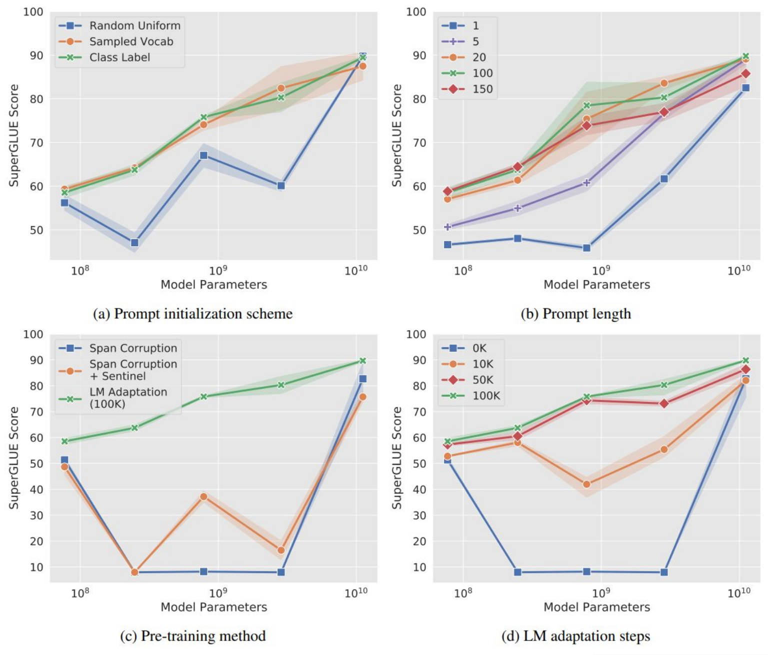
Ablations:

Each plot is SuperGLUE score vs the number of parameters in the frozen model for "Something we ablated".

- Prompt initialization
- Prompt length
- Pre-trained objective
- LM objective train time

As the scale of the frozen model grows, a lot of these design decisions matter a lot less.

Google Research



(a) Prompt initialization scheme

(b) Prompt length

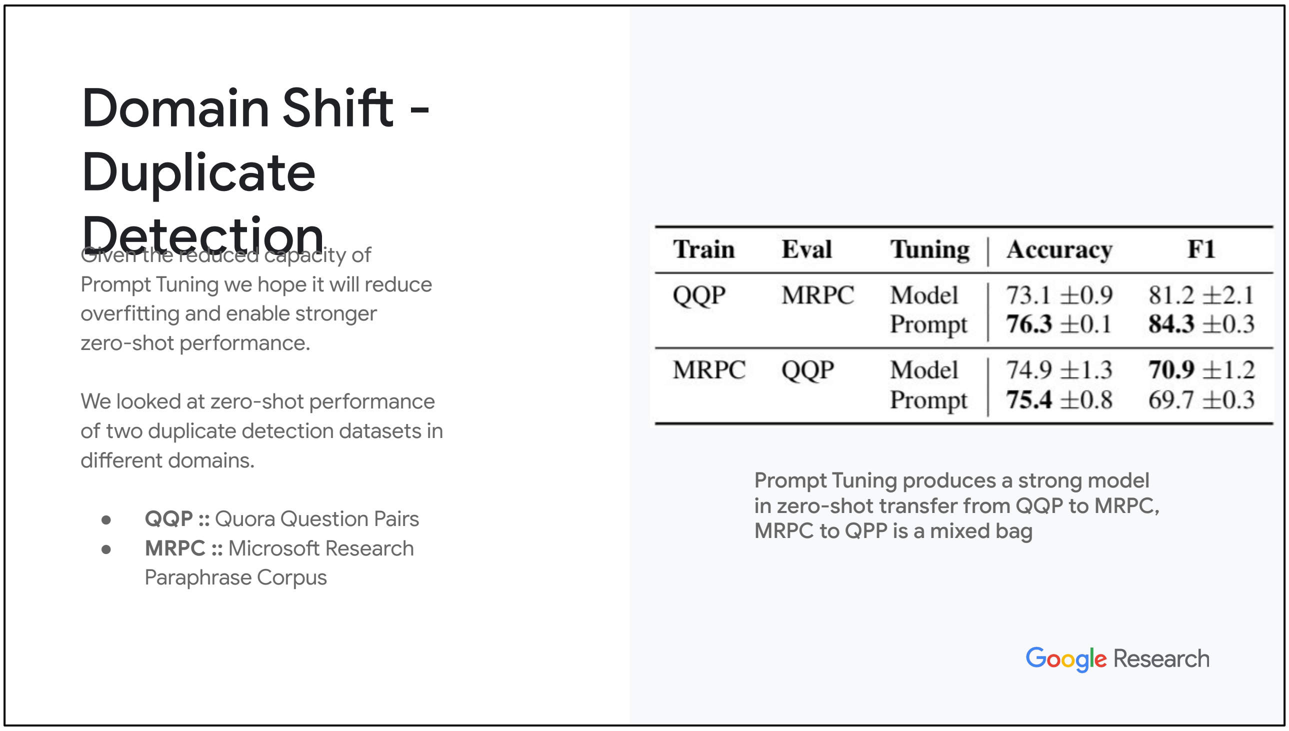
(c) Pre-training method

(d) LM adaptation steps

So I'm throwing at lot of information at you, but it's ok, all these graphs really highlight the same trend.

- They are SuperGLUE score vs # params in the frozen model, as we ablate something.
- They show that as you scale the frozen model, a lot of these decisions matter less.

Take the prompt length for example, as you scale the frozen model you see prompts as short of 5 giving the same performance as longer ones, even a prompt with 1 token in it gives a respectable model.



Domain Shift - Duplicate Detection

Given the reduced capacity of Prompt Tuning we hope it will reduce overfitting and enable stronger zero-shot performance.

We looked at zero-shot performance of two duplicate detection datasets in different domains.

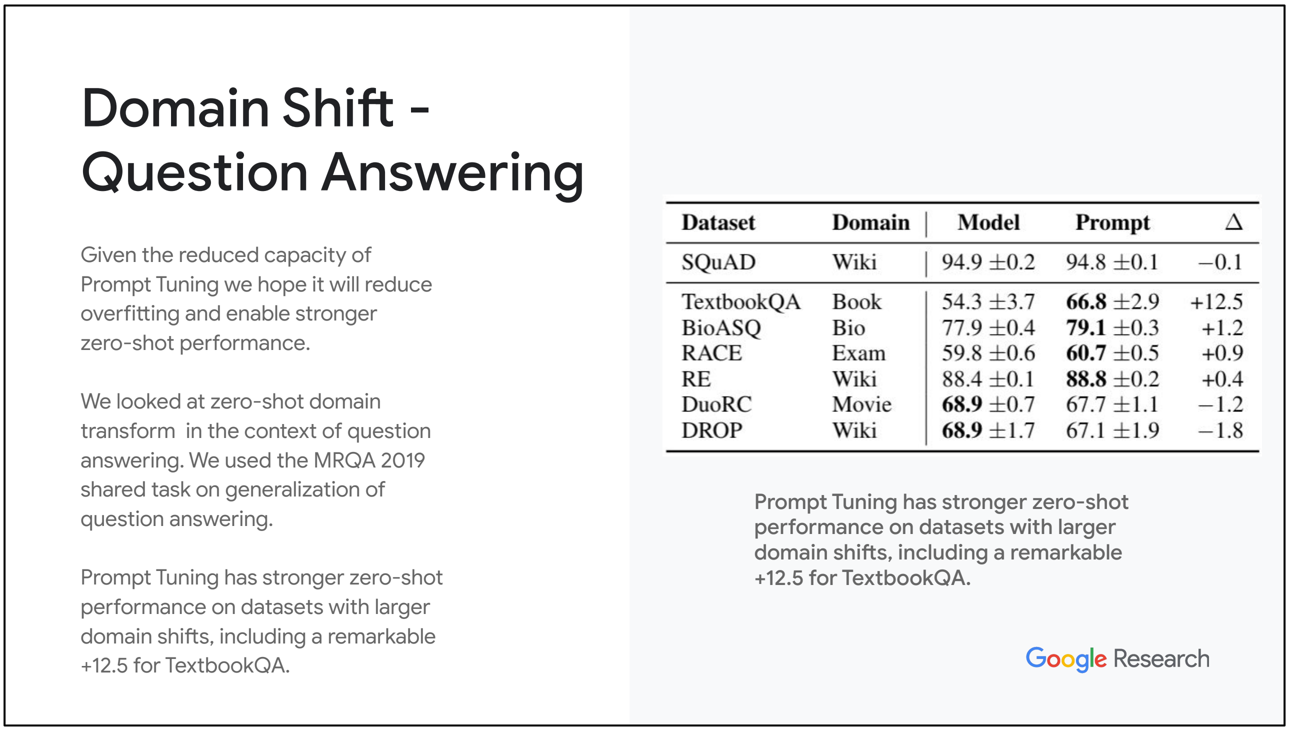
- **QQP ::** Quora Question Pairs
- **MRPC ::** Microsoft Research Paraphrase Corpus

Train	Eval	Tuning	Accuracy	F1
QQP	MRPC	Model	73.1 ±0.9	81.2 ±2.1
		Prompt	**76.3** ±0.1	**84.3** ±0.3
MRPC	QQP	Model	74.9 ±1.3	**70.9** ±1.2
		Prompt	**75.4** ±0.8	69.7 ±0.3

Prompt Tuning produces a strong model in zero-shot transfer from QQP to MRPC, MRPC to QPP is a mixed bag

Google Research

- Zero-shot setup:
  - Trained model/prompt on the training dataset (QQP or MRPC)
  - Selected the best checkpoint based on the development set for that training dataset (QQP or MRPC)
  - Evaluated that checkpoint on the other dataset (MRPC or QQP) development set
- Both are Duplicate Detection datasets:
  - MRPC is paraphrases from a newswire corpus
  - QQP is duplicate questions from the Quora website
    - Domain (newswire vs user generated content) shift
    - Questions only (QQP) vs all text (MRPC)



Domain Shift - Question Answering

Given the reduced capacity of Prompt Tuning we hope it will reduce overfitting and enable stronger zero-shot performance.

We looked at zero-shot domain transform in the context of question answering. We used the MRQA 2019 shared task on generalization of question answering.

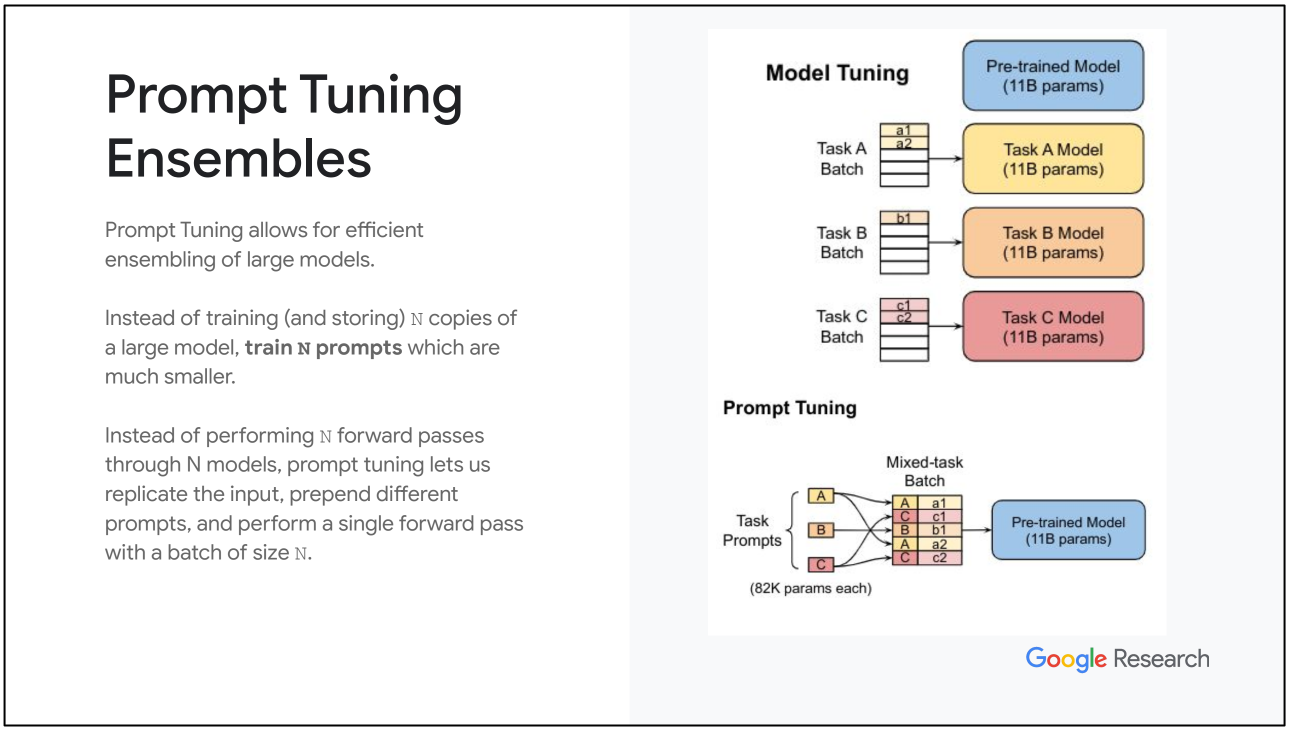
Prompt Tuning has stronger zero-shot performance on datasets with larger domain shifts, including a remarkable +12.5 for TextbookQA.

Dataset	Domain	Model	Prompt	Δ
SQuAD	Wiki	94.9 ±0.2	94.8 ±0.1	−0.1
TextbookQA	Book	54.3 ±3.7	**66.8** ±2.9	+12.5
BioASQ	Bio	77.9 ±0.4	**79.1** ±0.3	+1.2
RACE	Exam	59.8 ±0.6	**60.7** ±0.5	+0.9
RE	Wiki	88.4 ±0.1	**88.8** ±0.2	+0.4
DuoRC	Movie	**68.9** ±0.7	67.7 ±1.1	−1.2
DROP	Wiki	**68.9** ±1.7	67.1 ±1.9	−1.8

Prompt Tuning has stronger zero-shot performance on datasets with larger domain shifts, including a remarkable +12.5 for TextbookQA.

Google Research

- We tested Prompt Tuning’s robustness to domain shift in the context of question answering:
- Used the MRQA 2019 shared task on QA generalization, multiple QA datasets all converted to extractive QA.
- Zero-shot setup:
  - Trained model/prompt on SQuAD
  - Selected the best checkpoint based on the SQuAD development set
  - Evaluated that checkpoint on the other datasets development sets
- Stronger zero-shot, especially when there are larger domain shifts
  - +12.5 for TextbookQA



Prompt Tuning Ensembles

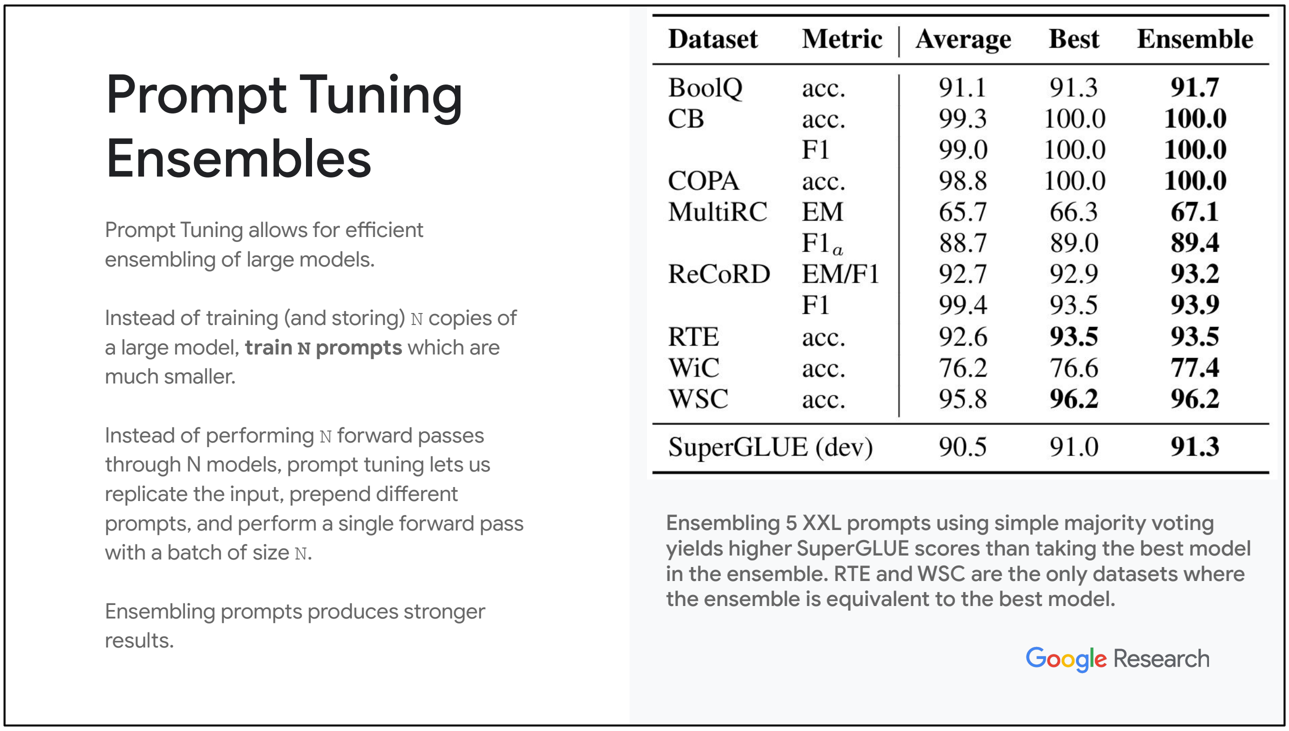
Prompt Tuning allows for efficient ensembling of large models.

Instead of training (and storing) `N` copies of a large model, **train `N` prompts** which are much smaller.

Instead of performing `N` forward passes through N models, prompt tuning lets us replicate the input, prepend different prompts, and perform a single forward pass with a batch of size `N`.

- Prompt Tuning unlocks efficient ensembles for large models
- The same machinery that enable efficient multi-task serving
- Normally:
  - Train N models
  - Run and example though each of N models
    - Need O(N) compute/memory resources to run the ensemble in O(1) (embarrassingly parallel)
    - Need to run the ensemble in O(N) time (plus time to swap models in memory) to use O(1) compute
  - Compute the ensemble prediction
- Prompt Tuning:
  - Train N prompts (much storage requirements)
  - Replicate incoming example N times
  - Add each of the N prompts to one of the replicated inputs
  - Run an single batch of size N through the frozen model
    - O(1) compute/memory to store the model itself.
    - O(1) run time, modulo difference in batch of size 1 vs N
  - Compute ensemble prediction

There is also the possibility of ensembling prompts by combining the actual prompt values.



Prompt Tuning Ensembles

Prompt Tuning allows for efficient ensembling of large models.

Instead of training (and storing) N copies of a large model, **train N prompts** which are much smaller.

Instead of performing N forward passes through N models, prompt tuning lets us replicate the input, prepend different prompts, and perform a single forward pass with a batch of size N.

Ensembling prompts produces stronger results.

Dataset	Metric	Average	Best	Ensemble
BoolQ	acc.	91.1	91.3	**91.7**
CB	acc.	99.3	100.0	**100.0**
	F1	99.0	100.0	**100.0**
COPA	acc.	98.8	100.0	**100.0**
MultiRC	EM	65.7	66.3	**67.1**
	$F1_a$	88.7	89.0	**89.4**
ReCoRD	EM/F1	92.7	92.9	**93.2**
	F1	99.4	93.5	**93.9**
RTE	acc.	92.6	**93.5**	**93.5**
WiC	acc.	76.2	76.6	**77.4**
WSC	acc.	95.8	**96.2**	**96.2**
SuperGLUE (dev)		90.5	91.0	**91.3**

Ensembling 5 XXL prompts using simple majority voting yields higher SuperGLUE scores than taking the best model in the ensemble. RTE and WSC are the only datasets where the ensemble is equivalent to the best model.

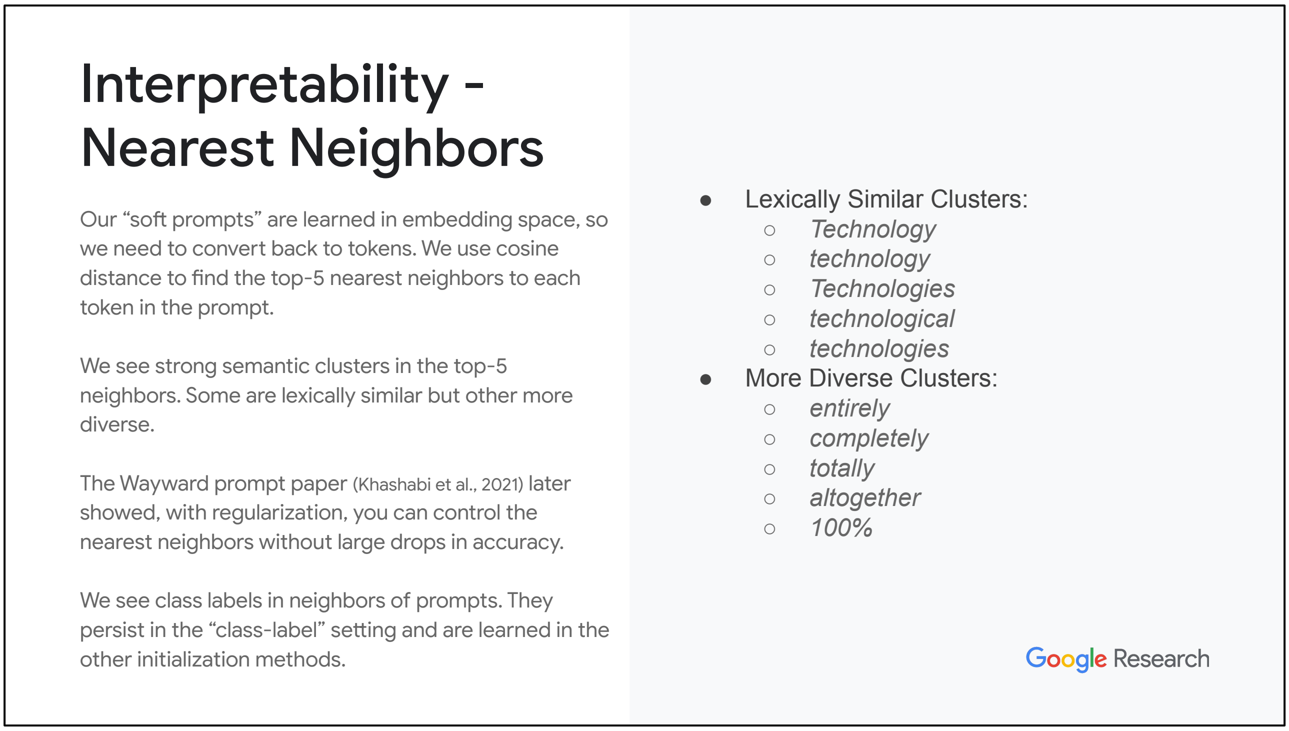
Google Research

Ensembling has strong results than selecting the best model for each task from the ensemble on all datasets.

04

Interpretability and Analysis

Google Research



Interpretability - Nearest Neighbors

Our "soft prompts" are learned in embedding space, so we need to convert back to tokens. We use cosine distance to find the top-5 nearest neighbors to each token in the prompt.

We see strong semantic clusters in the top-5 neighbors. Some are lexically similar but other more diverse.

The Wayward prompt paper (Khashabi et al., 2021) later showed, with regularization, you can control the nearest neighbors without large drops in accuracy.

We see class labels in neighbors of prompts. They persist in the "class-label" setting and are learned in the other initialization methods.

- Lexically Similar Clusters:
  - *Technology*
  - *technology*
  - *Technologies*
  - *technological*
  - *technologies*
- More Diverse Clusters:
  - *entirely*
  - *completely*
  - *totally*
  - *altogether*
  - *100%*

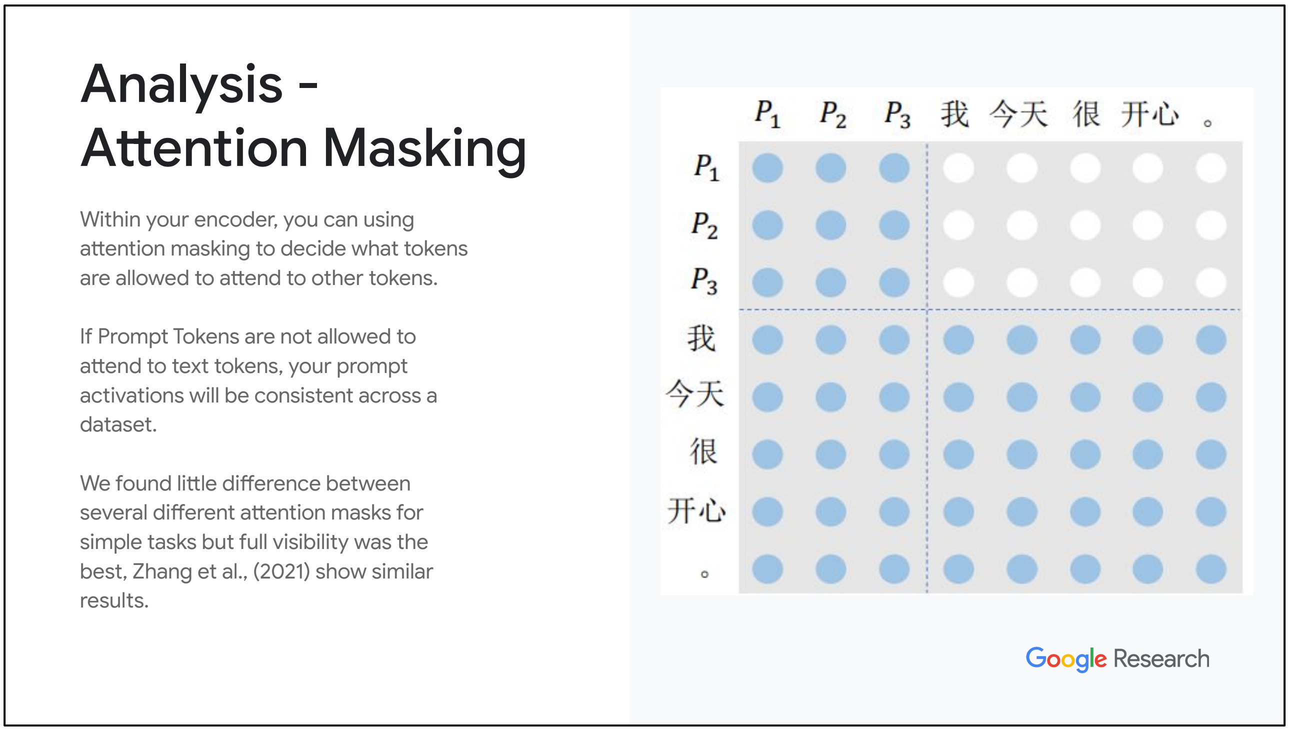
Google Research

- We convert our "soft-prompts" to text by finding the tokens whose embedded representation is closest to a prompt token based on cosine distance.
- Each prompt (size [T, H]) position gets N nearest neighbors
  - A prompt with 10 positions would result in 10 lists of 5 nearest neighbors
- Looking at the clusters:
  - Prompts are very "word-like", even when not initialized from word embeddings.
  - Point out the clusters
  - A random point in the embedding space doesn't tend to have these clusters, these clusters are more akin to adding noise to a pre-existing word
- Class labels are often nearest neighbors
  - The persist when using the class-label initialization.
  - They appear when using other initialization.
    - Class labels appear as NN for many prompt tokens. Possible that the model has trouble localizing the information to a single token. Because relative position information puts a lot of the prompts in the same, max distance, bucket from a lot of the input? Because of Dropout which is applied to the prompt?
- Definitely reading the tea leaves, but many words like "science", "engineering", "technology" in the BoolQ prompt, that dataset has ~20% of questions in the "Nature/Science" category. The prompt could be used to prime the model to think in a specific context?

Analysis - Prompt Discretization

- Turn prompts into the embedding of their nearest neighbors
- Use that as the prompt
- Use that for generation (all zeros?)
- Use that for rank classification?

Google Research

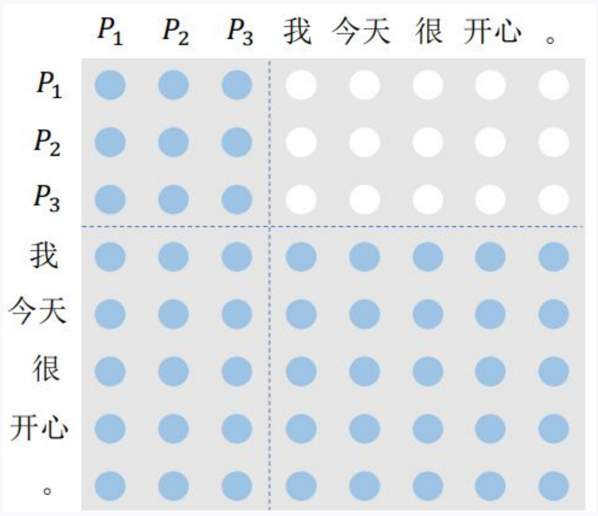


Analysis - Attention Masking

Within your encoder, you can using attention masking to decide what tokens are allowed to attend to other tokens.

If Prompt Tokens are not allowed to attend to text tokens, your prompt activations will be consistent across a dataset.

We found little difference between several different attention masks for simple tasks but full visibility was the best, Zhang et al., (2021) show similar results.

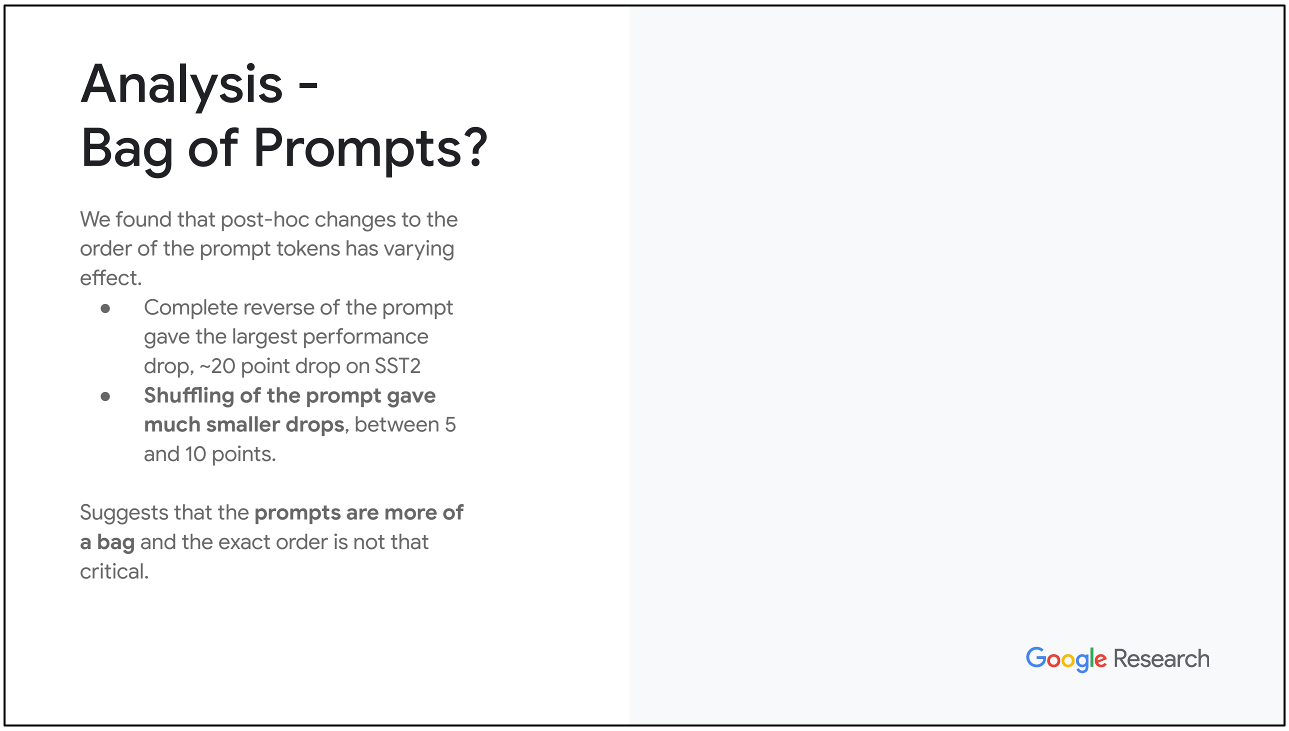


Google Research

- It seems natural that text input should attend to the prompt variables (easiest way for the prompt to influence the computation), but does it make sense for the prompt to attend to the text tokens?
    - You prompt representation becomes dependent on your input. Is this good? Or should the prompt representation through the model remain consistent across all examples in a dataset?
  - Zhang, et al., 2021 attention masking removes Attn(P_q, K, V) and shows large performance losses.

- This is very related to our talk about what prompts on different parts of the input do. Masking the attention is similar to not having that prompt.

Image from CPM-2: Large-scale Cost-effective Pre-trained Language Models (https://arxiv.org/abs/2106.10715)



Analysis - Bag of Prompts?

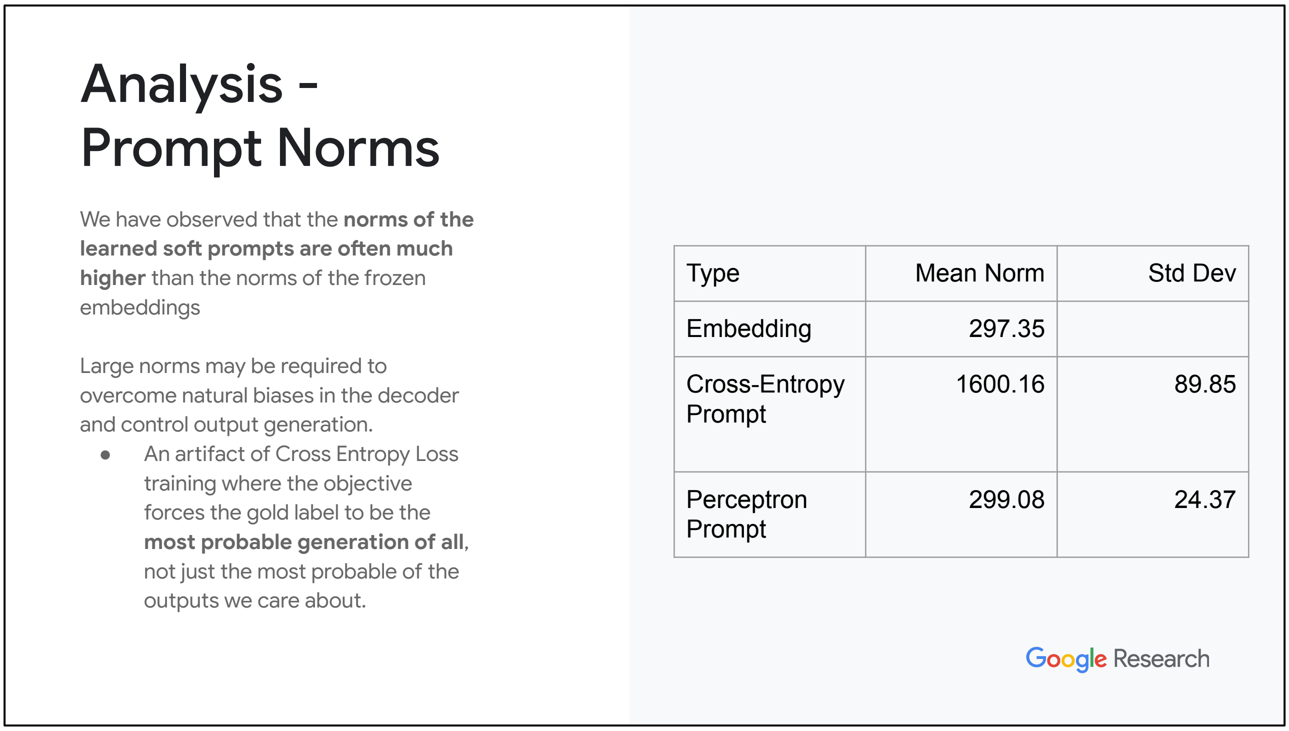
We found that post-hoc changes to the order of the prompt tokens has varying effect.

- Complete reverse of the prompt gave the largest performance drop, ~20 point drop on SST2
- **Shuffling of the prompt gave much smaller drops**, between 5 and 10 points.

Suggests that the **prompts are more of a bag** and the exact order is not that critical.

Google Research

- Part of this may come from the fact t5 uses relative attention and the max bucket length is 128, some prompts are 100 tokens long meaning the majority of the input will see the front section of the prompt as all the same distance away.



Analysis - Prompt Norms

We have observed that the **norms of the learned soft prompts are often much higher** than the norms of the frozen embeddings

Large norms may be required to overcome natural biases in the decoder and control output generation.

- An artifact of Cross Entropy Loss training where the objective forces the gold label to be the **most probable generation of all**, not just the most probable of the outputs we care about.

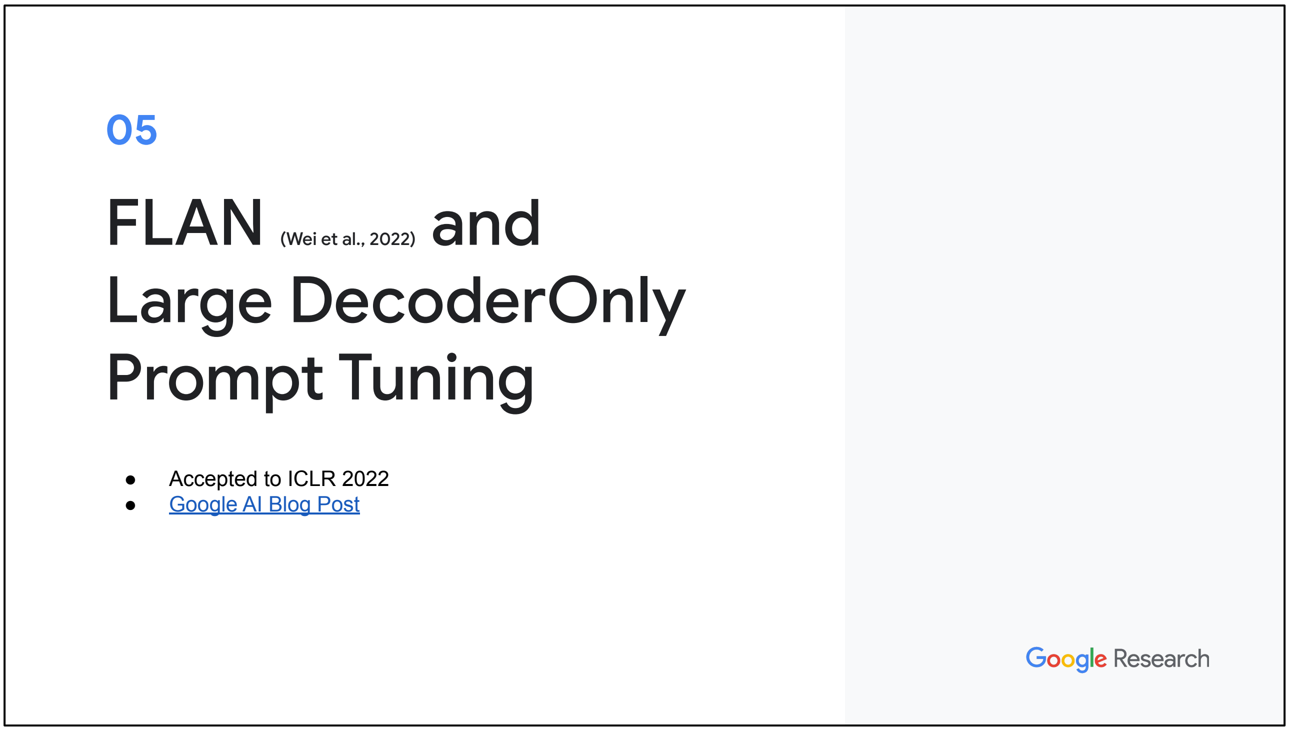
Type	Mean Norm	Std Dev
Embedding	297.35	
Cross-Entropy Prompt	1600.16	89.85
Perceptron Prompt	299.08	24.37

Google Research

- The norms of the learned prompt tokens are often much higher than the embedded words.
- We think this may have to do with the prompts needing to make large changes in attention scores.
- We think this may be because of training with the token-level cross-entropy loss.
  - This makes us want to control the model output, not just rank the gold class above others (this second approach only works on datasets with known label outputs).
  - When we train using a ranking loss (Perceptron in this case) we see much smaller norms.
    - When training like this we don't have to worry about what the model actually outputs.

Analysis -
Prompt Removal

Google Research



05

FLAN (Wei et al., 2022) and Large DecoderOnly Prompt Tuning

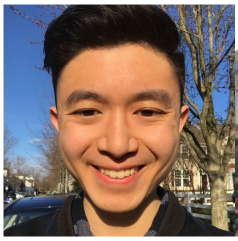
- Accepted to ICLR 2022
- Google AI Blog Post

Google Research

The next part of my work revolved around using prompt tuning with large decoder only language models.

FLAN was accepted to ICLR 2022!

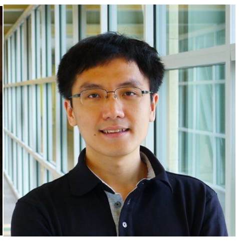
Collaborators


Jason Wei

Maarten Bosma

Vincent Y Zhao

Kelvin Guu


Adams Wei Yu

Me Lol

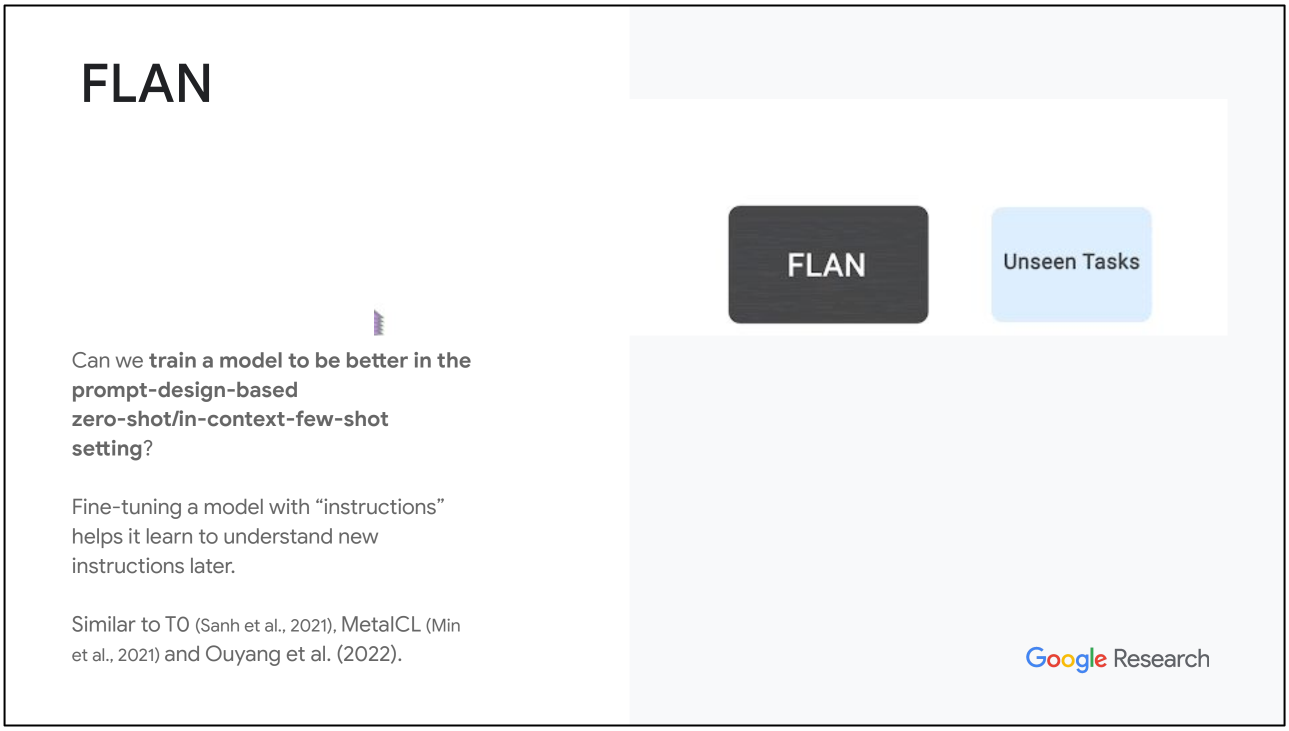
Nan Du

Andrew M. Dai

Quoc V. Le

Google Research

Jason Wei is also an AI Resident



FLAN

Can we **train a model to be better in the prompt-design-based zero-shot/in-context-few-shot setting**?

Fine-tuning a model with "instructions" helps it learn to understand new instructions later.

Similar to T0 (Sanh et al., 2021), MetalCL (Min et al., 2021) and Ouyang et al. (2022).

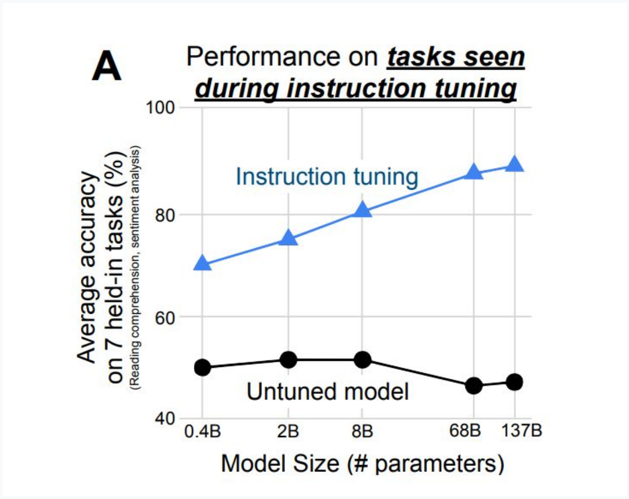
FLAN
Unseen Tasks
Google Research

- Our base model is a 137B parameter dense transformer based language model, trained on a mix of web, code, and conversational data.
- Few/Zero-Shot performance "just fell out" of GPT-3, it happened to work, especially for tasks that could essentially be rewritten as LM completion
- Can we steer the model to be good at this instead of just hoping?

FLAN - Power of Scale

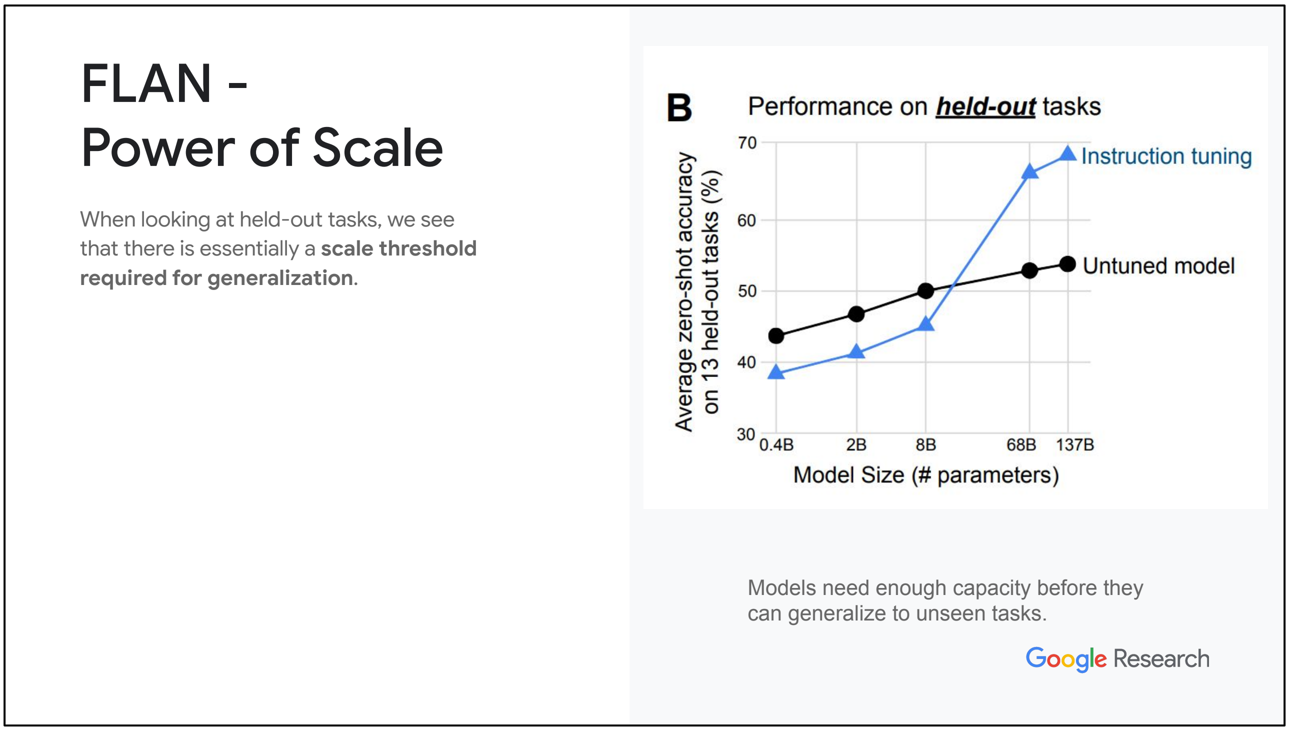
As we scale a model, we expect it to improve.

Here we see testing on the tasks we use in training continues to go up with scale.



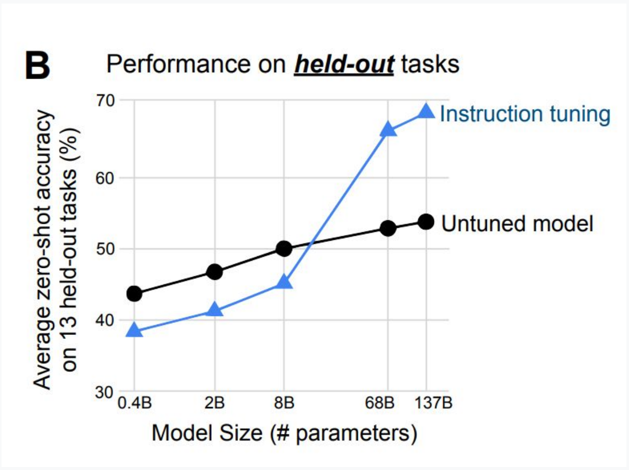
Google Research

- As expected, larger models do better
- The untuned models have a slight dip when it hits larger scales because these datasets are not super aligned to LM modeling so they tend to have a lot of variance.



FLAN - Power of Scale

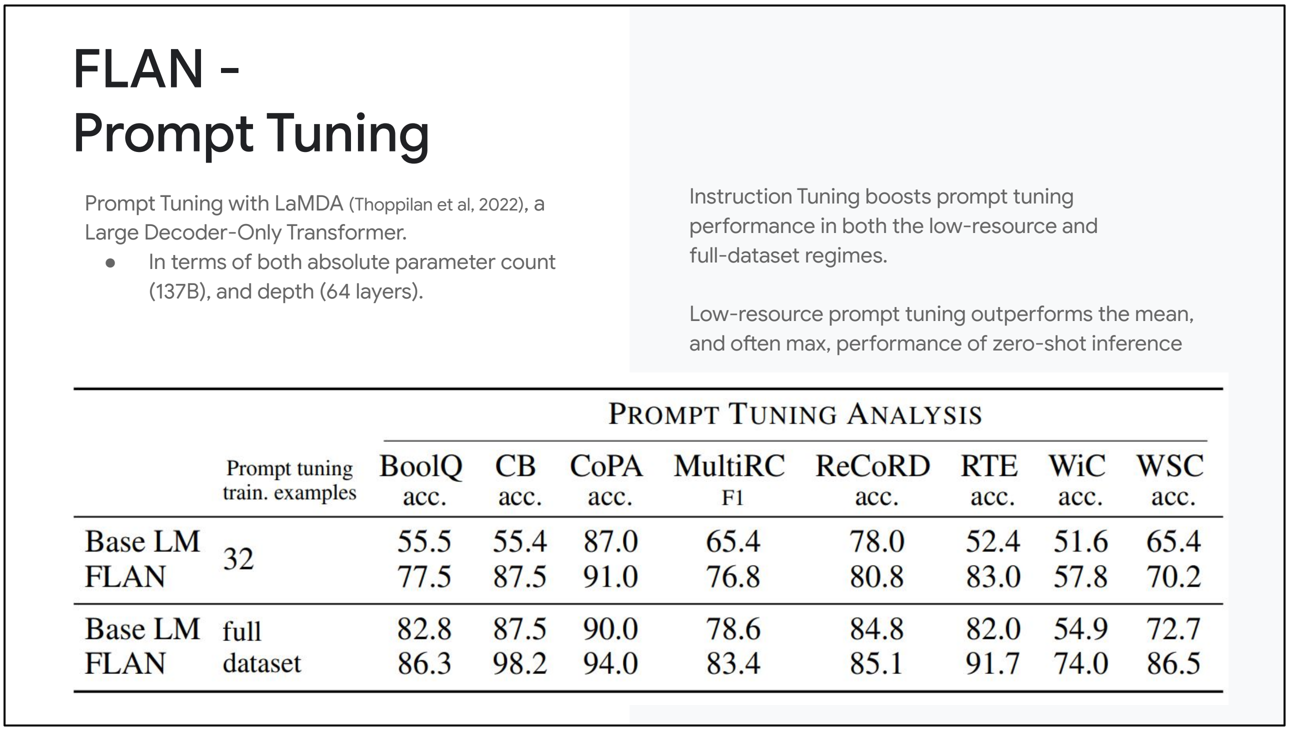
When looking at held-out tasks, we see that there is essentially a **scale threshold required for generalization**.



Models need enough capacity before they can generalize to unseen tasks.

Google Research

- Echoing our findings in "The Power of Scale for Parameter-Efficient Prompt Tuning" there is a scale threshold for generalization of strong Few/Zero-Shot performance on held out tasks.



FLAN - Prompt Tuning

Prompt Tuning with LaMDA (Thoppilan et al, 2022), a Large Decoder-Only Transformer.

- In terms of both absolute parameter count (137B), and depth (64 layers).

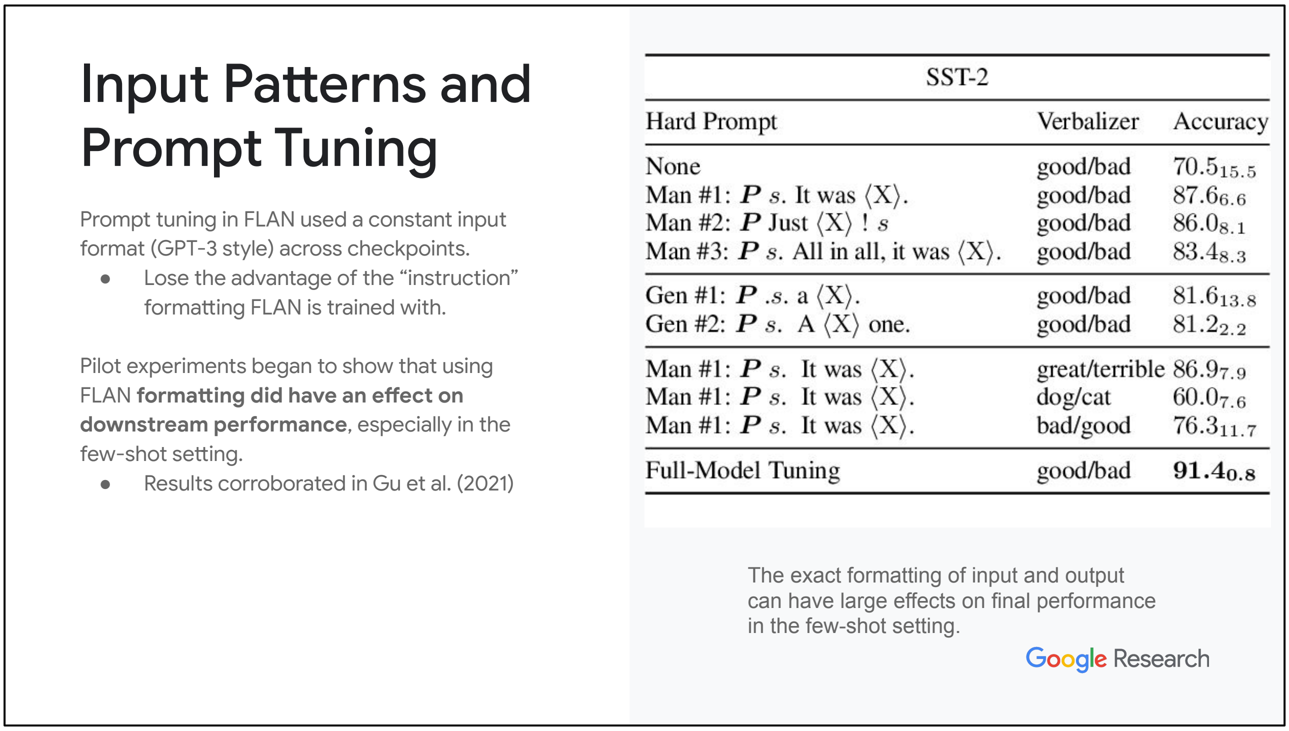
Instruction Tuning boosts prompt tuning performance in both the low-resource and full-dataset regimes.

Low-resource prompt tuning outperforms the mean, and often max, performance of zero-shot inference

PROMPT TUNING ANALYSIS

	Prompt tuning train. examples	BoolQ acc.	CB acc.	CoPA acc.	MultiRC F1	ReCoRD acc.	RTE acc.	WiC acc.	WSC acc.
Base LM	32	55.5	55.4	87.0	65.4	78.0	52.4	51.6	65.4
FLAN		77.5	87.5	91.0	76.8	80.8	83.0	57.8	70.2
Base LM	full dataset	82.8	87.5	90.0	78.6	84.8	82.0	54.9	72.7
FLAN		86.3	98.2	94.0	83.4	85.1	91.7	74.0	86.5

- Prompt Tuning as part of the FLAN paper was out first chance to try it in the context of a DecoderOnly LM, all work with T5 uses EncoderDecoder models where the prompt was on the Encoder.
    - This was also the first time we used Rank Classification, where the Language Model scores over set of possible labels are calculated and used instead of actually generating a label.
- The FLAN instruction tuning procedure creates a stronger NLP model in addition to allowing it to deal with Instruction formatted inputs.
- Low-Resource Prompt Tuning with the FLAN model outperforms the mean, and often max, performance of the Zero-Shot and Multi-Shot FLAN models, even when using non-optimal (the GPT-3 style) input formatting.



Input Patterns and Prompt Tuning

Prompt tuning in FLAN used a constant input format (GPT-3 style) across checkpoints.

- Lose the advantage of the “instruction” formatting FLAN is trained with.

Pilot experiments began to show that using FLAN **formatting did have an effect on downstream performance**, especially in the few-shot setting.

- Results corroborated in Gu et al. (2021)

SST-2		
Hard Prompt	Verbalizer	Accuracy
None	good/bad	$70.5_{15.5}$
Man #1: $\boldsymbol{P}$ s. It was ⟨X⟩.	good/bad	$87.6_{6.6}$
Man #2: $\boldsymbol{P}$ Just ⟨X⟩ ! s	good/bad	$86.0_{8.1}$
Man #3: $\boldsymbol{P}$ s. All in all, it was ⟨X⟩.	good/bad	$83.4_{8.3}$
Gen #1: $\boldsymbol{P}$ .s. a ⟨X⟩.	good/bad	$81.6_{13.8}$
Gen #2: $\boldsymbol{P}$ s. A ⟨X⟩ one.	good/bad	$81.2_{2.2}$
Man #1: $\boldsymbol{P}$ s. It was ⟨X⟩.	great/terrible	$86.9_{7.9}$
Man #1: $\boldsymbol{P}$ s. It was ⟨X⟩.	dog/cat	$60.0_{7.6}$
Man #1: $\boldsymbol{P}$ s. It was ⟨X⟩.	bad/good	$76.3_{11.7}$
Full-Model Tuning	good/bad	$\mathbf{91.4_{0.8}}$

The exact formatting of input and output can have large effects on final performance in the few-shot setting.

Google Research

- FLAN comparison used the LM-focused GPT-3 style prompts, even though FLAN was tuned with a more “instruction” focused prompts. How does this effect results? We saw that the learnable nature of Prompt Tuning does make up for a lot of this, but it could still be leaving some performance on the floor.
- Similar to results from combining Prompt Design and Fine Tuning (Schick and Schütze, 2021; Le Scao and Rush, 2021; Webson and Pavlick, 2021) where the input formatting can help learnability.

06

SPoT: Soft Prompt Transfer (Vu et al., 2021)

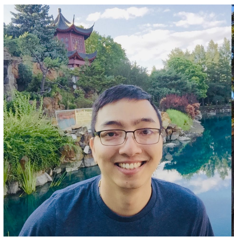
- Accepted to ACL 2022

Google Research

Now some work on increasing the performance of Prompt Tuning and some interesting applications you can do with it.

Accepted to ACL 2022

Collaborators


Tu Vu

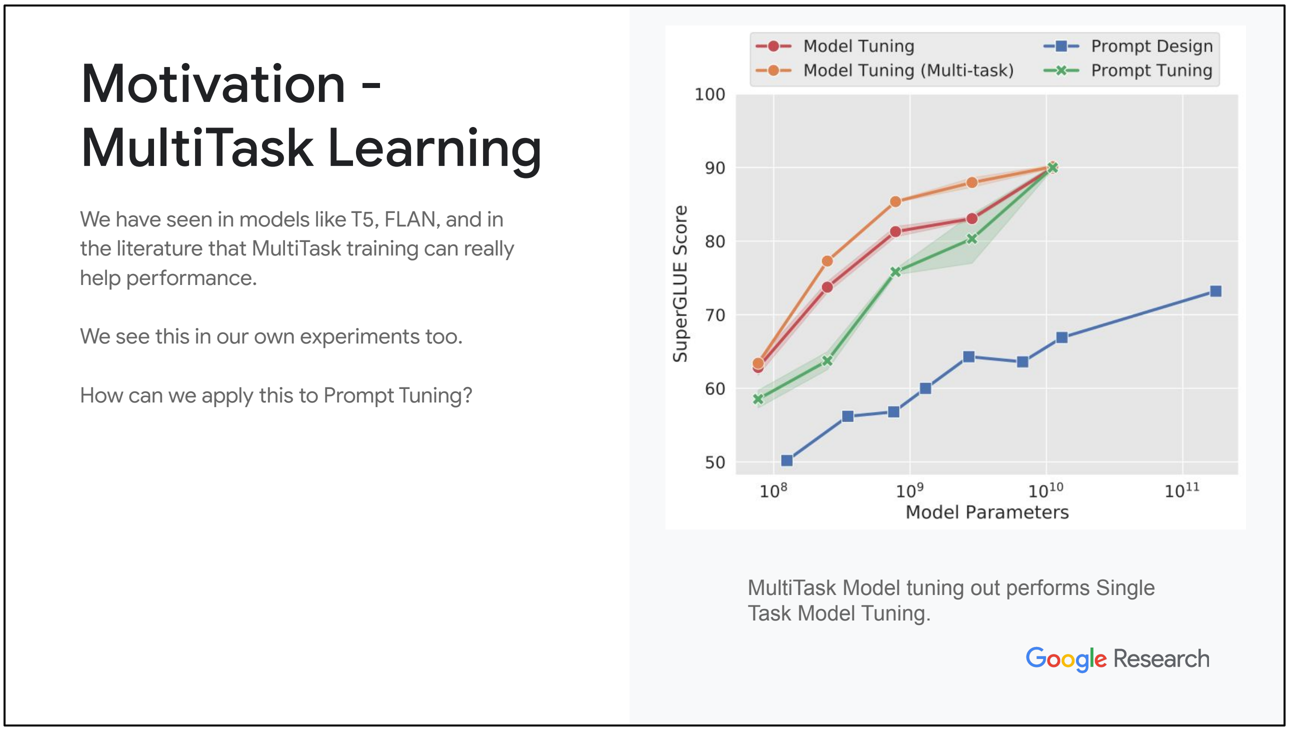
Me lol

Noah Constant

Rami Al-Rfou

Daniel Cer

Google Research

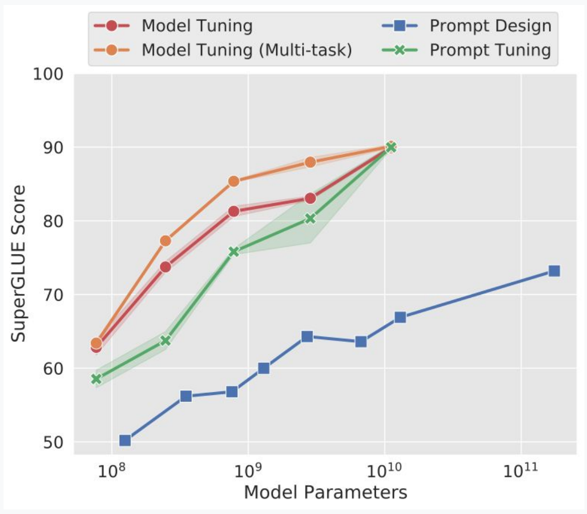


Motivation - MultiTask Learning

We have seen in models like T5, FLAN, and in the literature that MultiTask training can really help performance.

We see this in our own experiments too.

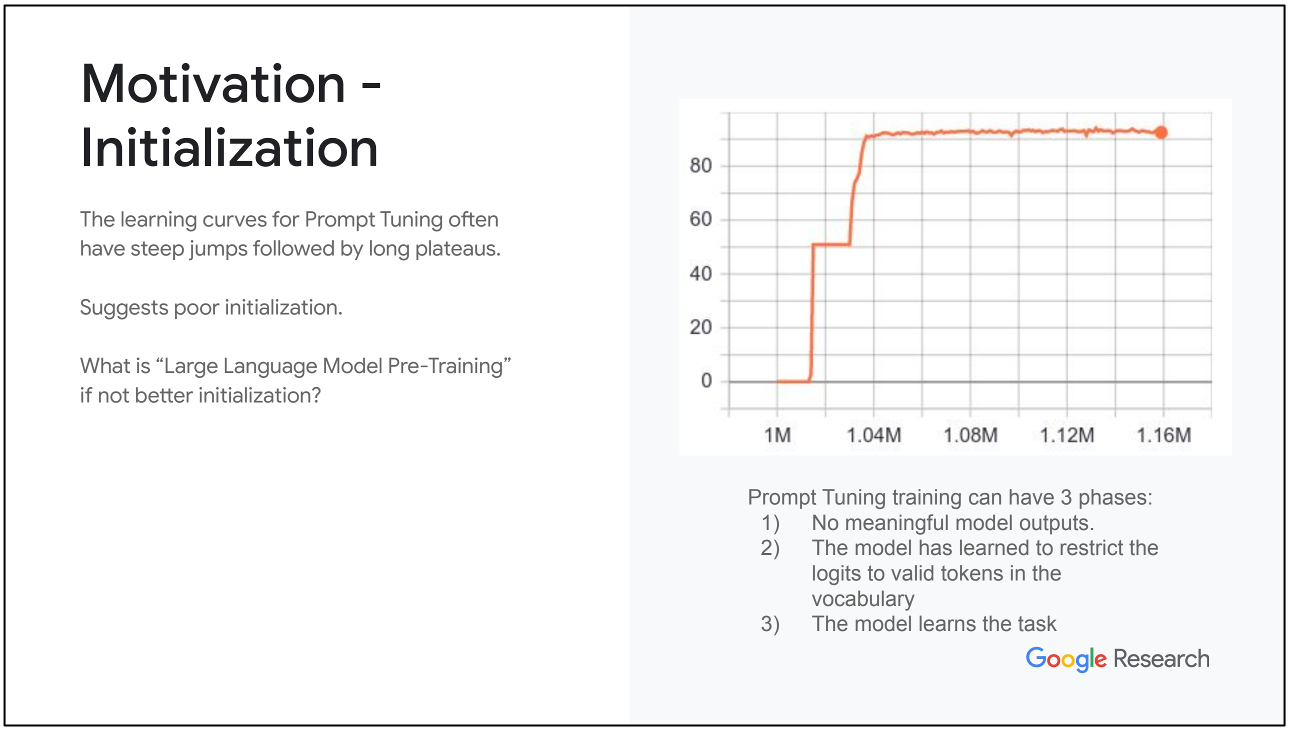
How can we apply this to Prompt Tuning?



MultiTask Model tuning out performs Single Task Model Tuning.

Google Research

- We have seen that multi-task training or multi-task pretraining can help boost model performance.
    - We even see that in our original baselines with Model Tuning
- How can we use this for prompt tuning?
    - Especially when the limited capacity of the prompt makes putting multiple tasks in the prompt hard.

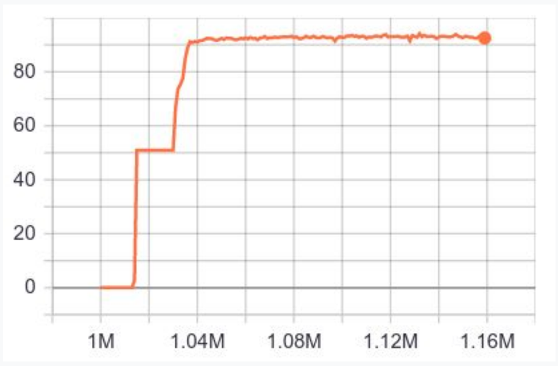


Motivation - Initialization

The learning curves for Prompt Tuning often have steep jumps followed by long plateaus.

Suggests poor initialization.

What is “Large Language Model Pre-Training” if not better initialization?

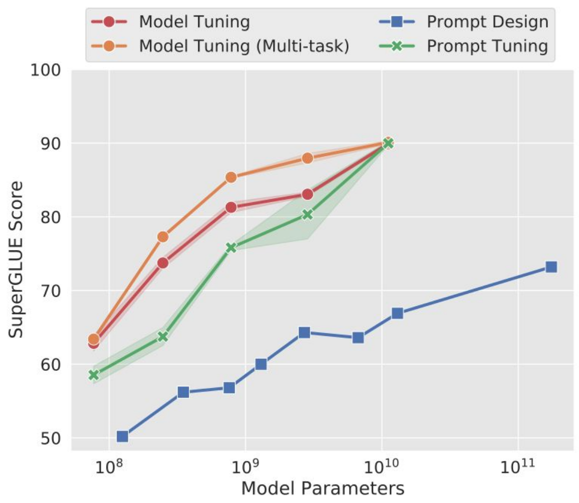


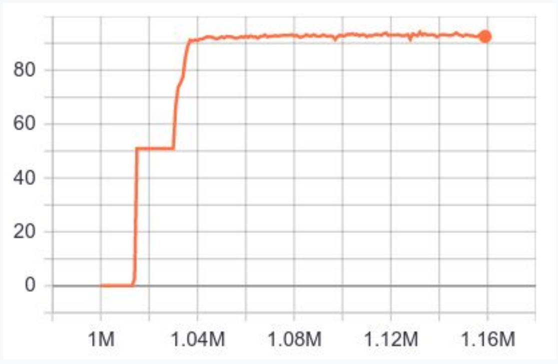
Prompt Tuning training can have 3 phases:
1) No meaningful model outputs.
2) The model has learned to restrict the logits to valid tokens in the vocabulary
3) The model learns the task

Google Research

- Often when training prompts you can end up with a bit of unhealthy looking learning curves.
    - You start with the prompt being unable to control the model output at all, it outputs junk or nothing.
    - Then it jumps to restricting the model outputs to just the valid classes—generally, the most common label.
    - Then it jumps to solving the task, with slow growth after that.
    - This is especially common with randomly initialized prompts.
- One of our collaborators (Dan) mentioned this is a bit like learning curves from the 80s, when we didn’t know how to initialize models well.
    - Suggests poor prompt Initialization
        - Pre-training is at it core better initialization.

SPoT Motivation - MultiTask Learning



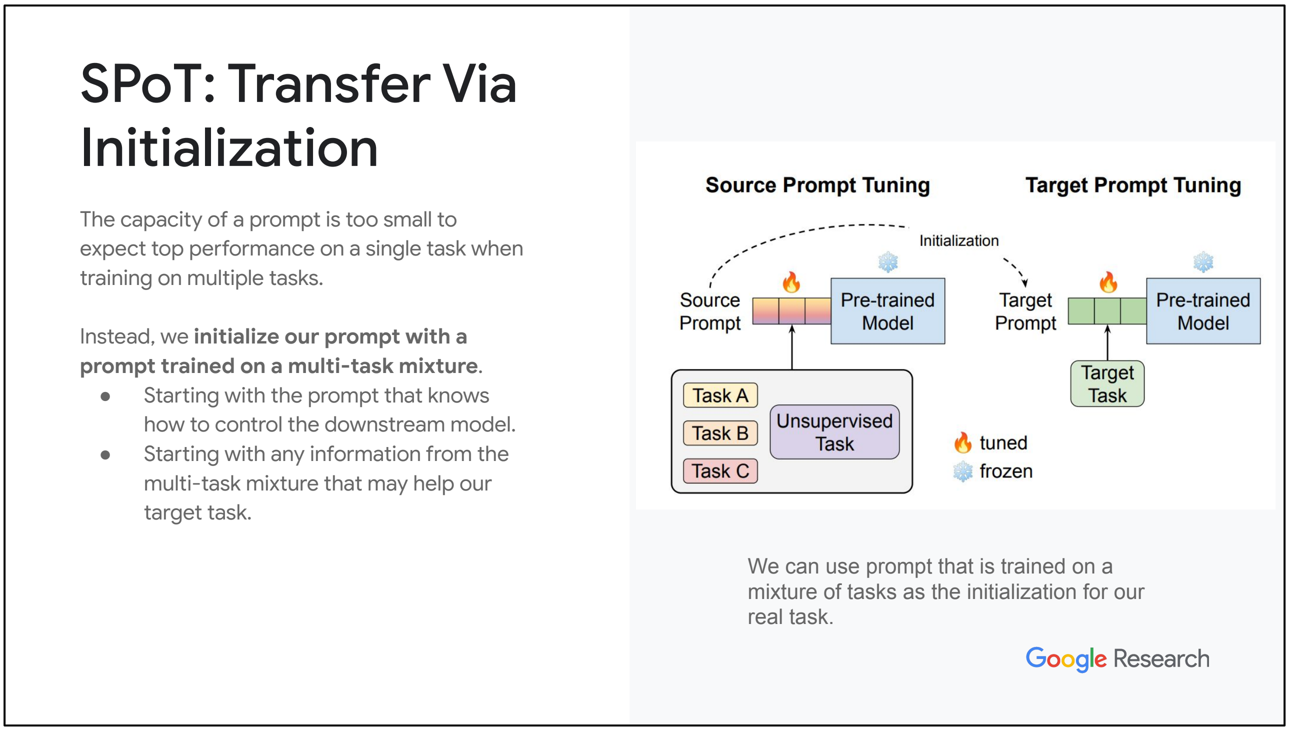


SPoT Motivation - Initialization

Google Research

- We have seen that multi-task training or multi-task pretraining can help boost model performance.
  - We even see that in our original baselines with Model Tuning
- How can we use this for prompt tuning?
  - Especially when the limited capacity of the prompt makes putting multiple tasks in the prompt hard.

- Often when training prompts you can end up with a bit of unhealthy looking learning curves.
  - You start with the prompt being unable to control the model output at all, it outputs junk or nothing.
  - Then it jumps to restricting the model outputs to just the valid classes—generally, the most common label.
  - Then it jumps to solving the task, with slow growth after that.
  - This is especially common with randomly initialized prompts.
- One of our collaborators (Dan) mentioned this is a bit like learning curves from the 80s, when we didn't know how to initialize models well.
  - Suggests poor prompt Initialization
    - Pre-training is at it core better initialization.

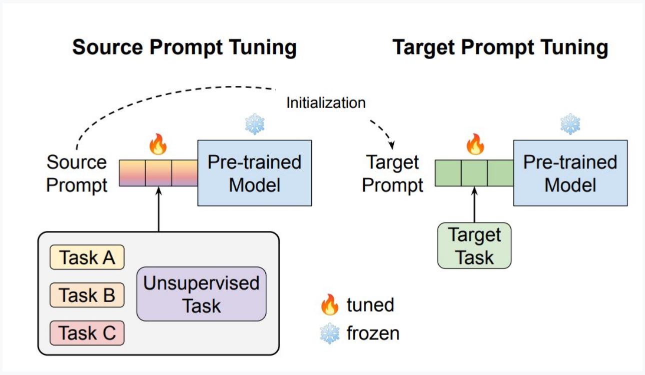


SPoT: Transfer Via Initialization

The capacity of a prompt is too small to expect top performance on a single task when training on multiple tasks.

Instead, we **initialize our prompt with a prompt trained on a multi-task mixture**.

- Starting with the prompt that knows how to control the downstream model.
- Starting with any information from the multi-task mixture that may help our target task.



We can use prompt that is trained on a mixture of tasks as the initialization for our real task.

Google Research

- SPoT: **S**oft **P**r**o**mpt **T**ransfer via Initialization.
- General/Manually targeted Pretraining:
  - Train a Prompt on a large mixture of tasks.
  - Use that prompt as the initialization for our target task.
  - Experimentally find mixtures that work well.

SPoT: Transfer Via Initialization

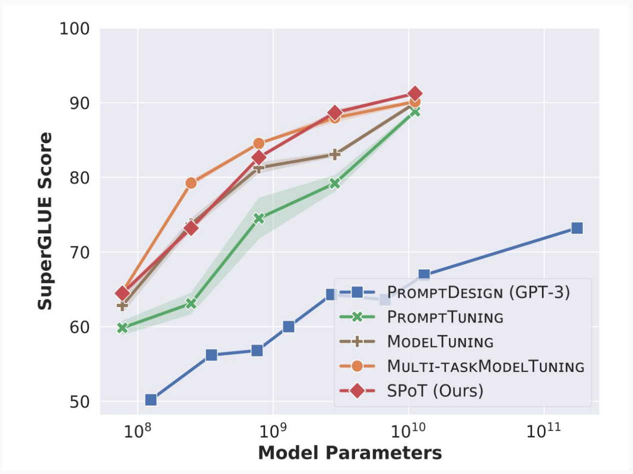
We **no longer need scale** to rival Model Tuning.

Trained a Prompt on a Mixture of all the GLUE (Wang et al., 2018) tasks

Then trained individual prompts for each SuperGLUE task.

- Each prompt is initialized with the GLUE mixture prompt at the start of training.

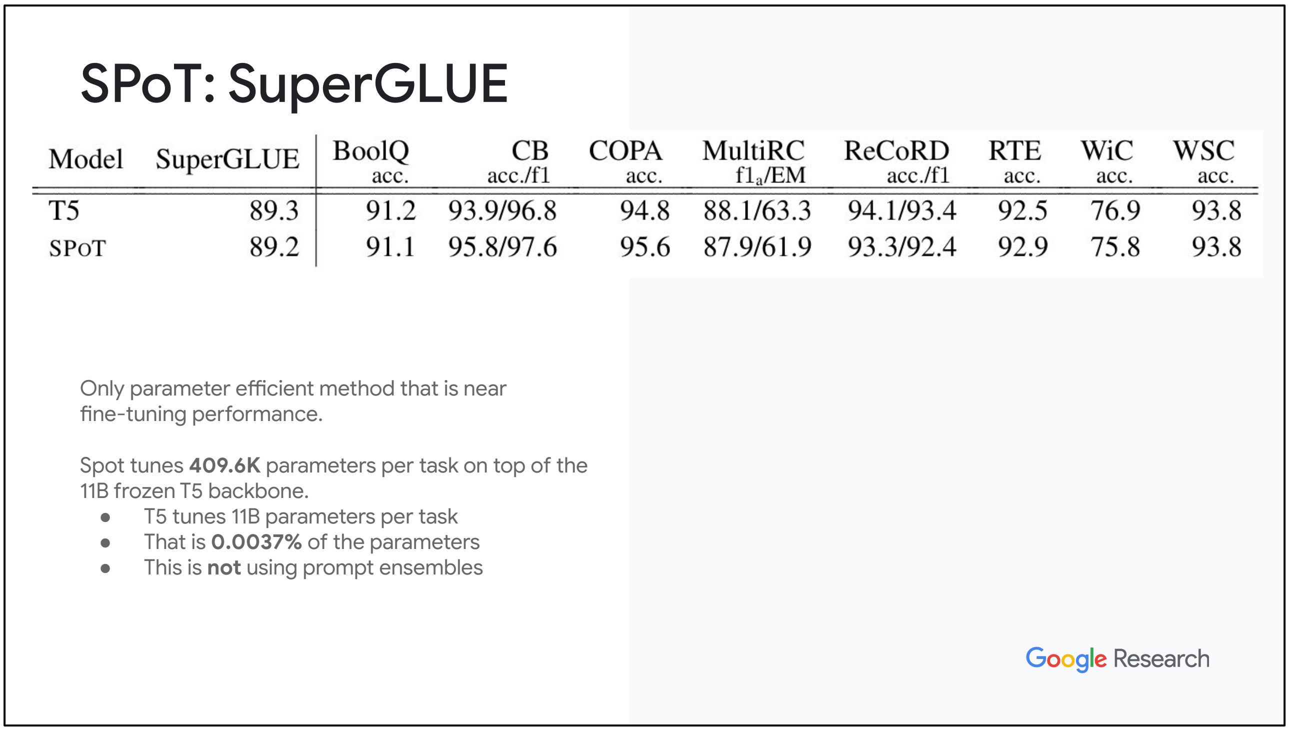
A large increase (1.1 points) in performance when using the XXL model.



SPoT transfer increases model performance across sizes until it is competitive with Model Tuning with smaller models and much stronger at size XXL.

Google Research

- For strong SuperGLUE performance we transferred a prompt from GLUE.
- We trained one Prompt on a mixture of GLUE tasks.
- We then trained a new prompt for each SuperGLUE task
  - Initialized with the GLUE multi-task prompt.
- We catch up to Model Tuning at smaller model scales.
- There is a large performance gain at the XXL size.



SPoT: SuperGLUE

Model	SuperGLUE	BoolQ acc.	CB acc./f1	COPA acc.	MultiRC $f1_a$/EM	ReCoRD acc./f1	RTE acc.	WiC acc.	WSC acc.
T5	89.3	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8
SPoT	89.2	91.1	95.8/97.6	95.6	87.9/61.9	93.3/92.4	92.9	75.8	93.8

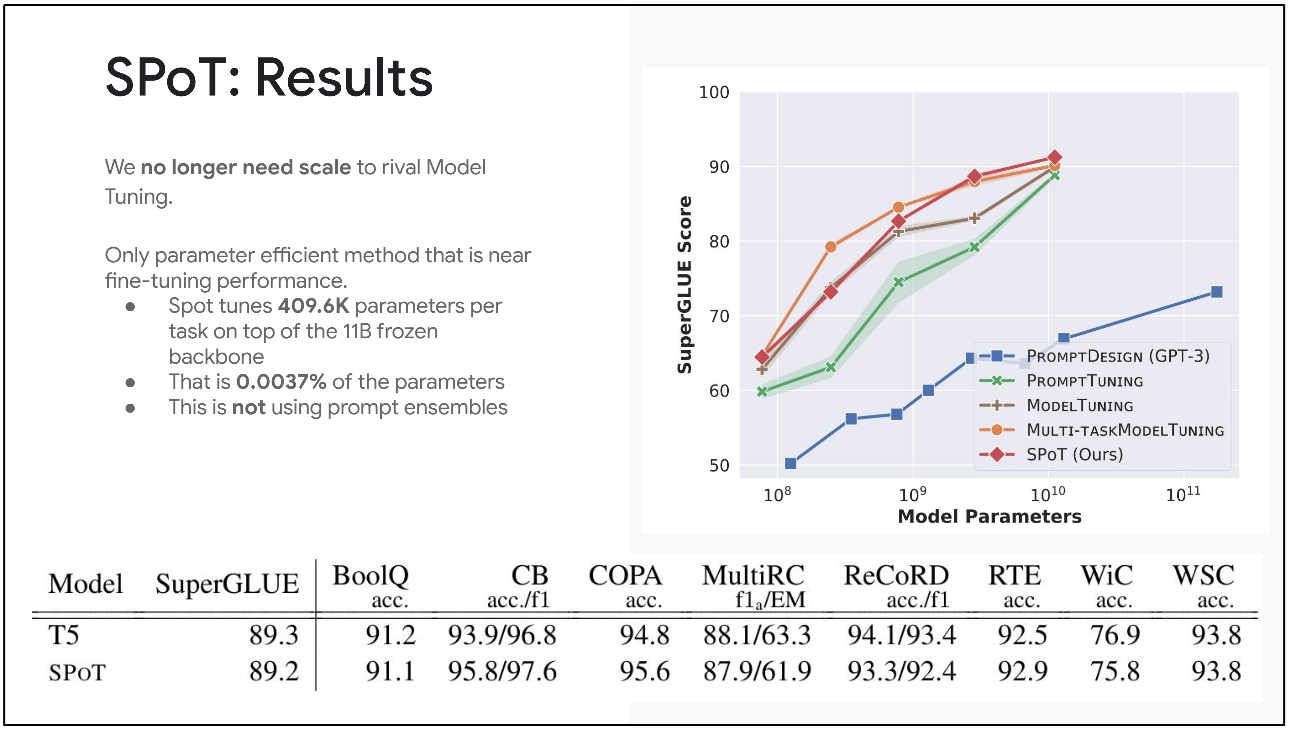
Only parameter efficient method that is near fine-tuning performance.

Spot tunes **409.6K** parameters per task on top of the 11B frozen T5 backbone.

- T5 tunes 11B parameters per task
- That is **0.0037%** of the parameters
- This is **not** using prompt ensembles

Google Research

- SPoT gave a really strong SuperGLUE performance
- We are only .1 points behind T5 with model tuning.
- This is really a whole different ball game when it comes to parameter efficiency (0.0037%)
    - Many other methods that are “parameter-efficient” add around 3% of parameters.
    - Again, 3% of the parameters for T5-XXL is basically a whole BERT-Large (330M vs 340M).

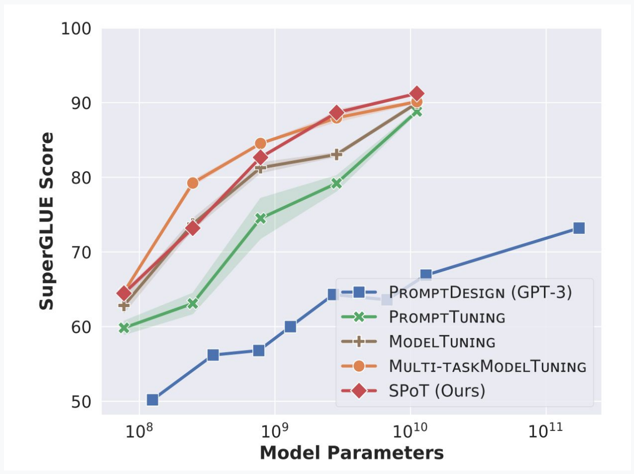


SPoT: Results

We **no longer need scale** to rival Model Tuning.

Only parameter efficient method that is near fine-tuning performance.

- Spot tunes **409.6K** parameters per task on top of the 11B frozen backbone
- That is **0.0037%** of the parameters
- This is **not** using prompt ensembles



Model	SuperGLUE	BoolQ acc.	CB acc./f1	COPA acc.	MultiRC $f1_a$/EM	ReCoRD acc./f1	RTE acc.	WiC acc.	WSC acc.
T5	89.3	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8
SPoT	89.2	91.1	95.8/97.6	95.6	87.9/61.9	93.3/92.4	92.9	75.8	93.8

- For strong SuperGLUE performance we transferred a prompt from GLUE.
- We trained one Prompt on a mixture of GLUE tasks.
- We then trained a new prompt for each SuperGLUE task
  - Initialized with the GLUE multi-task prompt.
- We catch up to Model Tuning at smaller model scales.
- There is a large performance gain at the XXL size.

- SPoT gave a really strong SuperGLUE performance
- We are only .1 points behind T5 with model tuning.
- This is really a whole different ball game when it comes to parameter efficiency (0.0037%)
  - Many other methods that are “parameter-efficient” add around 3% of parameters.
  - Again, 3% of the parameters for T5-XXL is basically a whole BERT-Large (330M vs 340M).

SPoT: Targeted Transfer via Prompt Similarity

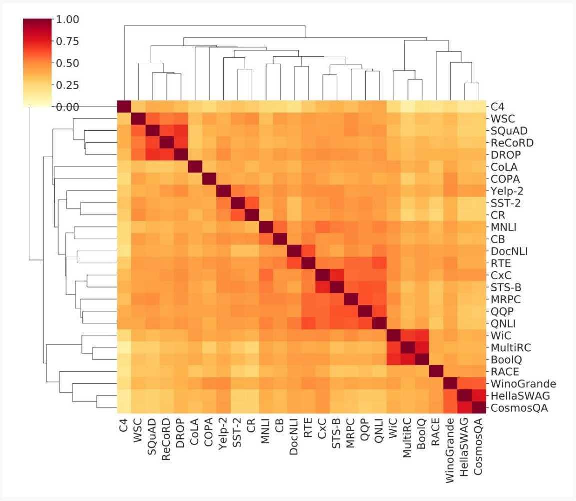
A model is some location in a parameter space that solves some task

We expect models that are close in that space solve similar tasks.

- Large models like t5 are hard to compare.

Prompts have already shown “word-like” behavior.

- Prompt Similarity can be a proxy for what prompts solve similar tasks, and therefore which tasks are similar.



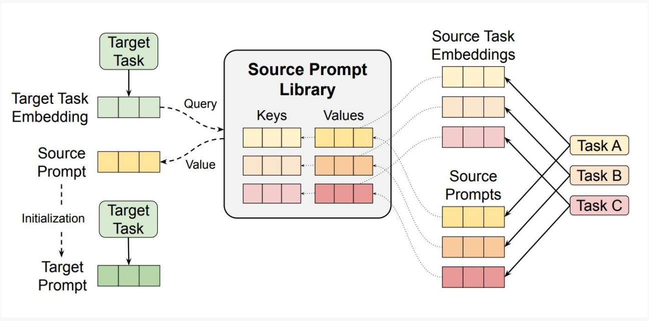
Prompt Similarity shows reasonable task clusters. Clusters seem to be more related to task than domain as QNLI is very dissimilar to SQuAD, despite them being built on the same dataset.

Google Research

CxC and STS-B are actually the same task, collected with the same method, by our collaborator Daniel Cer. The model found the cofounder of Dan

SPoT: Targeted Transfer via Prompt Similarity

1. Build a bank of Source tasks
   - We found having “Key” and “Value” versions of the prompts worked best.
   - “Keys” are prompts trained for only a few thousand steps.
   - “Values” are prompts are the prompts the performed best on the dev set.
2. Train a “query” prompt on your target task.
3. Find the best source task via query and key similarity.
4. Initialize with that source task’s value and train on the target task.



Instead of empirically finding which tasks transfer the best to other tasks, we can use the similarity between prompts to select the task to use as an initialization point.

Google Research

SPoT:
Targeted Transfer via Prompt Similarity

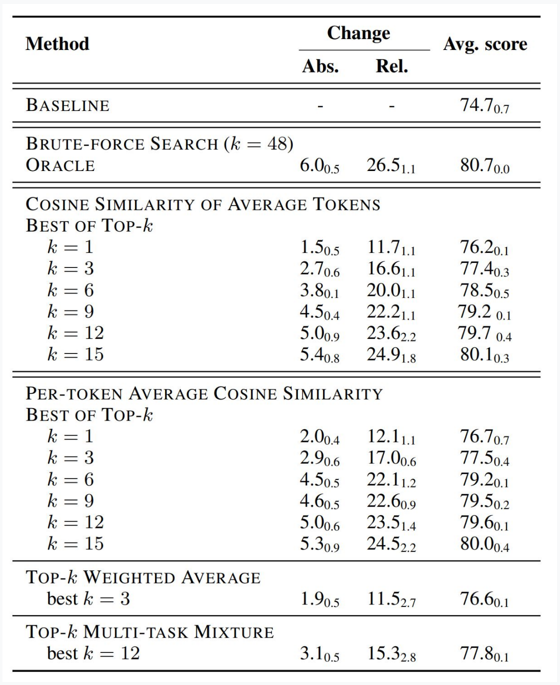
Trained 16 source tasks (3 different prompts with different seeds) and trained 10 target tasks prompts

Selected the top-K source tasks based on prompt similarity

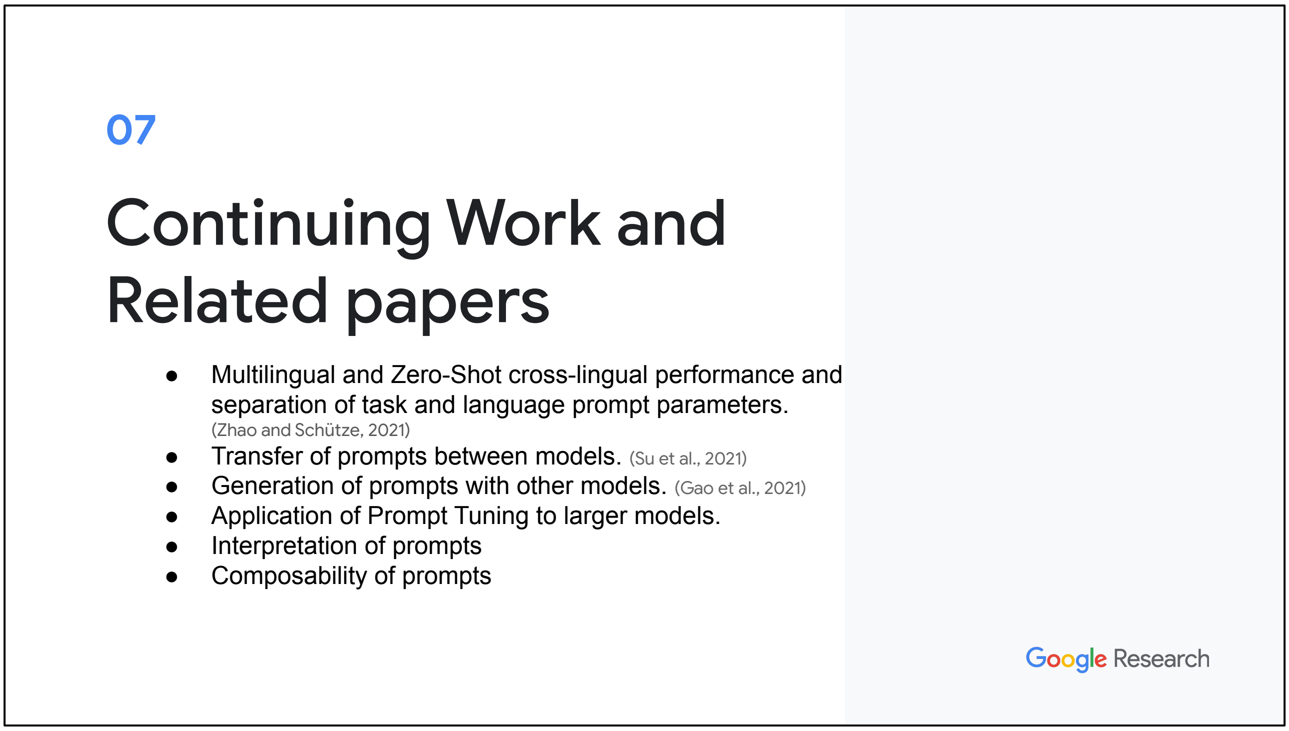
Trained on the target task, initialized with one of the top-K source prompts.

Results:

- Training with **only the top-3 tasks**, we already recover over **half of the oracle performance**
- K=15 recovers almost all of the performance



Method	Change		Avg. score
	Abs.	Rel.	
BASELINE	-	-	$74.7_{0.7}$
BRUTE-FORCE SEARCH ($k = 48$)			
ORACLE	$6.0_{0.5}$	$26.5_{1.1}$	$80.7_{0.0}$
COSINE SIMILARITY OF AVERAGE TOKENS			
BEST OF TOP-k			
$k = 1$	$1.5_{0.5}$	$11.7_{1.1}$	$76.2_{0.1}$
$k = 3$	$2.7_{0.6}$	$16.6_{1.1}$	$77.4_{0.3}$
$k = 6$	$3.8_{0.1}$	$20.0_{1.1}$	$78.5_{0.5}$
$k = 9$	$4.5_{0.4}$	$22.2_{1.1}$	$79.2_{0.1}$
$k = 12$	$5.0_{0.9}$	$23.6_{2.2}$	$79.7_{0.4}$
$k = 15$	$5.4_{0.8}$	$24.9_{1.8}$	$80.1_{0.3}$
PER-TOKEN AVERAGE COSINE SIMILARITY			
BEST OF TOP-k			
$k = 1$	$2.0_{0.4}$	$12.1_{1.1}$	$76.7_{0.7}$
$k = 3$	$2.9_{0.6}$	$17.0_{0.6}$	$77.5_{0.4}$
$k = 6$	$4.5_{0.5}$	$22.1_{1.2}$	$79.2_{0.1}$
$k = 9$	$4.6_{0.5}$	$22.6_{0.9}$	$79.5_{0.2}$
$k = 12$	$5.0_{0.6}$	$23.5_{1.4}$	$79.6_{0.1}$
$k = 15$	$5.3_{0.9}$	$24.5_{2.2}$	$80.0_{0.4}$
TOP-k WEIGHTED AVERAGE			
best $k = 3$	$1.9_{0.5}$	$11.5_{2.7}$	$76.6_{0.1}$
TOP-k MULTI-TASK MIXTURE			
best $k = 12$	$3.1_{0.5}$	$15.3_{2.8}$	$77.8_{0.1}$


07

Continuing Work and Related papers

- Multilingual and Zero-Shot cross-lingual performance and separation of task and language prompt parameters. (Zhao and Schütze, 2021)
- Transfer of prompts between models. (Su et al., 2021)
- Generation of prompts with other models. (Gao et al., 2021)
- Application of Prompt Tuning to larger models.
- Interpretation of prompts
- Composability of prompts

Google Research

Some examples of the current projects we are working on. I've also included a few related papers where if you read that one and then thing "prompt tuning" you should be able to get a clear idea of what we are doing.

- Using mT5, and tasks where you train in one language and then apply the model to the same task in a different language.
- Can you use a previous prompt as a starting point when you refresh a model, or move them between different sizes.
- Can you generate a real-valued prompt from a snapshot of a dataset and can that generalize to new datasets.
- Bigger is Better?

References

- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, et al. 2020. Language Models Are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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), pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.
- Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. PPT: Pre-Trained Prompt Tuning for Few-Shot Learning. *arXiv:2109.04332 [cs]*, September.
- Tianyu Gao, Adam Fisch and Danqi Chen, 2021. Making Pre-trained Language Models Better Few-shot Learners. 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)*, pages 3816–3830, Online, August. Association for Computational Linguistics
- Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-Level Adversarial ReProgramming. 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)*, pages 4921–4933, Online, August. Association for Computational Linguistics.
- Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick and Graham Neubig. 2021. Towards a Unified View of Parameter-Efficient Transfer Learning. *ArXiv:2110.04366*, October.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2790–2799. PMLR, June.
- Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics*, 8:423–438.
- Daniel Khashabi, Shane Lyu, Sewon Min, Lianhui Qin, Kyle Richardson, Sameer Singh, Sean Welleck, Hannaneh Hajishirzi, Tushar Khot, Ashish Sabharwal and Yejin Choi. 2021. PROMPT WAYWARDNESS: The Curious Case of Discretized Interpretation of Continuous Prompts. *ArXiv:2112.08348 [cs]*. Dec.
- Teven Le Scao and Alexander Rush. 2021. How Many Data Points Is a Prompt Worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online, June. Association for Computational Linguistics.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online and Punta Cana, Dominican Republic, November. Association for Computational Linguistics.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *arXiv:2107.13586 [cs]*, July.
- Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT Understands, Too. *arXiv:2103.10385 [cs]*, March.

Google Research

References

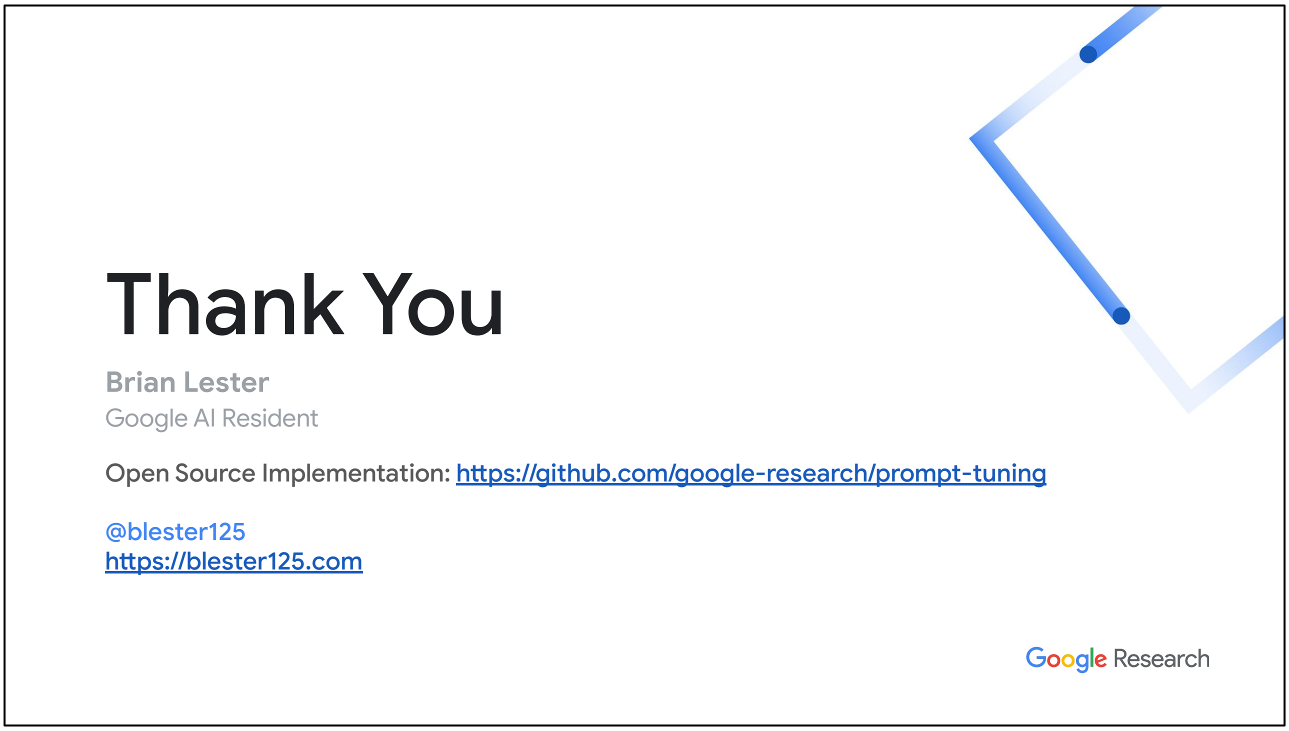
- Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August. Association for Computational Linguistics.
- Lajanugen Logeswaran, Ann Lee, Myle Ott, Honglak Lee, Marc'Aurelio Ranzato, and Arthur Szlam. 2020. Few-Shot Sequence Learning with Transformers. *arXiv:2012.09543 [cs]*, December.
- Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2021. MetaICL: Learning to Learn In Context. *arXiv:2110.15943 [cs]*, October.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, November. Association for Computational Linguistics.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv:2203.02155 [cs]*, March.
- Guanghui Qin and Jason Eisner. 2021. Learning How to Ask: Querying LMs with Mixtures of Soft Prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online, June. Association for Computational Linguistics.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.
- Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf and Alexander M. Rush. 2021. Multitask Prompted Training Enables Zero-Shot Task Generalization. *arXiv:2110.08207 [cs]*, Oct.
- Timo Schick and Hinrich Schütze. 2021. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*: Main Volume, pages 255–269, Online, April. Association for Computational Linguistics.
- Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online, November. Association for Computational Linguistics.
- Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Zhiyuan Liu, Peng Li, Juanzi Li, Lei Hou, Maosong Sun and Jie Zhou. 2021. On Transferability of Prompt Tuning for Natural Language Understanding. *arXiv:2111.06719 [cs]*, Nov.

Google Research

References

- Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. LaMDA: Language Models for Dialog Applications. *ArXiv:2201.08239 [cs]*, Jan.
- Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. SPoT: Better Frozen Model Adaptation through Soft Prompt Transfer. *arXiv:2110.07904 [cs]*, October.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. 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*, pages 353–355, Brussels, Belgium, November. Association for Computational Linguistics.
- Albert Webson and Ellie Pavlick. 2021. Do Prompt-Based Models Really Understand the Meaning of Their Prompts? *arXiv:2109.01247 [cs]*, September.
- Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models Are Zero-Shot Learners. In *Proceedings of ICLR 2022*, April.
- Zhengyan Zhang, Yuxian Gu, Xu Han, Shengqi Chen, Chaojun Xiao, Zhenbo Sun, Yuan Yao, Fanchao Qi, Jian Guan, Pei Ke, Yanzheng Cai, Guoyang Zeng, Zhixing Tan, Zhiyuan Liu, Minlie Huang, Wentao Han, Yang Liu, Xiaoyan Zhu, and Maosong Sun. 2021. CPM-2: Large-Scale Cost-Effective Pre-Trained Language Models. *arXiv:2106.10715 [cs]*, June.
- Mengjie Zhao and Hinrich Schütze. 2021. Discrete and Soft Prompting for Multilingual Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8547–8555, Online and Punta Cana, Dominican Republic, Nov. Association for Computational Linguistics
- Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual Probing Is [MASK]: Learning vs. Learning to Recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online, June. Association for Computational Linguistics.

Google Research



Thank You

Brian Lester
Google AI Resident

Open Source Implementation: https://github.com/google-research/prompt-tuning

@blester125
https://blester125.com

Google Research

Thanks for your time, hopefully this was either interesting or you got a good nap in.

We have open-sourced our implementation, it includes instructions to get running on Google Cloud with TPU training to enable really big models. We have also been adding it. For example, we added the ability to control the nearest neighbors of the prompt tokens like in the Wayward Prompts (Khashabi et al., 2021) paper.

If think prompt tuning is a cool idea hit me up on twitter. If you think the whole thing is stupid and just want to yell at me about how dumb it is, feel free to reach out too.