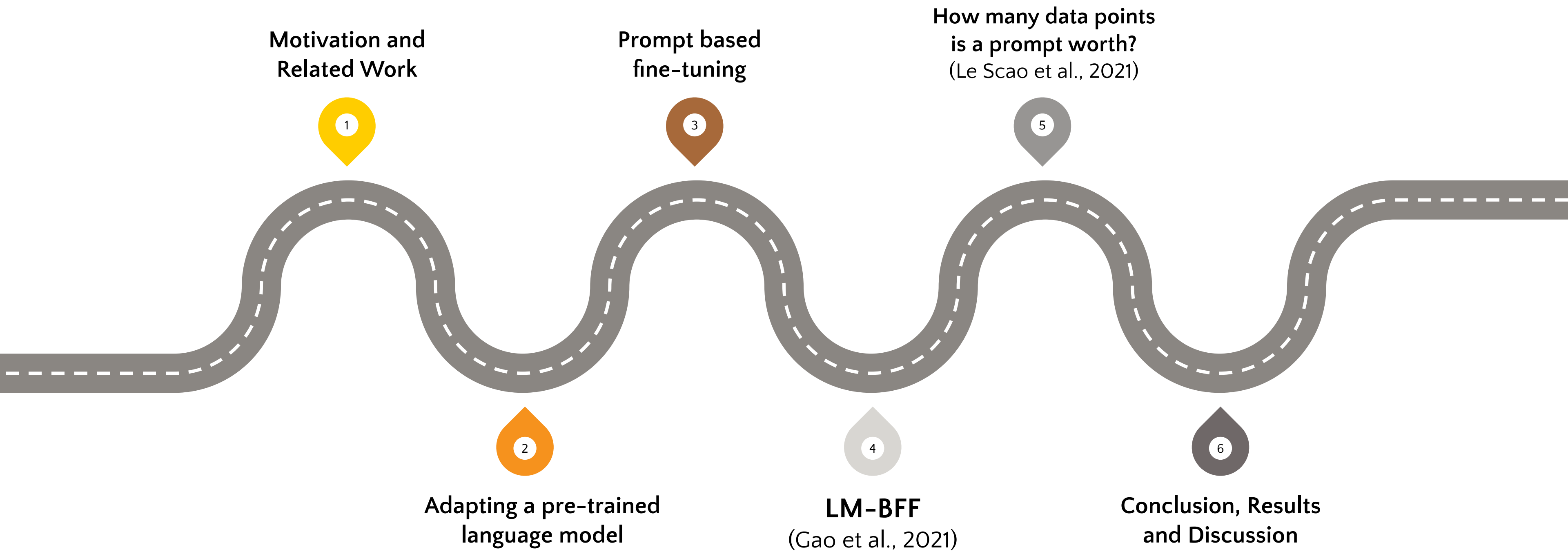
Prompting for Few-shot Learning

Edward Tian and Kaixuan Huang

Roadmap



1
Motivation and Related Work
2
Adapting a pre-trained language model
3
Prompt based fine-tuning
4
LM-BFF (Gao et al., 2021)
5
How many data points is a prompt worth? (Le Scao et al., 2021)
6
Conclusion, Results and Discussion

What is a good prompt?

GPT3: “A good prompt is one that is general enough to be used for a variety of tasks, but specific enough to be helpful for a particular task”

What makes a good prompt? for an NLP task,

*GPT3: “a good prompt is one that is specific and **provides enough context for the model** to be able to generate a response that is relevant to the task.”*

Large Language Models are Few-shot Learners
(Brown, et al.)

- **GPT-3** huge motivator for **prompting**
- **Earliest work in prompts traces back to GPT-1/2** (Radford et al., 2018,2019)
- **With good prompts, LMs can achieve decent zero-shot performance** on tasks from sentiment classification to reading comprehension

Can we make “smaller” LMs also work with few examples?

GPT

Natural language prompts, gigantic model, few in-context examples, no-parameters updated

BERT

110M Parameters

1000x smaller than GPT3

Generic [CLS] Token

Fine-tuned to 2.5 to 400k examples for GLUE tasks

How to adapt a pre-trained Language model?

Head-based fine-tuning

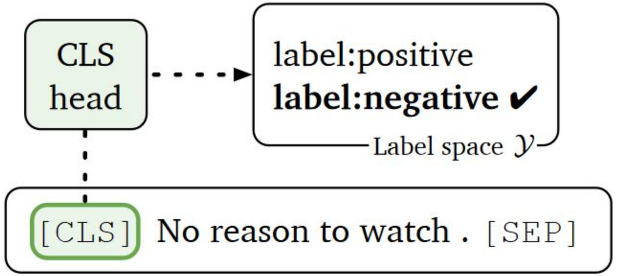


Image Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

How to adapt a pre-trained Language model?

Prompt-based fine-tuning

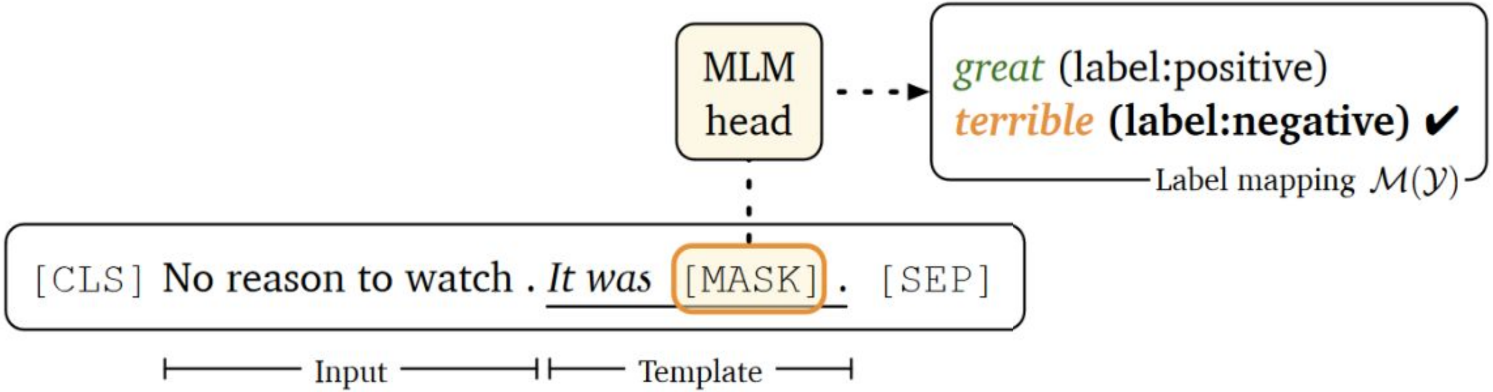


Image Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

Head-based v.s. Prompt-based Fine-tuning

	Head-based	Prompt-based
New parameters?	Yes. hidden_size * num_classes	No
Few-shot friendly?	😓	😊

Prompt-based Fine-tuning (Classification task)

Input: x_1 = No reason to watch.

Step 1. Formulate the downstream task into a (Masked) LM problem using a *template:*

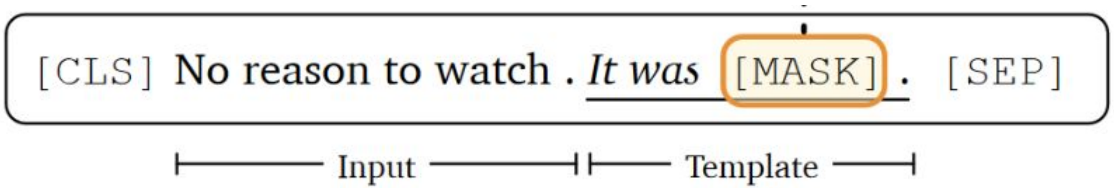
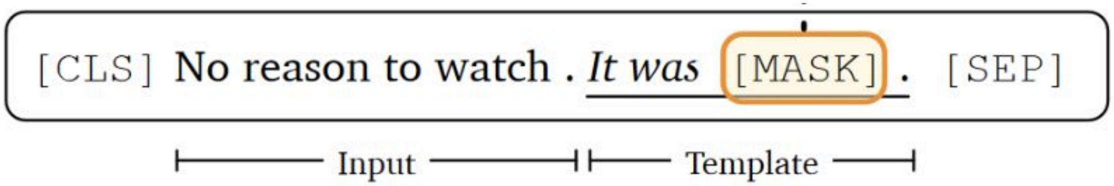


Image Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

Prompt-based Fine-tuning (Classification task)

Input: x_1 = No reason to watch.

Step 1. Formulate the downstream task into a (Masked) LM problem using a *template:*



Step 2. Choose a *label word mapping* $\mathcal{M}$, which maps task labels to individual words.

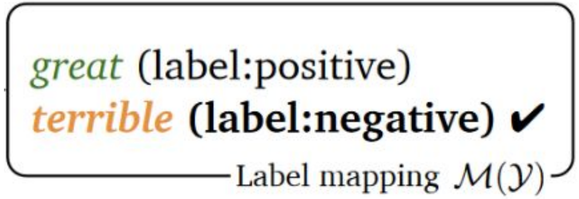


Image Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

Prompt-based Fine-tuning (Classification Task)

Step 3. Fine-tune the LM to fill in the correct label word.

$$p(y \mid x_{\text{in}}) = p\left(\texttt{[MASK]} = \mathcal{M}(y) \mid x_{\text{prompt}}\right)$$
$$= \frac{\exp\left(\mathbf{w}_{\mathcal{M}(y)} \cdot \mathbf{h}_{\texttt{[MASK]}}\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(\mathbf{w}_{\mathcal{M}(y')} \cdot \mathbf{h}_{\texttt{[MASK]}}\right)},$$

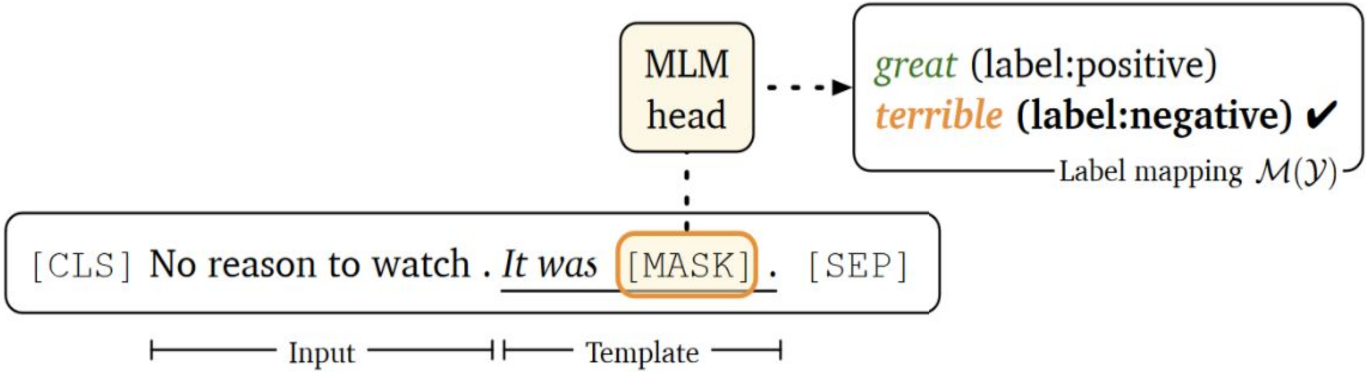
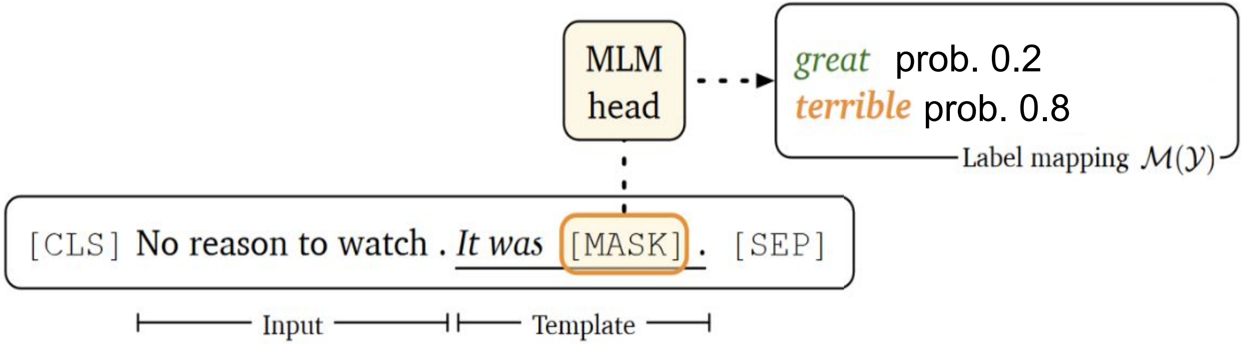


Image Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

Prompt-based Fine-tuning (Regression Task)

Regression: interpolating between two extremes

$$y = v_l \cdot p(y_l \mid x_{\text{in}}) + v_u \cdot p(y_u \mid x_{\text{in}})$$



Predicted y = 0.2

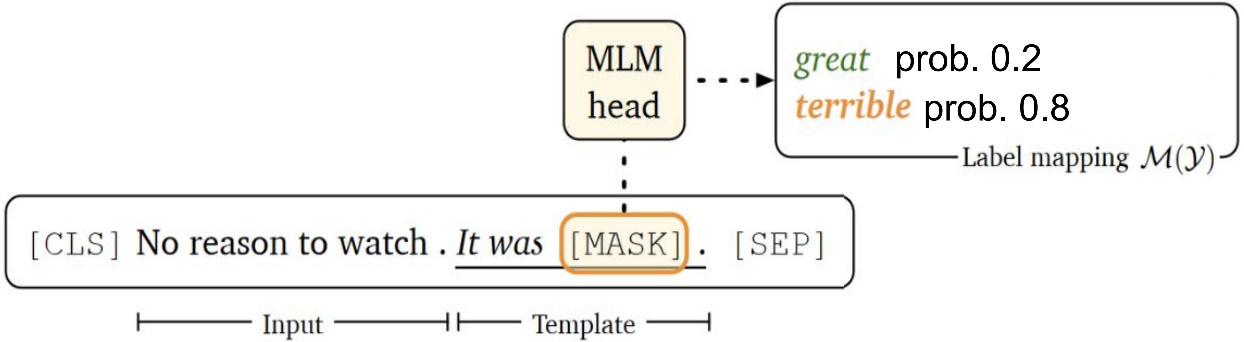
Image adapted from: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

Prompt-based Fine-tuning (Regression Task)

Regression: interpolating between two extremes

$$y = v_l \cdot p(y_l \mid x_{\text{in}}) + v_u \cdot p(y_u \mid x_{\text{in}})$$

The LM is fine-tuned to minimize the KL-divergence between the inferred $p(y_u \mid x_{\text{in}})$ and the observed mixture weight $(y - v_l)/(v_u - v_l)$



Predicted y = 0.2

Image adapted from: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

Q1. How does prompt-based fine-tuning work and why does it outperform head-based fine-tuning (as the method described in BERT) in low-data regimes?

A1. Prompt-based fine-tuning involves:

- a **template** which turns the downstream task into a (masked) language modelling problem, and
- a set of **label words** that map the textual output of the LM to the classification labels.

In this way, we don't need to introduce any new parameters so all the pre-trained parameters can be fine-tuned more sample-efficiently.

It outperforms head-based fine-tuning in low-data regimes since BERT introduces new randomly-initialized parameters (often more than 1k), which are hard to learn well from only a few examples.

Making Pre-trained Language Models Better Few-shot Learners

Tianyu Gao, Adam Fisch, Danqi Chen

Datasets

Category	Dataset	$\lvert\mathcal{Y}\rvert$	Type	Labels (classification tasks)
single-sentence	SST-2	2	sentiment	positive, negative
	SST-5	5	sentiment	v. pos., positive, neutral, negative, v. neg.
	MR	2	sentiment	positive, negative
	CR	2	sentiment	positive, negative
	MPQA	2	opinion polarity	positive, negative
	Subj	2	subjectivity	subjective, objective
	TREC	6	question cls.	abbr., entity, description, human, loc., num.
	CoLA	2	acceptability	grammatical, not_grammatical
sentence-pair	MNLI	3	NLI	entailment, neutral, contradiction
	SNLI	3	NLI	entailment, neutral, contradiction
	QNLI	2	NLI	entailment, not_entailment
	RTE	2	NLI	entailment, not_entailment
	MRPC	2	paraphrase	equivalent, not_equivalent
	QQP	2	paraphrase	equivalent, not_equivalent
	STS-B	$\mathcal{R}$	sent. similarity	-

- Most tasks: # Labels <= 3

Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

Datasets

Category	Dataset	$\|\mathcal{Y}\|$	Type	Labels (classification tasks)
single-sentence	SST-2	2	sentiment	positive, negative
	SST-5	5	sentiment	v. pos., positive, neutral, negative, v. neg.
	MR	2	sentiment	positive, negative
	CR	2	sentiment	positive, negative
	MPQA	2	opinion polarity	positive, negative
	Subj	2	subjectivity	subjective, objective
	TREC	6	question cls.	abbr., entity, description, human, loc., num.
	CoLA	2	acceptability	grammatical, not_grammatical
sentence-pair	MNLI	3	NLI	entailment, neutral, contradiction
	SNLI	3	NLI	entailment, neutral, contradiction
	QNLI	2	NLI	entailment, not_entailment
	RTE	2	NLI	entailment, not_entailment
	MRPC	2	paraphrase	equivalent, not_equivalent
	QQP	2	paraphrase	equivalent, not_equivalent
	STS-B	$\mathcal{R}$	sent. similarity	-

- Most tasks: # Labels <= 3
- SST-5, TREC have 5 or 6 labels

Source: *Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021*

Datasets

Category	Dataset	$\|\mathcal{Y}\|$	Type	Labels (classification tasks)
single-sentence	SST-2	2	sentiment	positive, negative
	SST-5	5	sentiment	v. pos., positive, neutral, negative, v. neg.
	MR	2	sentiment	positive, negative
	CR	2	sentiment	positive, negative
	MPQA	2	opinion polarity	positive, negative
	Subj	2	subjectivity	subjective, objective
	TREC	6	question cls.	abbr., entity, description, human, loc., num.
	CoLA	2	acceptability	grammatical, not_grammatical
sentence-pair	MNLI	3	NLI	entailment, neutral, contradiction
	SNLI	3	NLI	entailment, neutral, contradiction
	QNLI	2	NLI	entailment, not_entailment
	RTE	2	NLI	entailment, not_entailment
	MRPC	2	paraphrase	equivalent, not_equivalent
	QQP	2	paraphrase	equivalent, not_equivalent
	→ STS-B	$\mathcal{R}$	sent. similarity	-

- Most tasks: # Labels <= 3
- SST-5, TREC have 5 or 6 labels
- STS-B is a regression task

Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

Examples

SST-2: sentiment analysis.

- E.g. **S1** = "The movie is ridiculous". **Label**: negative.
- Manual prompt:

Template	**Label words**
$<S_1>$ It was `[MASK]` .	great/terrible

Examples

SNLI: Natural Language Inference

- **S1** = "A soccer game with multiple males playing". **S2** = "Some men are playing sport". **Label**: Entailment.
- Manual prompt:

Template	**Label words**
$<S_1>$? [MASK] , $<S_2>$	Yes/Maybe/No

Few-shot Learning & Evaluation Protocol

Q2. How does (Gao et al., 2021) conduct evaluations in few-shot settings?

- Training dataset: **K=16** examples per class.
- Dev dataset: **same** size as training dataset.

Performance measured across 5 random splits of {train, dev} set.

What is a “True” Few-shot Learning setting?

- Perez et al. (2021): “Tuned few-shot learning algorithms should be compared against data-rich supervised learning algorithms that use the same amount of total data |*D* train| + |*D* val|”
- Larger dev set leads to better performance.

Fine-tuning	SST-2	SNLI	TREC	MRPC
No $\mathcal{D}_{\text{dev}}$	79.5	49.2	83.9	77.8
$\lvert\mathcal{D}_{\text{dev}}\rvert = \lvert\mathcal{D}_{\text{train}}\rvert$	81.4	48.4	88.8	76.6
$\lvert\mathcal{D}_{\text{dev}}\rvert = 10\lvert\mathcal{D}_{\text{train}}\rvert$	83.5	52.0	89.4	79.6
Prompt-based FT	**SST-2**	**SNLI**	**TREC**	**MRPC**
No $\mathcal{D}_{\text{dev}}$	92.1	75.3	84.8	70.2
$\lvert\mathcal{D}_{\text{dev}}\rvert = \lvert\mathcal{D}_{\text{train}}\rvert$	92.7	77.2	84.8	74.5
$\lvert\mathcal{D}_{\text{dev}}\rvert = 10\lvert\mathcal{D}_{\text{train}}\rvert$	93.0	79.7	89.3	80.9

Same setting as PET (Schick and Schütze, 2021a,b)

Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

Q2: Is it still true few-shot learning if we manually tune the prompt?

A2: It is still "true" few-shot learning, because the whole training process, including hyper-parameter/prompt tuning, still only involves a few examples, which is the training dataset plus the development dataset.

How important is a good prompt for few-shot learning?

Label words match the semantic classes → good final accuracy

Template	Label words	Accuracy
SST-2 (positive/negative)		mean (std)
<S_1> It was `[MASK]` .	great/terrible	**92.7 (0.9)**
<S_1> It was `[MASK]` .	good/bad	92.5 (1.0)
<S_1> It was `[MASK]` .	cat/dog	91.5 (1.4)
<S_1> It was `[MASK]` .	dog/cat	86.2 (5.4)
<S_1> It was `[MASK]` .	terrible/great	83.2 (6.9)
Fine-tuning	-	81.4 (3.8)

Experiments are done with **K=16** examples per class.

Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

How important is a good prompt for few-shot learning?

A small change in the template can make a huge difference (>10% performance drop)

Template	Label words	Accuracy
SNLI (entailment/neutral/contradiction)		mean (std)
$<S_1>$? `[MASK]` , $<S_2>$	Yes/Maybe/No	**77.2 (3.7)**
$<S_1>$. `[MASK]` , $<S_2>$	Yes/Maybe/No	76.2 (3.3)
$<S_1>$? `[MASK]` $<S_2>$	Yes/Maybe/No	74.9 (3.0)
$<S_1>$ $<S_2>$ `[MASK]`	Yes/Maybe/No	65.8 (2.4)
$<S_2>$? `[MASK]` , $<S_1>$	Yes/Maybe/No	62.9 (4.1)
$<S_1>$? `[MASK]` , $<S_2>$	Maybe/No/Yes	60.6 (4.8)
Fine-tuning	-	48.4 (4.8)

Put the <MASK> to the end

Swap <S1> and <S2>

Experiments are done with **K=16** examples per class.

Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

LM-BFF

GPT-3 🤖	Very very large language Unchanged Model Parameters Usually human-designed **prompts** and **demonstrations**
PET 🐱	**Small** language model **Fine-tuning** model parameters Manually-designed **prompts**
LM-BFF 👯	**Small** language model **Fine-tuning** model parameters **Automatically-searched** prompts and **demonstrations**

Several slides on LMBFF adapted from Tianyu Gao's conference presentation.

How do we design a good prompt?

BoolQ: given a passage q and question p, design a prompt for question answering

For **BoolQ**, given a passage p and question q:

p. Question: q? Answer: <MASK>.

p. Based on the previous passage, q? <MASK>.

Based on the following passage, q? <MASK>. p

with "yes" or "no" as verbalizers for True and False.

How do we design a good prompt?

WiC: given two sentences S1 and S2, and a word W, design a prompt to determine whether W was used in the same sense in both sentences.

For **WiC**, given two sentences s_1 and s_2 and a word w, we classify whether w was used in the same sense.

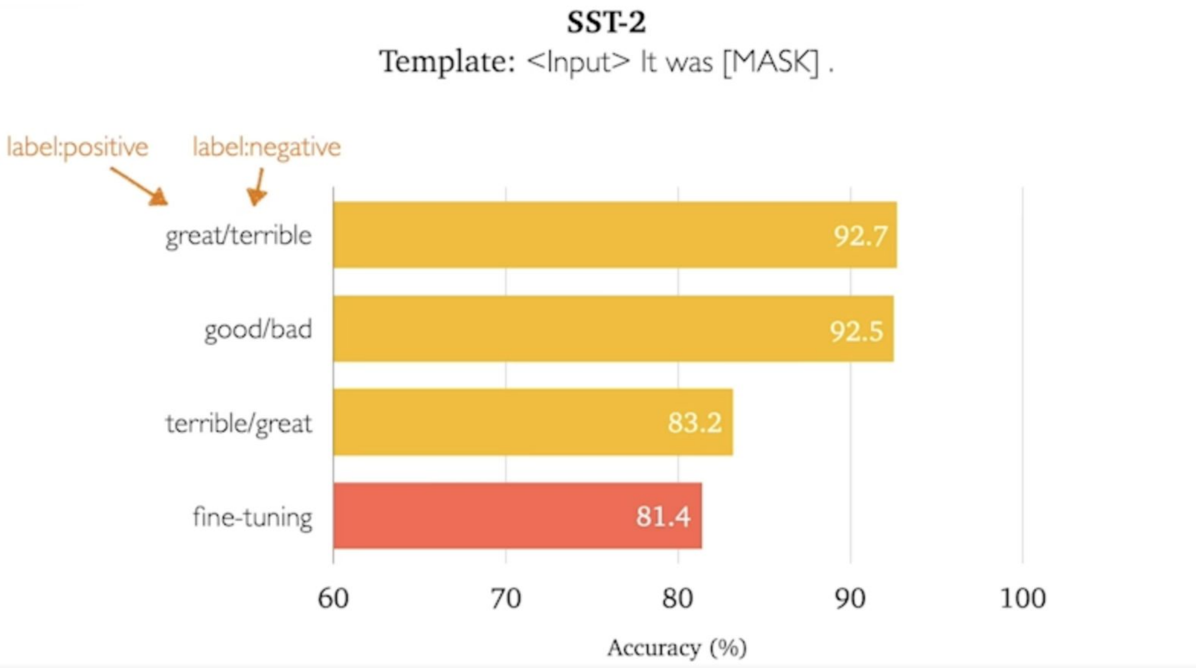
"s_1" / "s_2". Similar sense of "w"? <MASK>.

s_1 s_2 Does w have the same meaning in both sentences? <MASK>.

How do we design good prompts?

- Difficult problem, manually designed in previous works (Schick and Schutze, 2021 a.b)
- Requires domain expertise and trial and error
- Challenge to construct prompt P find a template *T* and label words *M(y)* that work in conjunction
- Low number of examples -> overfitting

Recall …



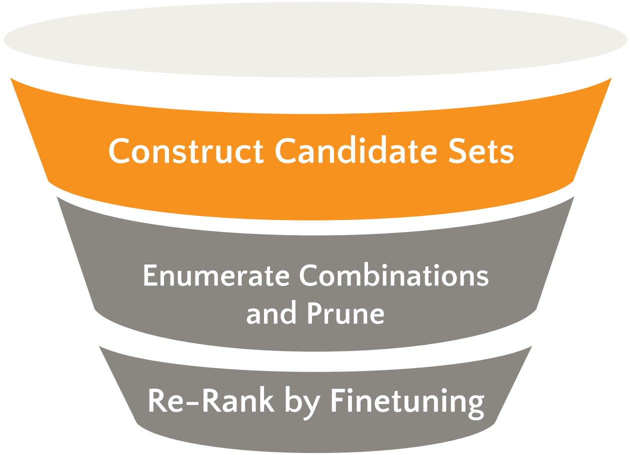
* Slight variations in prompts between terrible/great leads to sizable differences!

Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al.

Automatic
Prompt Generation

* In experiments assume access to a few-shot training and development set with 16 samples per class.

Automatic Label Search



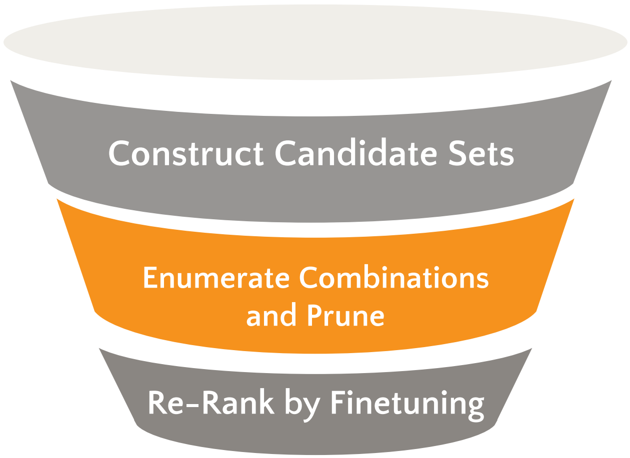
For a classification task, for each **label**, **construct** a set of top-k words with highest MLM **probabilities** conditioned on all training examples

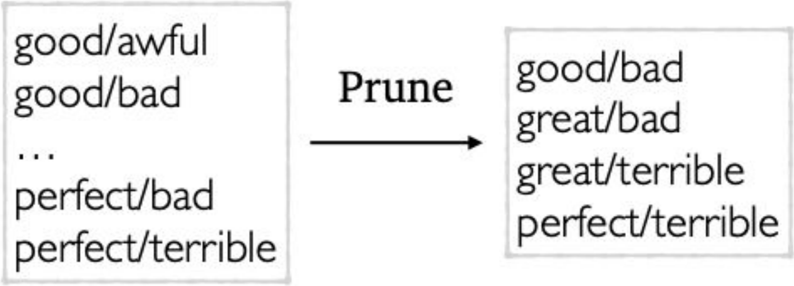
Given the **manual** template: <S> It was [MASK] .

label:positive	label:negative
good	awful
great	bad
perfect	terrible
...	...

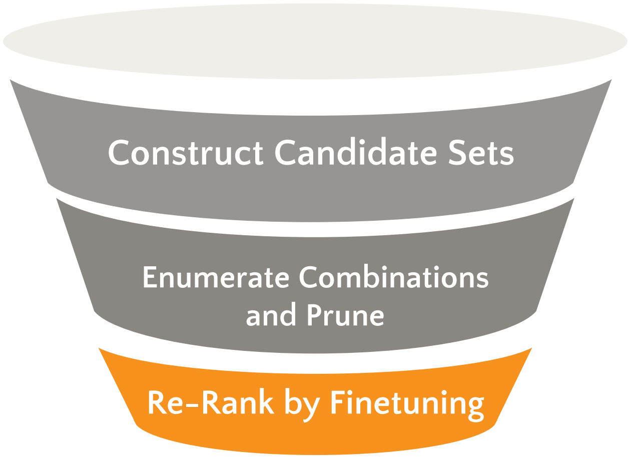
Automatic Label Search

Enumerate all combinations. **Prune** by Zero-shot results on training set.

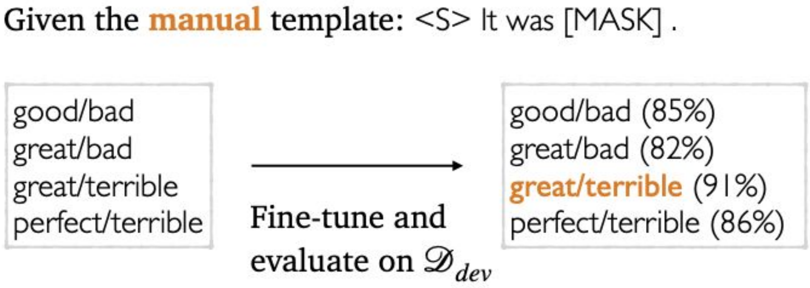




Automatic Label Search



Finetune all top n assignments and re-rank to find the best ones using development dataset.



Intuition

Mask the prompts and ask T5 🚀
to ___ in the blanks

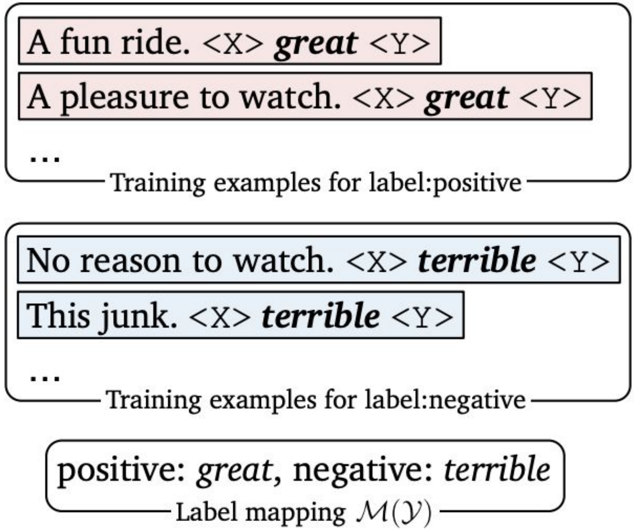
“

Automatic Template Search

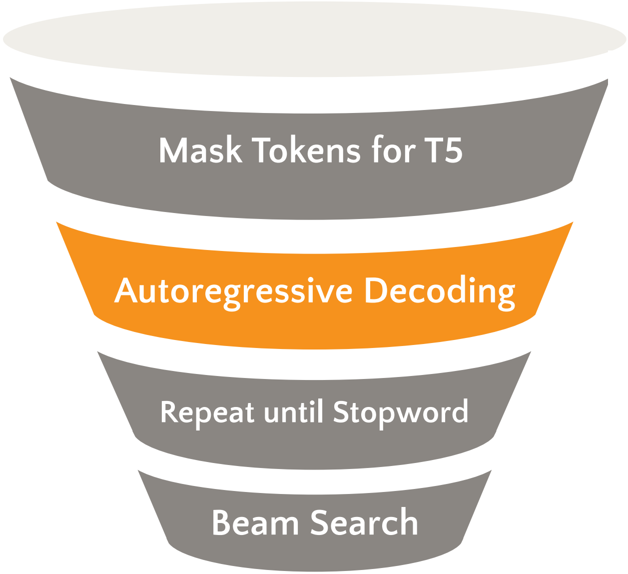
Heuristic

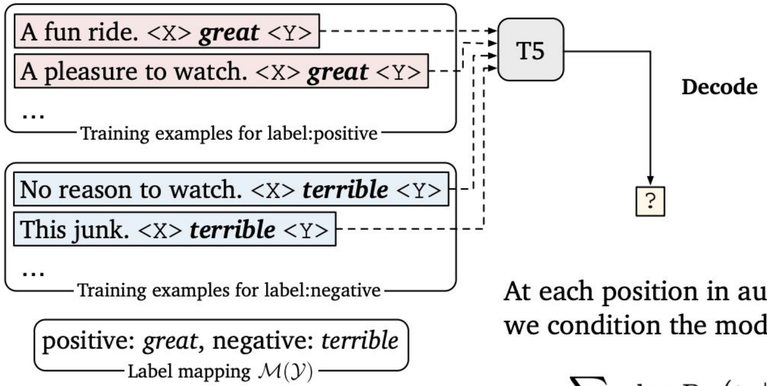
1. Use T5 to generate candidates.
2. Re-rank them based on performance on development set after fine-tuning.

Automatic Template Search



Automatic Template Search

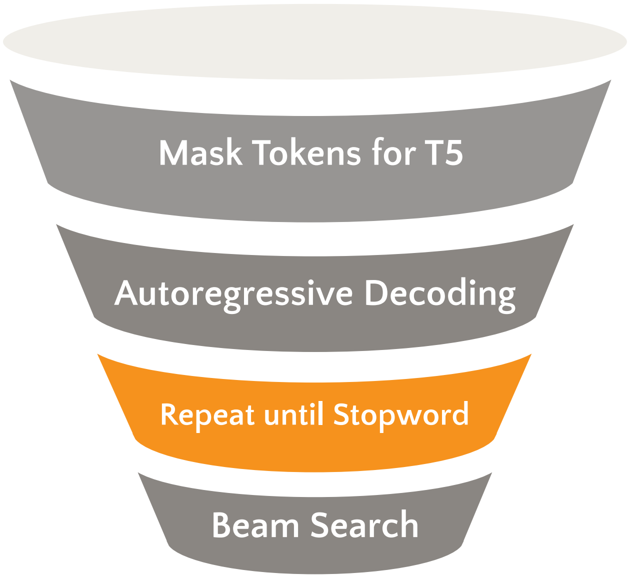


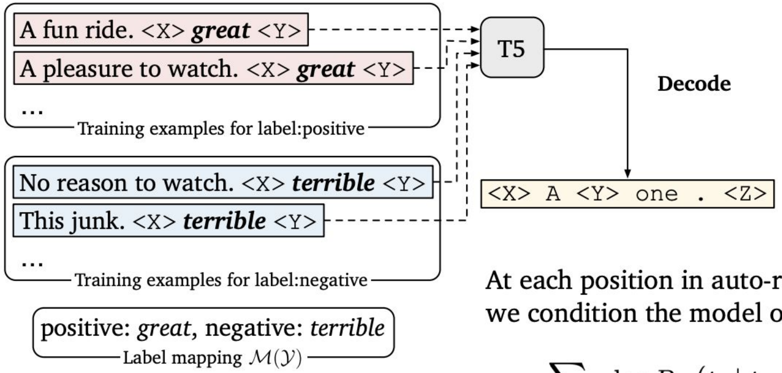


At each position in auto-regressive decoding, we condition the model on **all training examples**.

$$\sum_{(x_{\text{in}},y)\in\mathcal{D}_{\text{train}}} \log P_{\text{T5}}(t_j \mid t_1, ..., t_{j-1}, \mathcal{T}_{\text{g}}(x_{\text{in}}, y))$$

Automatic Template Search

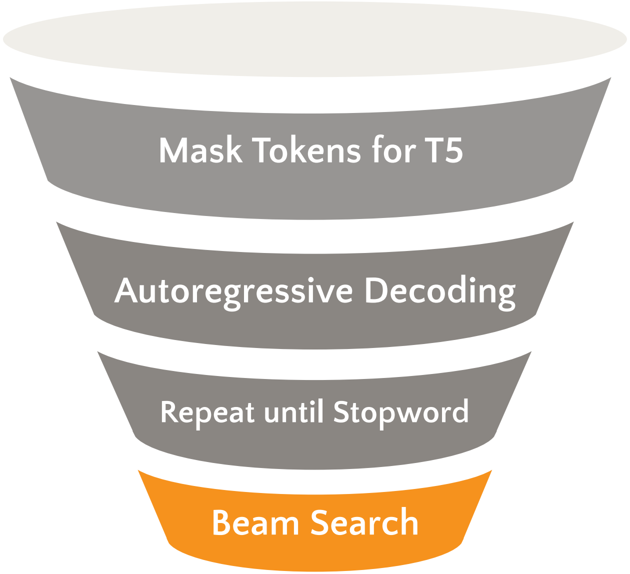




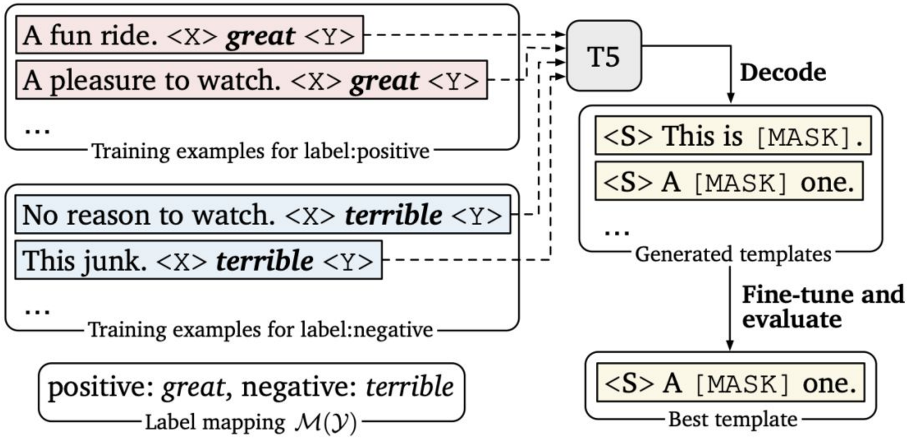
At each position in auto-regressive decoding, we condition the model on **all training examples**.

$$\sum_{(x_{\text{in}},y)\in\mathcal{D}_{\text{train}}} \log P_{\text{T5}}\left(t_j \mid t_1, \ldots, t_{j-1}, \mathcal{T}_{\text{g}}\left(x_{\text{in}}, y\right)\right)$$

Automatic Template Search

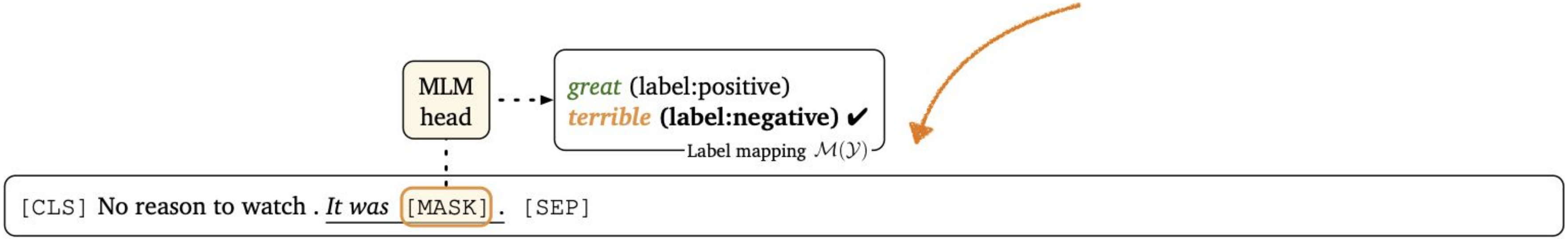


Apply **beam search** with large width ~100 to generate many templates to evaluate



Demonstrations

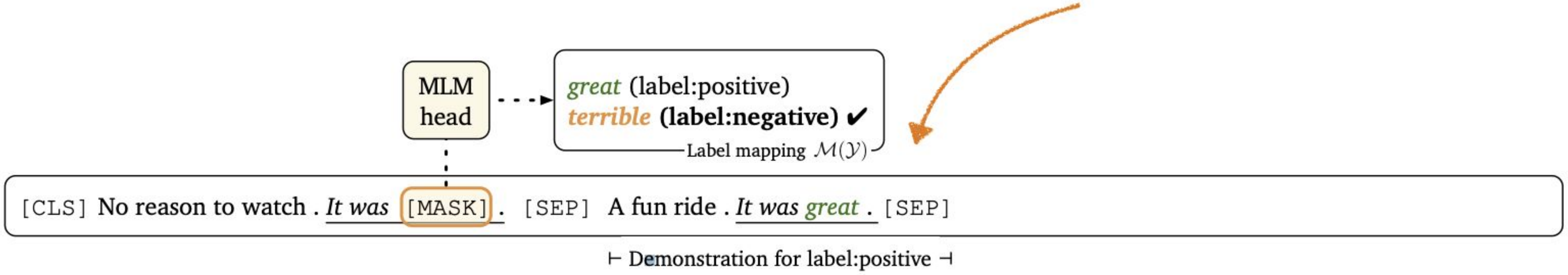
GPT3 In-context Learning:
Randomly Samples Examples and fills them in context 👎🏽



Prompt-based fine-tuning

Demonstrations

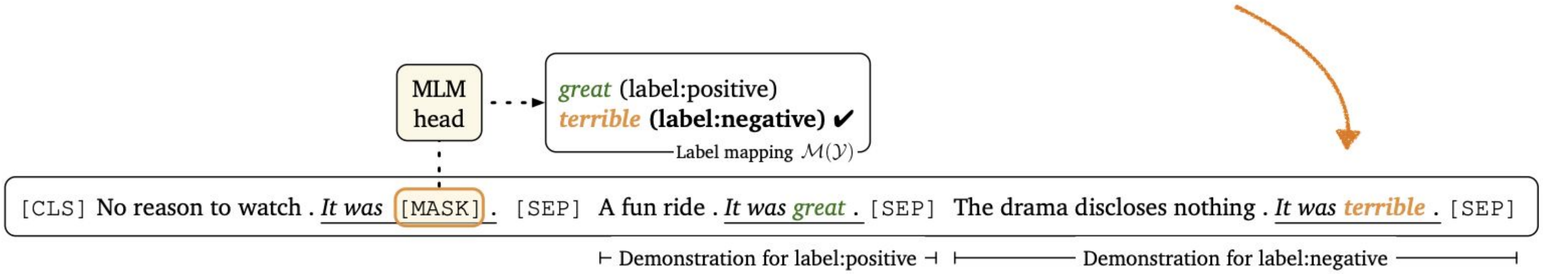
Improved: Selective Sampling, ie. for this example sample from then positive class 😎



Prompt-based fine-tuning with demonstrations

Demonstrations

And we can also sample one from a negative training instance



Prompt-based fine-tuning with demonstrations

Intuition

Selective apply **demonstrations** that are semantically close to the input for optimal results

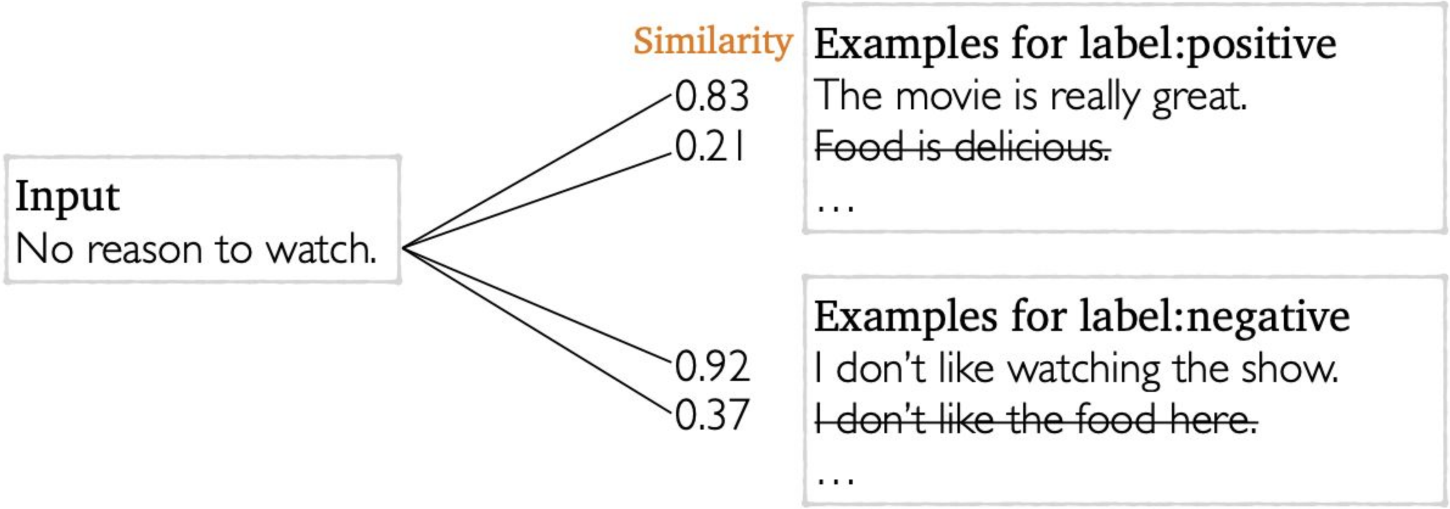
“

Demonstrations Sampling

Heuristic

1. Measure cosine similarity between all training examples and input.
2. Use pre-trained sentence encoder BERT to measure similarities
3. Only use top 50% of examples as demonstration candidates

Demonstrations Example



Recall: Experiment Setup

16 Experiments,
8 Single-Sentence
and 7 Sentence-pair tasks

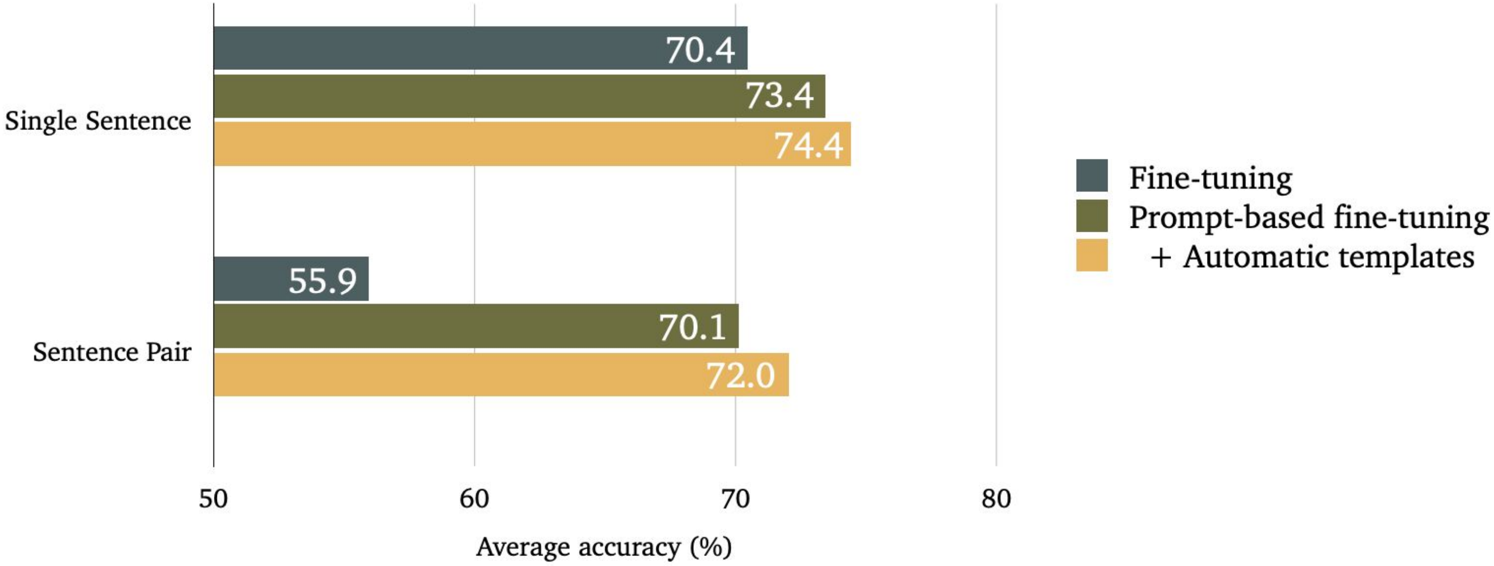
For each experiment, paper used **16 samples per class** for training and development sets

Sample **5 fewshot sets** for each dataset and **averaged** the results to address variance

```
python run.py \
    --task_name SST-2 \
    --data_dir data/k-shot/SST-2/16-42 \
    --overwrite_output_dir \
    --do_train \
    --do_eval \
    --do_predict \
    --evaluate_during_training \
    --model_name_or_path roberta-large \
    --few_shot_type prompt-demo \
    --num_k 16 \
    --max_steps 1000 \
    --eval_steps 100 \
    --per_device_train_batch_size 2 \
    --learning_rate 1e-5 \
    --num_train_epochs 0 \
    --output_dir result/tmp \
    --seed 42 \
    --template "*cls**sent_0*_It_was*mask*.*sep+*" \
    --mapping "{'0':'terrible','1':'great'}" \
    --num_sample 16 \
```

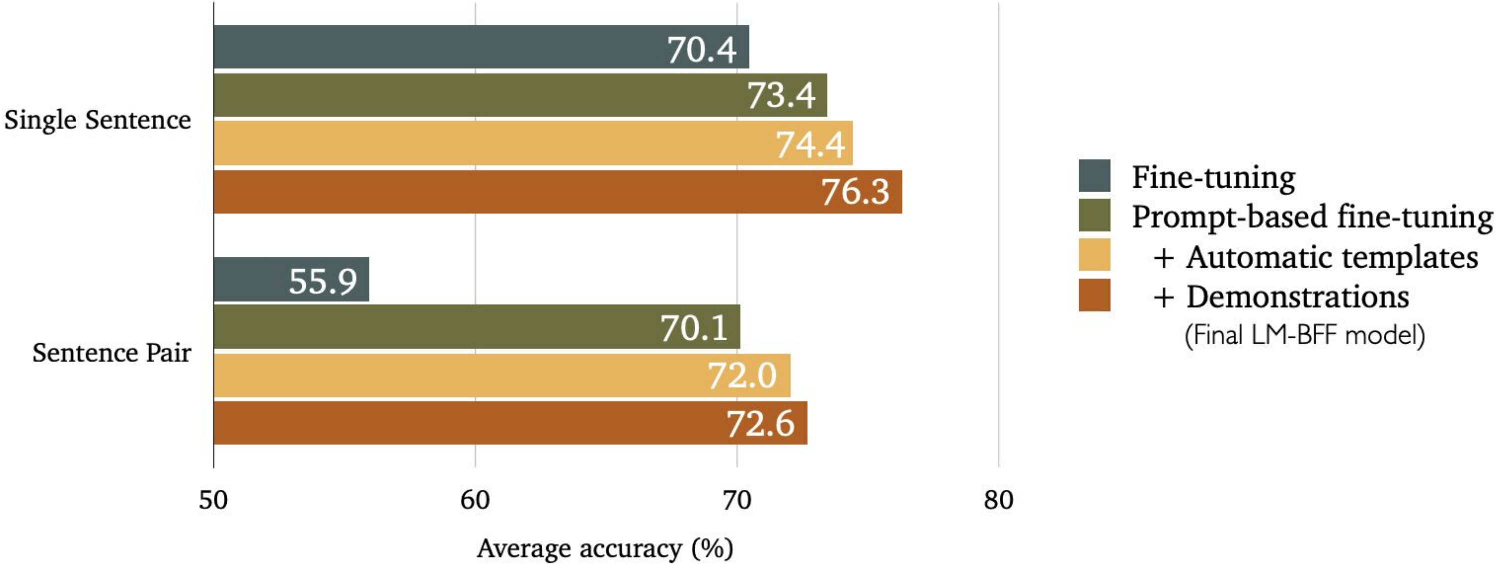
Figure: Example of running an output example source from github.com/princeton-nlp/LM-BFF

Results (single prompts)

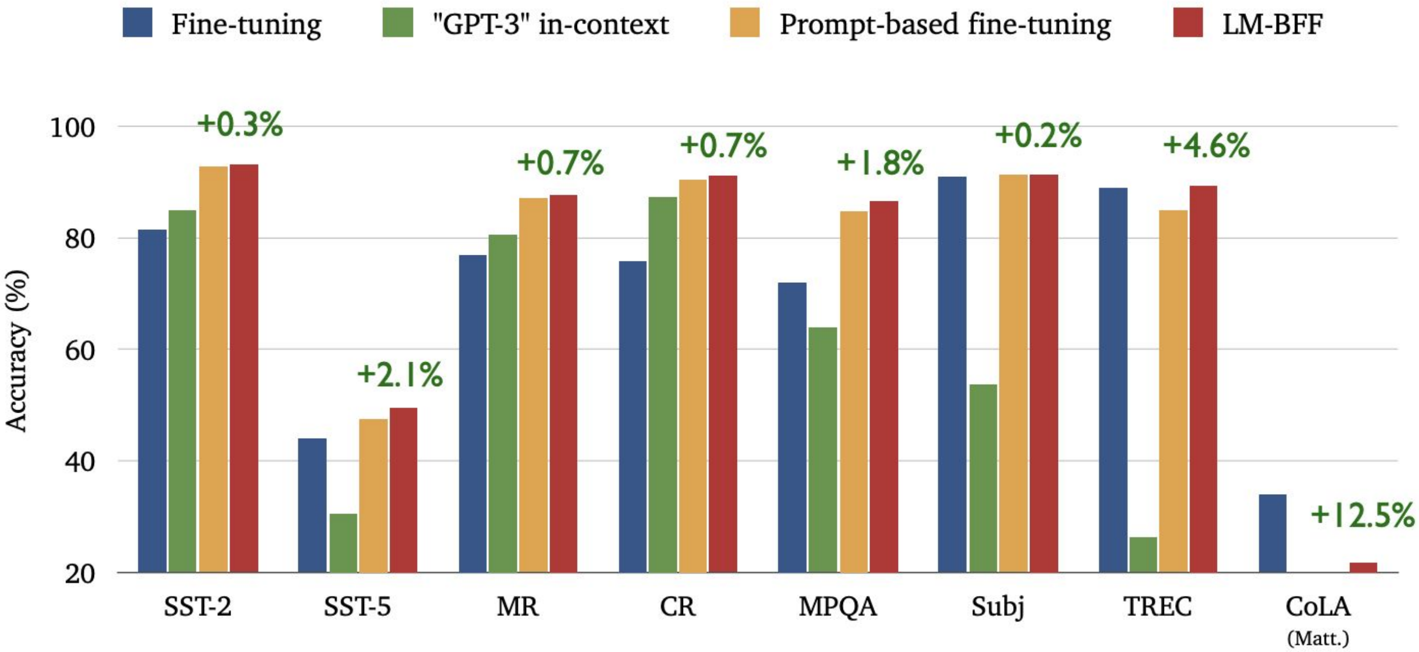


Tables on LMBFF results adapted from Tianyu Gao's conference presentation.

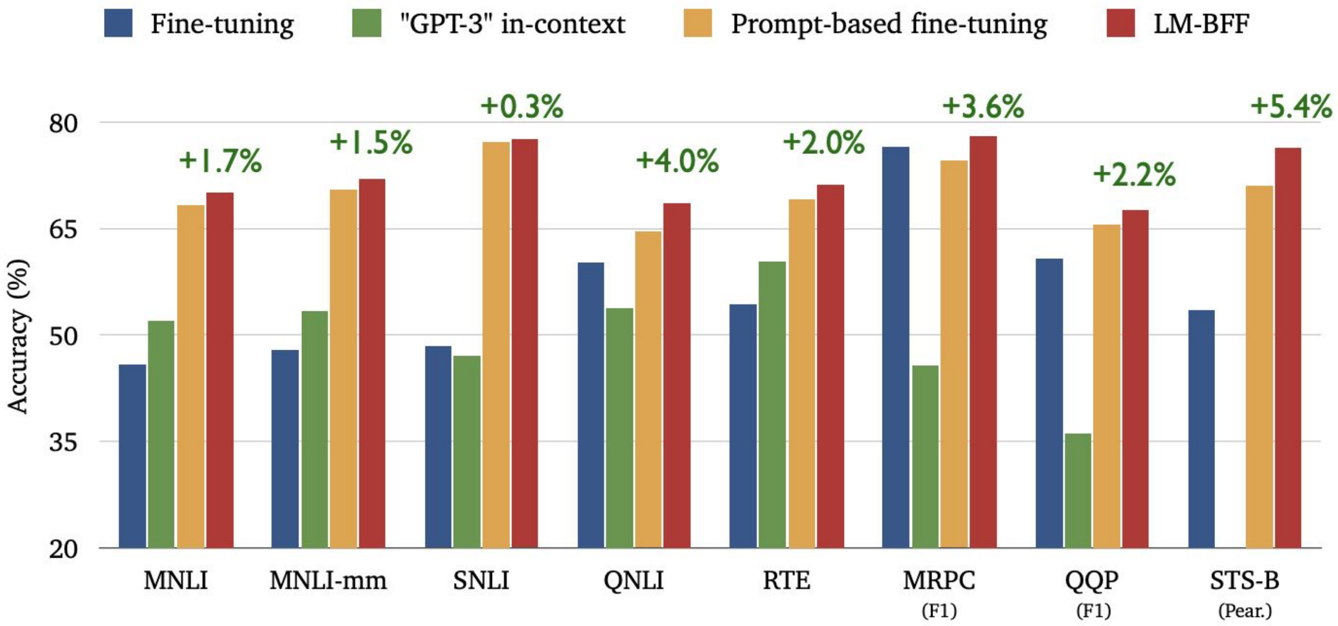
Results (single prompts)



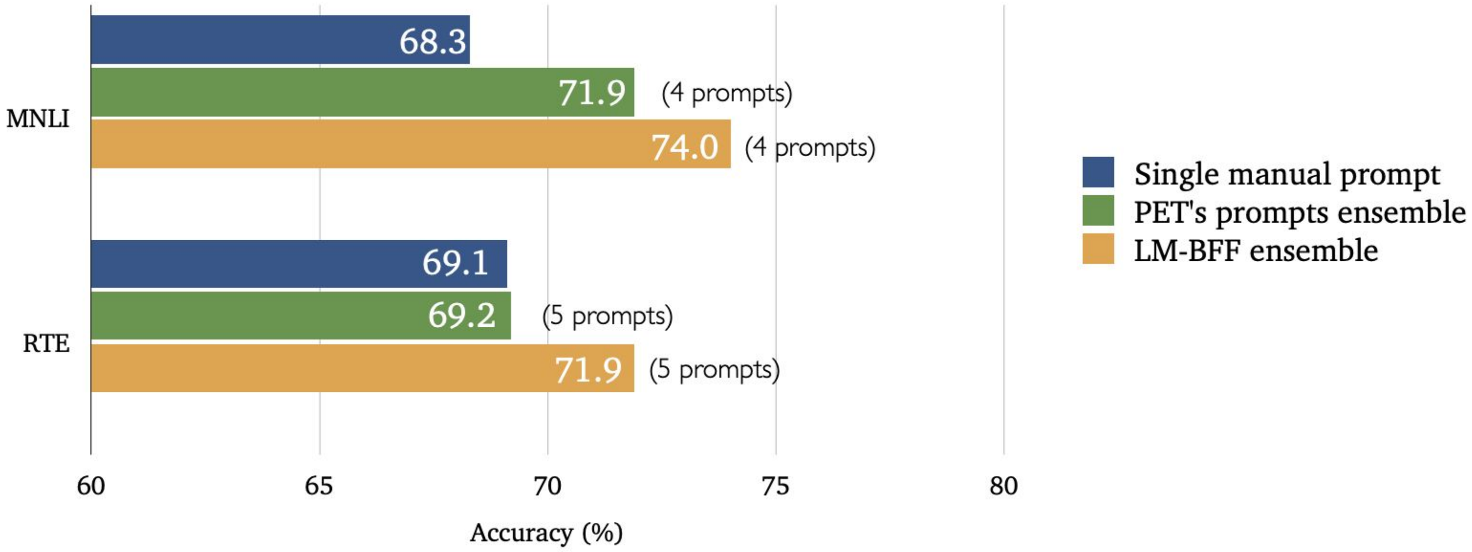
Results (single prompts)



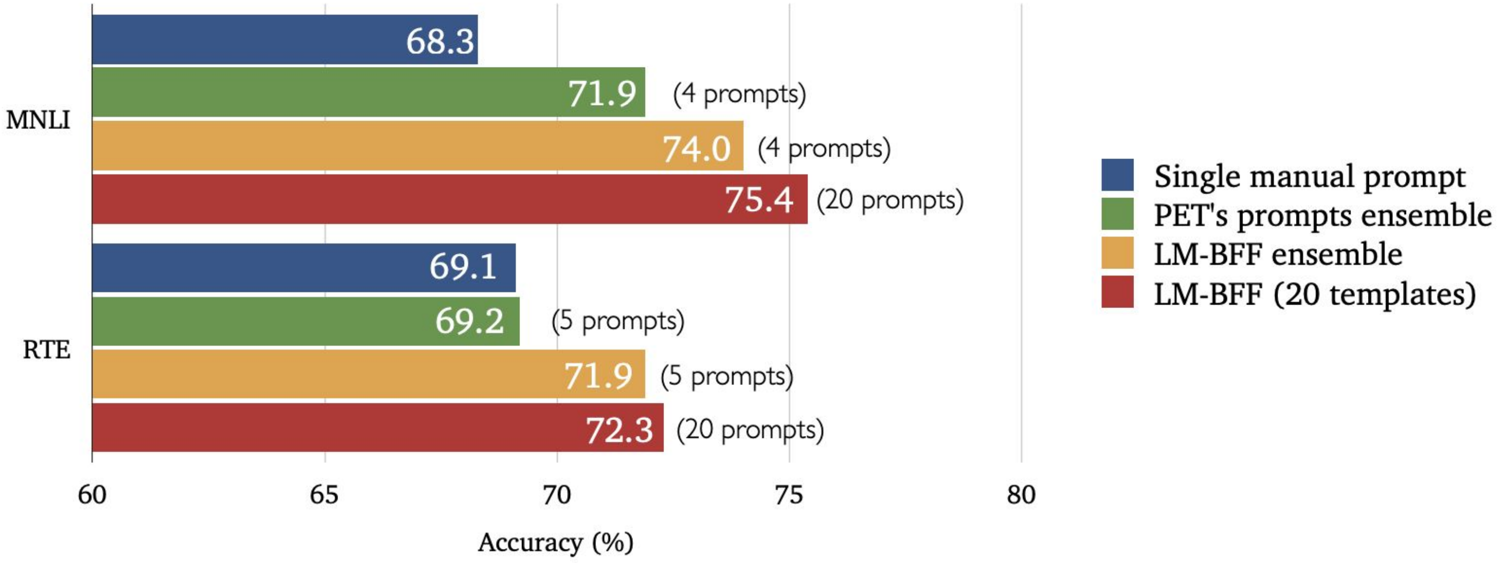
Results (single prompts)



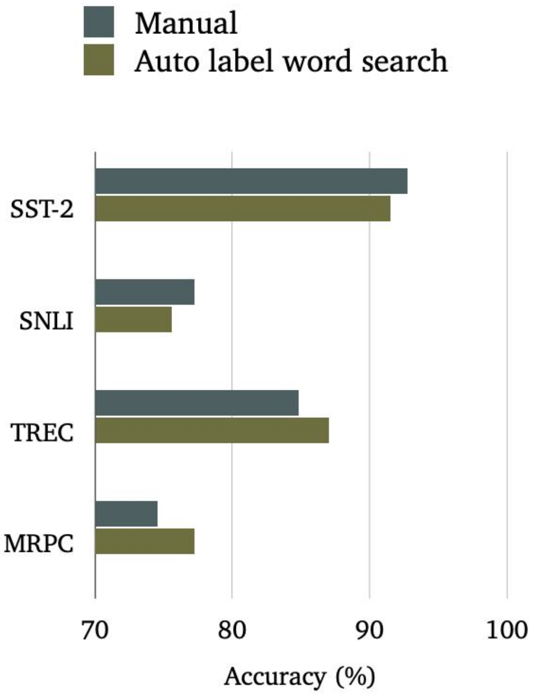
Results (ensemble)



Results (ensemble)



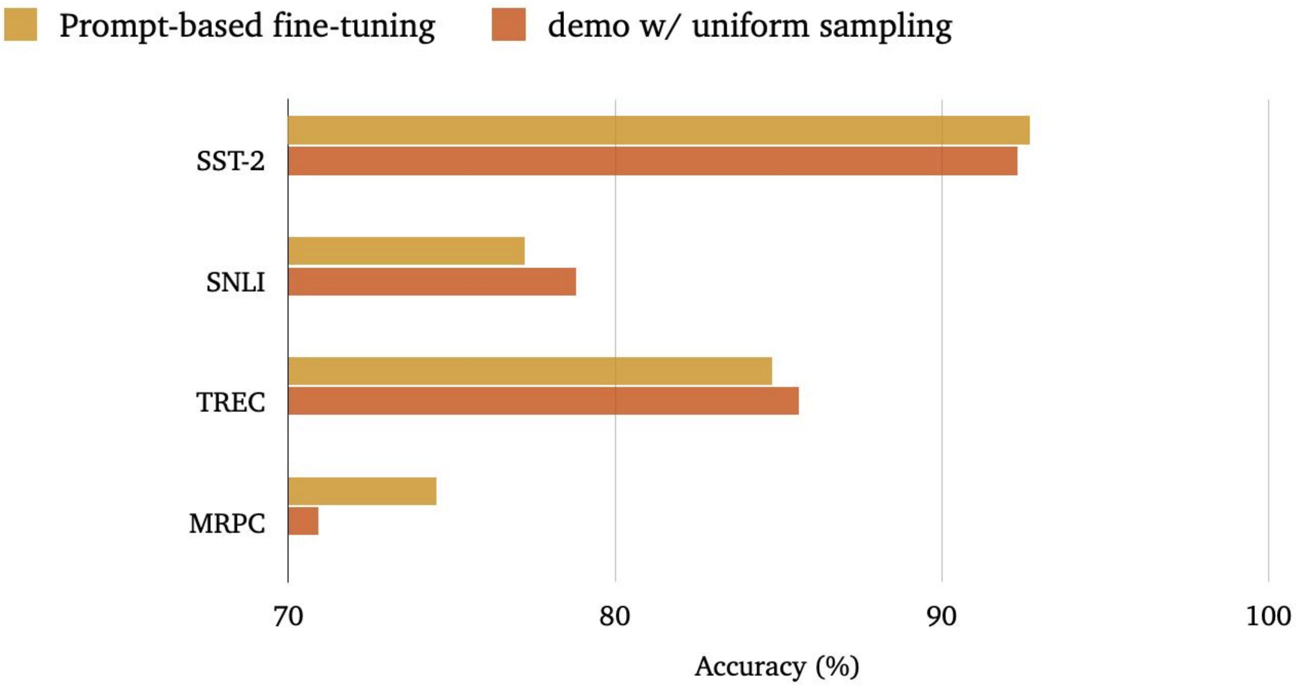
Ablation Study: Automatic Prompt Search



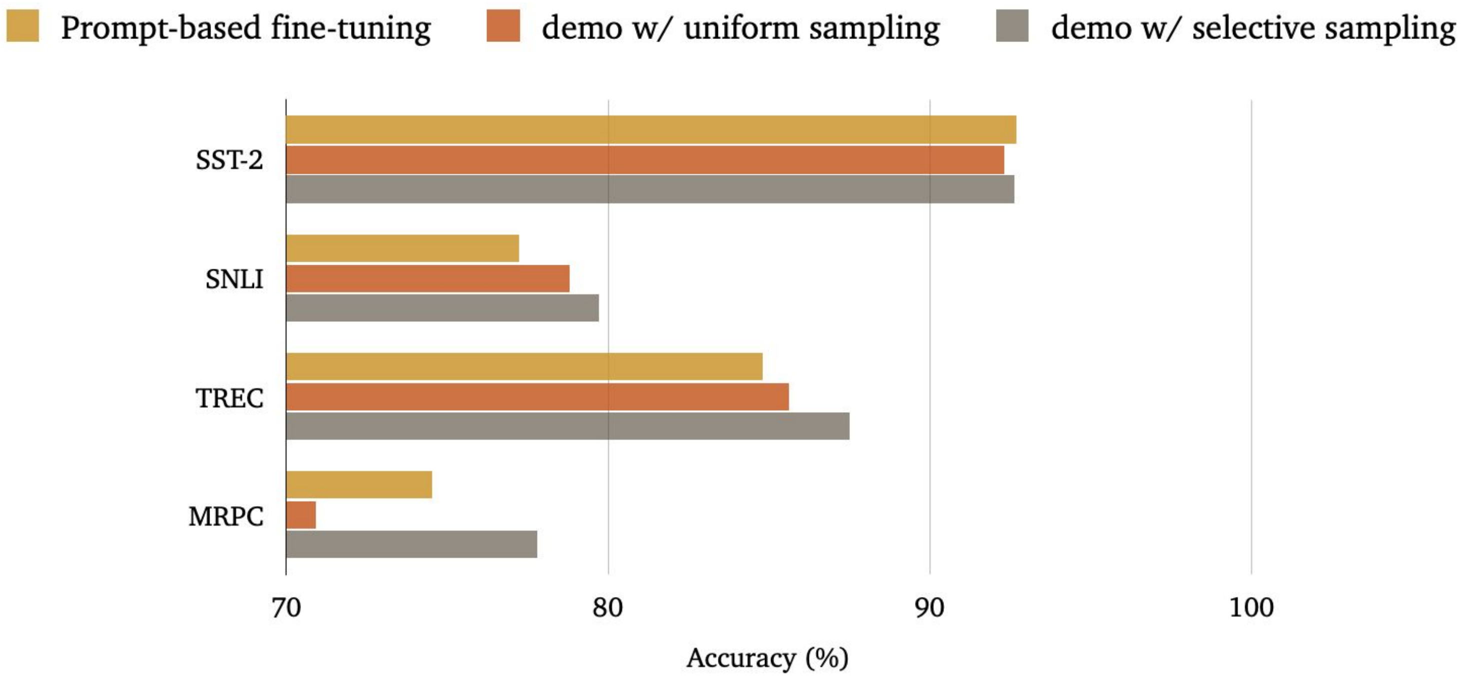
SST-2	(positive/negative)
	$\mathcal{T}(x_{\text{in}})$ = <S_1> It was [MASK].
	#1. irresistible/pathetic
	#2. wonderful/bad
	#3. delicious/bad

SNLI	(entailment/neutral/contradiction)
	$\mathcal{T}(x_{\text{in}})$ = <S_1> ? [MASK] , <S_2>
	#1. Alright/Watch/Except
	#2. Hi/Watch/Worse
	#3. Regardless/Fortunately/Unless

Ablation Study: Demonstrations



Ablation Study: Demonstrations



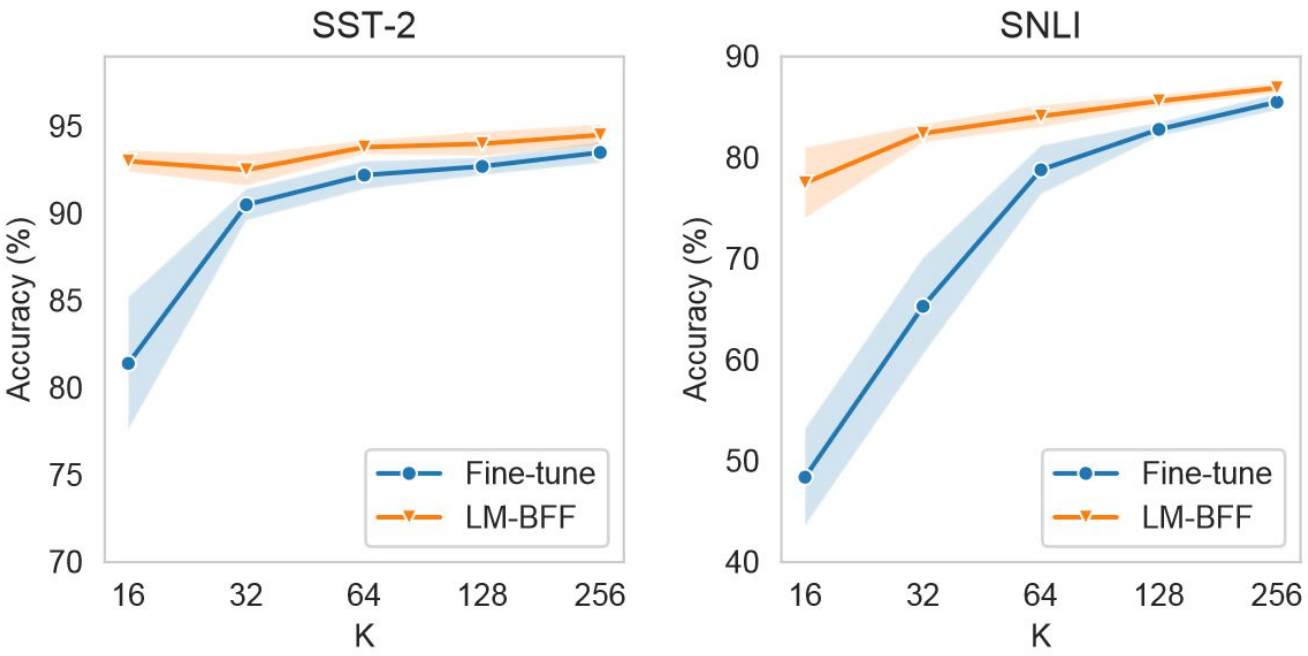
Key Findings

- LMBFF Introduced **automatic search** prompt based fine tuning and a selective way for **incorporating demonstrations**

- Provided few-shot evaluations on 15 tasks. LMBFF **dramatically outperforms** standard fine tuning

- Limitations include large variance and automatic search reliance on manual label words

Comment



The benefits of prompts are prominent when K is small.

Source: Making Pre-trained Language Models Better Few-shot Learners, Gao, et al. 2021

How Many Data Points is a Prompt Worth?

Teven Le Scao, Alexander M. Rush

Setting

- Compare head-based v.s. Prompt-based fine-tuning
- Model: **RoBERTa-large**
- **Manually-designed** prompts

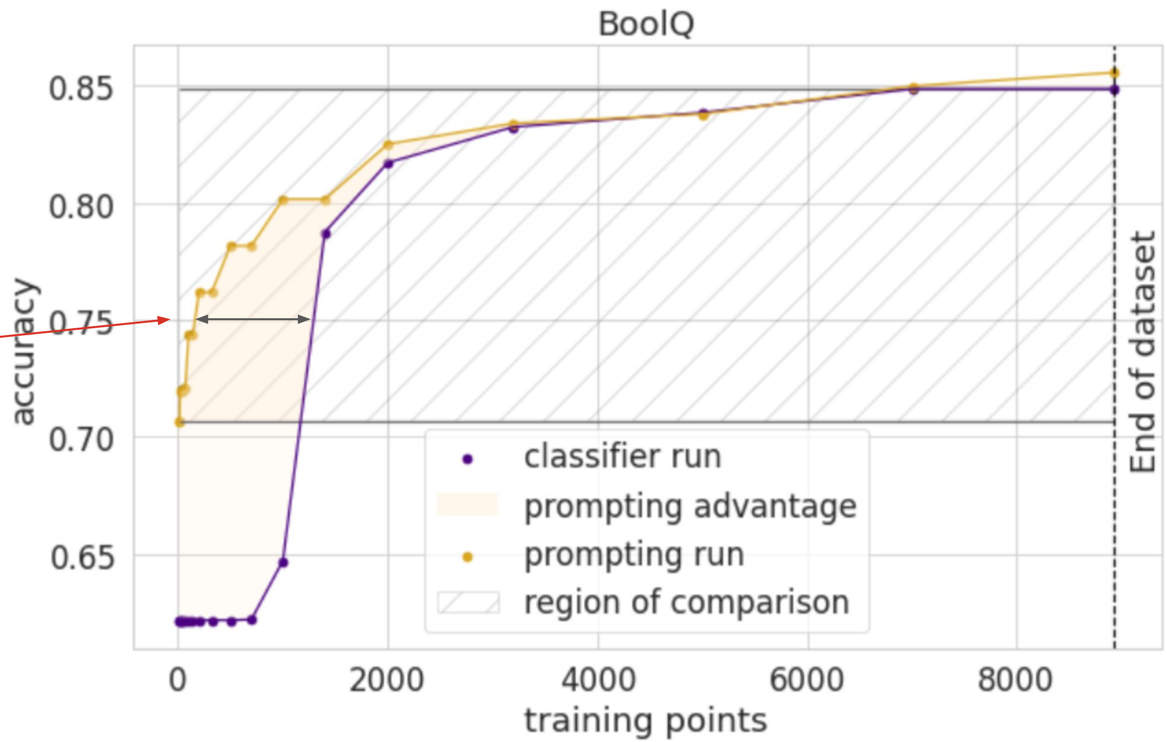
Datasets

7 datasets from SuperGLUE + MNLI.

- Entailment tasks: **CB, MNLI, RTE**
- Multiple-Choice Question Answering: **BoolQ, MultiRC**
- Common-sense Reasoning: **WSC, COPA, WiC**

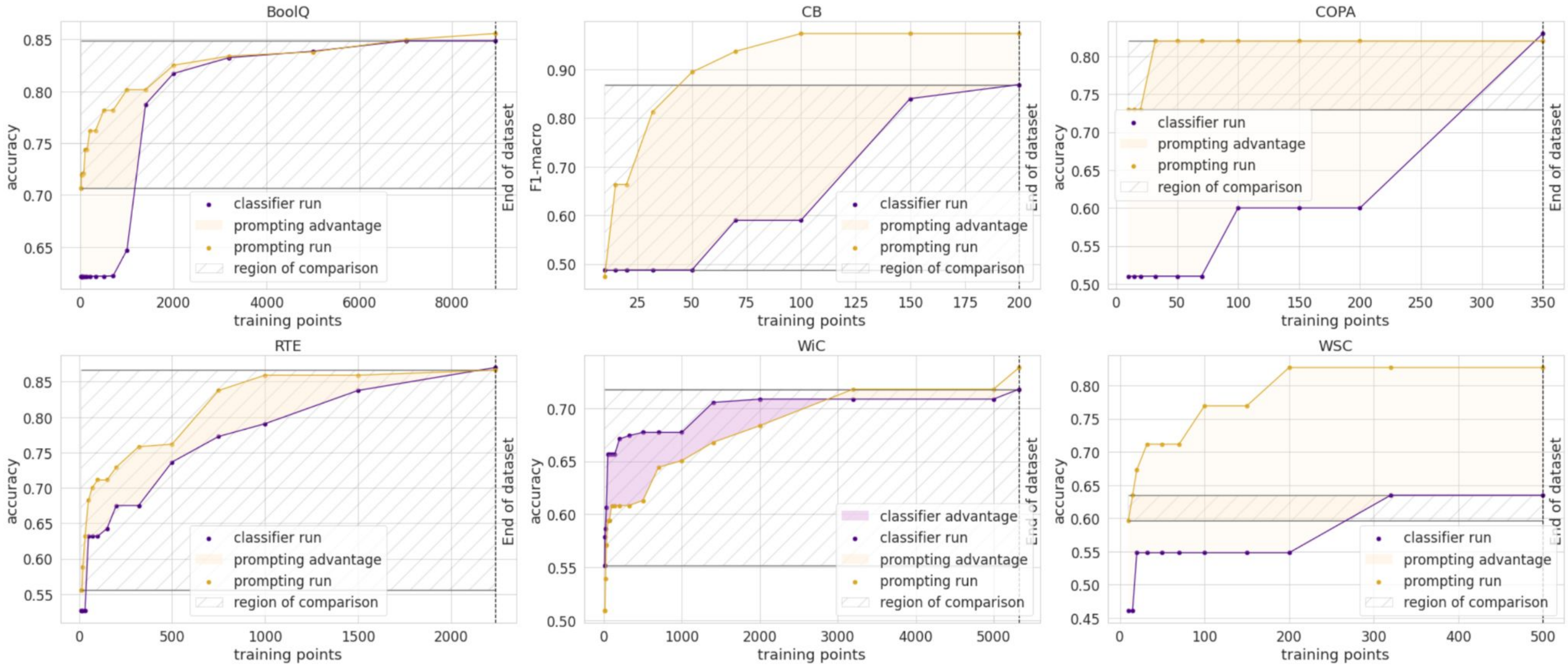
Prompt-based vs head-based

"Data advantage"
@ acc 0.75



Source: *How Many Data Points is a Prompt Worth?, Le Scao & Rush, 2021*

Prompt-based vs head-based



Source: How Many Data Points is a Prompt Worth?, Le Scao & Rush, 2021

Prompt-based vs head-based

Averaged data advantage over different accuracy levels:

Average Advantage (# Training Points)							
MNLI	BoolQ	CB	COPA	MultiRC*	RTE	WiC	WSC
3506 ± 536	752 ± 46	90 ± 2	288 ± 242	384 ± 378	282 ± 34	-424 ± 74	281 ± 137

Source: How Many Data Points is a Prompt Worth?, Le Scao & Rush, 2021

How important is a good prompt?

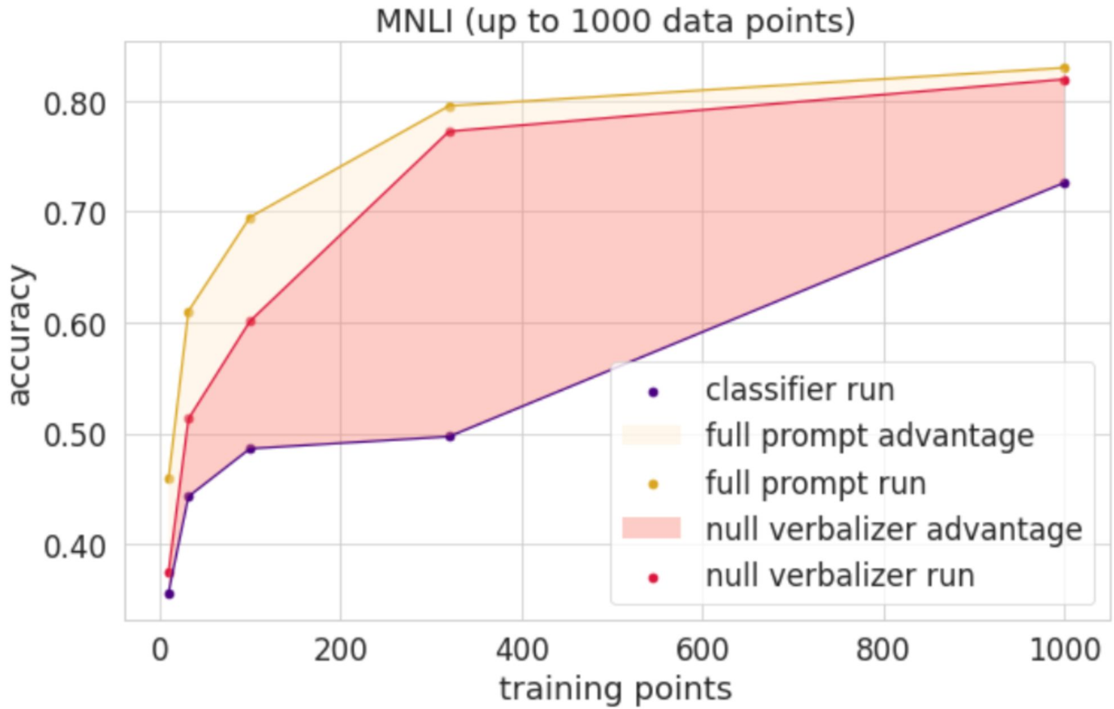
	Average Advantage (# Training Points)							
	MNLI	BoolQ	CB	COPA	MultiRC*	RTE	WiC	WSC
P vs H	3506 ± 536	752 ± 46	90 ± 2	288 ± 242	384 ± 378	282 ± 34	-424 ± 74	281 ± 137
P vs N	150 ± 252	299 ± 81	78 ± 2	-	74 ± 56	404 ± 68	-354 ± 166	-
N vs H	3355 ± 612	453 ± 90	12 ± 1	-	309 ± 320	-122 ± 62	-70 ± 160	-

P = good template + good label words
N = good template + non-sensical label words.
[e.g. Mike -> “Positive”, John -> “Negative”]

H = head-based

Source: How Many Data Points is a Prompt Worth?, Le Scao & Rush, 2021

How important is a good prompt in few-shot setting?



N catches up with **P** when training points are more than ~300

Source: How Many Data Points is a Prompt Worth?, Le Scao & Rush, 2021

Additional Method for Making Prompts

- Round-trip translation (Jiang et al., 2020)
- Replacement of phrases from thesaurus (Yuan et al, 2021)
- Neural prompt re-writer (Haviv et al, 2021)

Prompt Mining
(Jiang et. al, 2020)

Prompt Paraphrasing

Prompt Generation
(Gao et al., 2021)

Reference: Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing Liu, et al.

Ethical Considerations

"LMs ***appear to follow yet do not actually follow users' instructions*** *has important implications, especially considering the* ***increasing commercial use*** *of LMs. While traditional fine-tuned models also pose challenges in interpretability, with prompt-based models, an* ***illusion of instruction*** *following can be more pernicious than having no instructions at all"*

Source: Do Prompt-Based Models Really Understand the Meaning of Their Prompts? Webson, et al.

Credits and Special Thanks!

Professor Chen
Alexander Wettig
Tianyu Gao

***Q3**:* We already know that finding a good prompt is so important. Sometimes, it is also **challenging to find prompts that are natural and fit in pre-trained distributions.** For example, <S1> ? [MASK] , <S2>, the chance that "Maybe" can fill in [MASK] is very low (this is the prompt used for NLI tasks in Gao et al., 2021). **Do you have any ideas about how to improve this and find better prompts?**

Additional Prompt Engineering Methods (discrete / hard prompts)

Prompt Mining

Jiang et al. (2020) uses a mining-based method to automatically find templates given a set of training inputs x and y. Scrapes a large text corpus (e.g. Wikipedia) for strings containing x and y, and finds middle words or dependency paths between the inputs and outputs.

Prompt Paraphrasing

Takes an existing prompt, paraphrases into other prompts, and uses the prompt that achieves the best result. Prompt paraphrasing can be done with multiple methods including round-trip translation (Jiang et al., 2020); replacement of phrases from thesaurus (Yuan et al, 2021) and a neural prompt re-writer (Haviv et al, 2021)

Prompt Generation:

Gao et al. (2021) introduces pre-trained T5 seq to seq to fill in missing spans and generate template tokens. Ben-David et al. (2021) builds upon this method in introducing a domain adaptation algorithm that trains T5 to generate unique domain relevant features.

Reference: Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing Liu, et al.