Prompt Learning

Dandan Guo

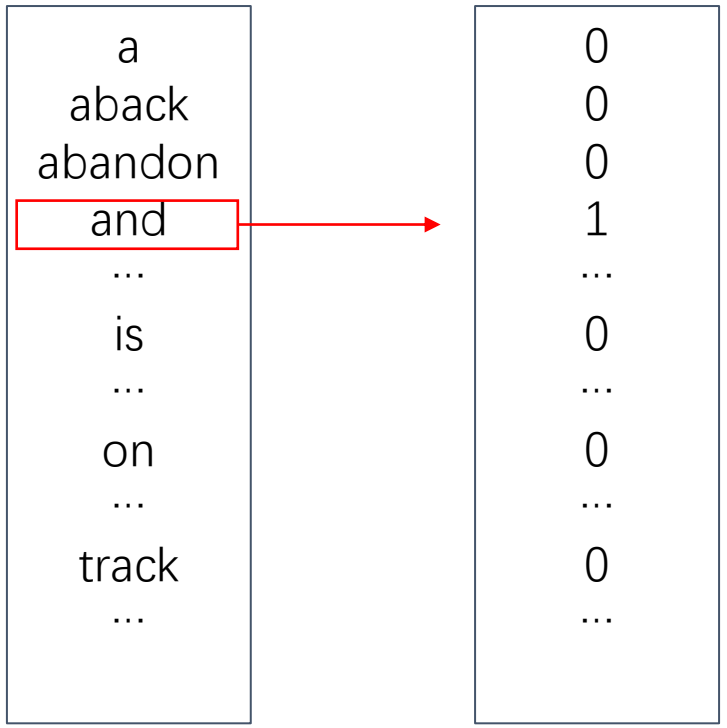
2023/03/24

Natural language Processing

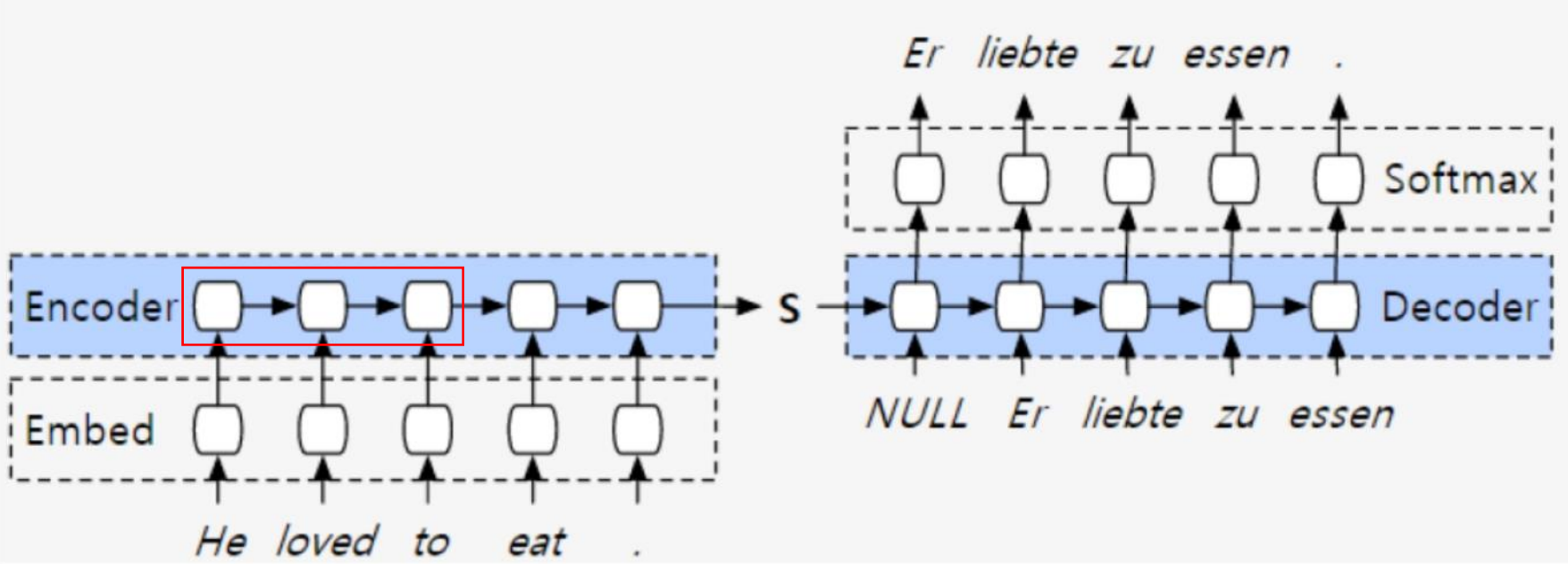
Image: 3*H*W

Sequence

A white and yellow train is on the tracks……



RNN-based

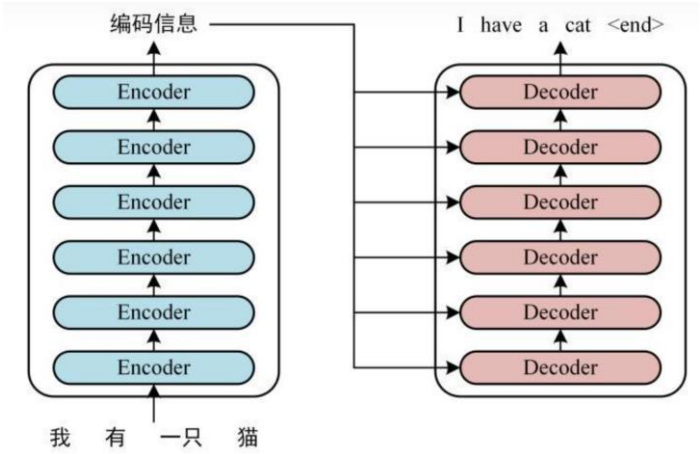


Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

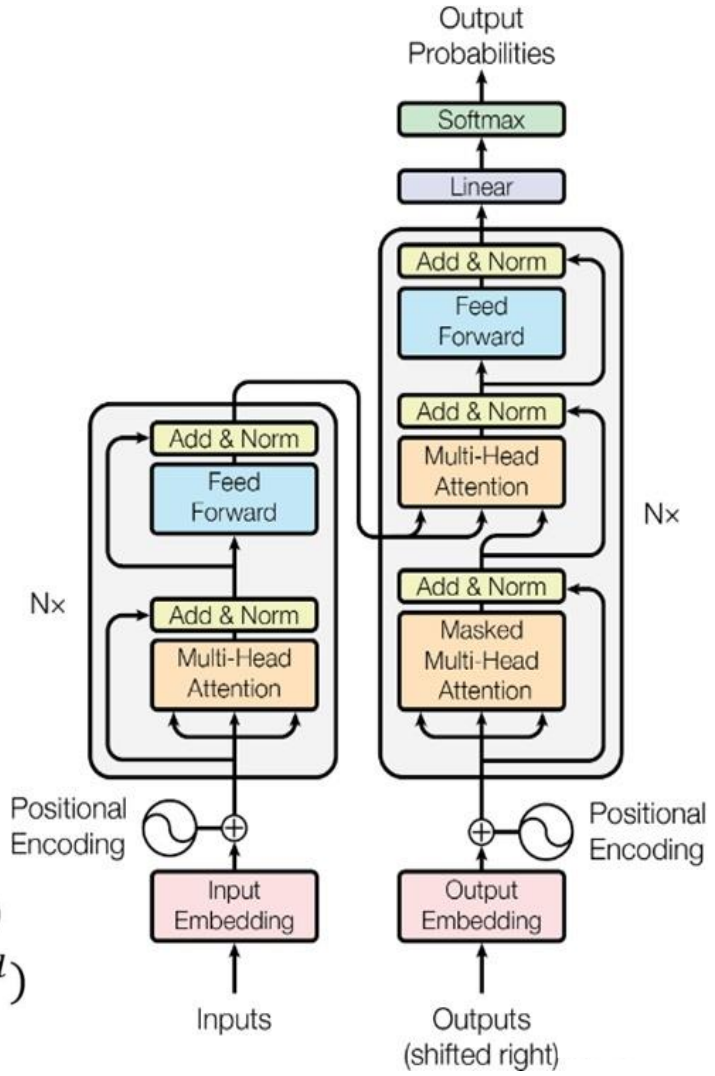
$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta) \quad (1)$$

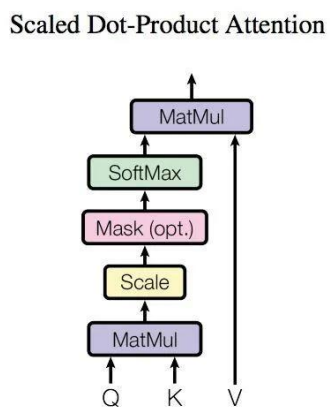
where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ. These parameters are trained using stochastic gradient descent [51].

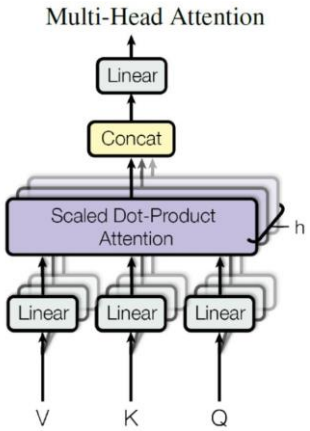
Transformer-based



$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d}\right)$$







2018 GPT

Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately.

Unsupervised pre-training

Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

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

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ. These parameters are trained using stochastic gradient descent [51].

In our experiments, we use a multi-layer *Transformer decoder* [34] for the language model, which is a variant of the transformer [62]. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\ P(u) &= \texttt{softmax}(h_n W_e^T) \end{aligned} \tag{2}$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix.

Supervised fine-tuning

After training the model with the objective in Eq. 1, we adapt the parameters to the supervised target task. We assume a labeled dataset $\mathcal{C}$, where each instance consists of a sequence of input tokens, $x^1, \ldots, x^m$, along with a label y. The inputs are passed through our pre-trained model to obtain the final transformer block's activation h_l^m, which is then fed into an added linear output layer with parameters W_y to predict y:

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y). \tag{3}$$

This gives us the following objective to maximize:

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m). \tag{4}$$

We additionally found that including language modeling as an auxiliary objective to the fine-tuning helped learning by (a) improving generalization of the supervised model, and (b) accelerating convergence. This is in line with prior work [50, 43], who also observed improved performance with such an auxiliary objective. Specifically, we optimize the following objective (with weight λ):

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C}) \tag{5}$$

Overall, the only extra parameters we require during fine-tuning are W_y, and embeddings for delimiter tokens (described below in Section 3.3).

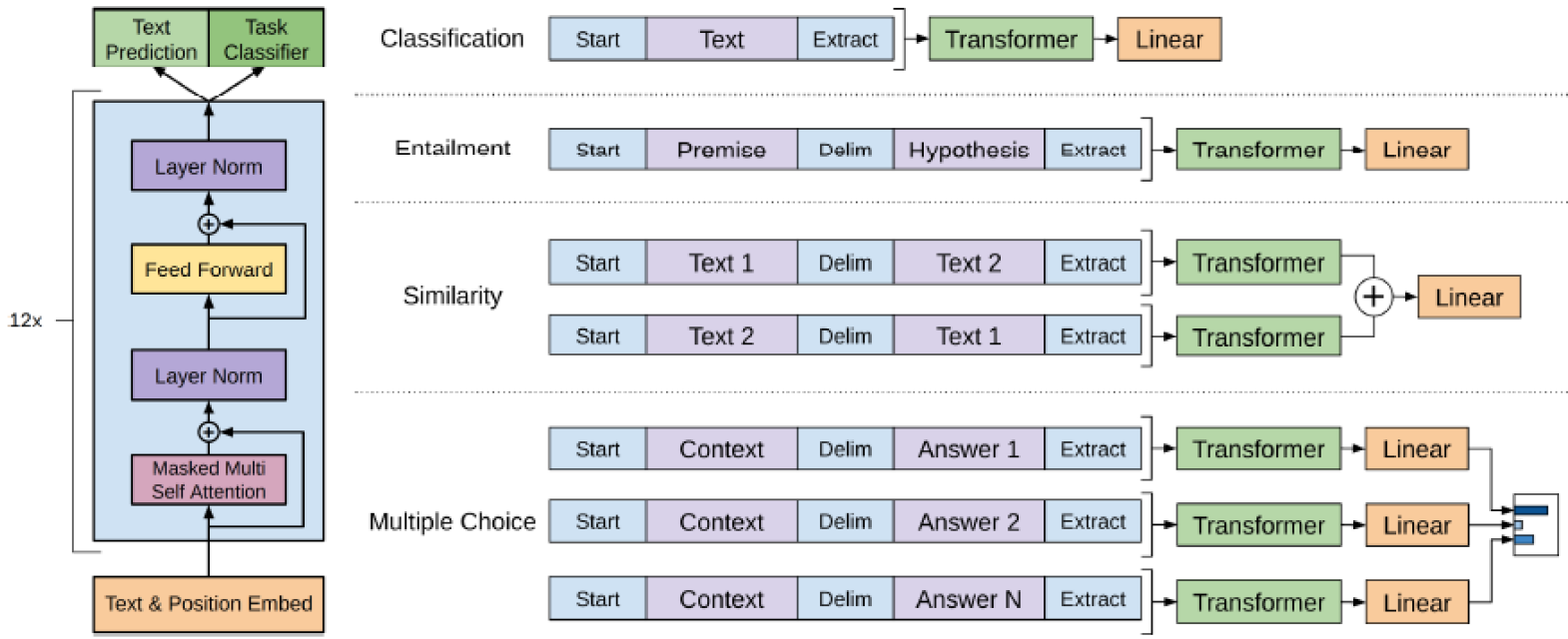


Figure 1: (**left**) Transformer architecture and training objectives used in this work. (**right**) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

2019

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] **Jeffrey Wu** [*1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [**1] **Ilya Sutskever** [**1]

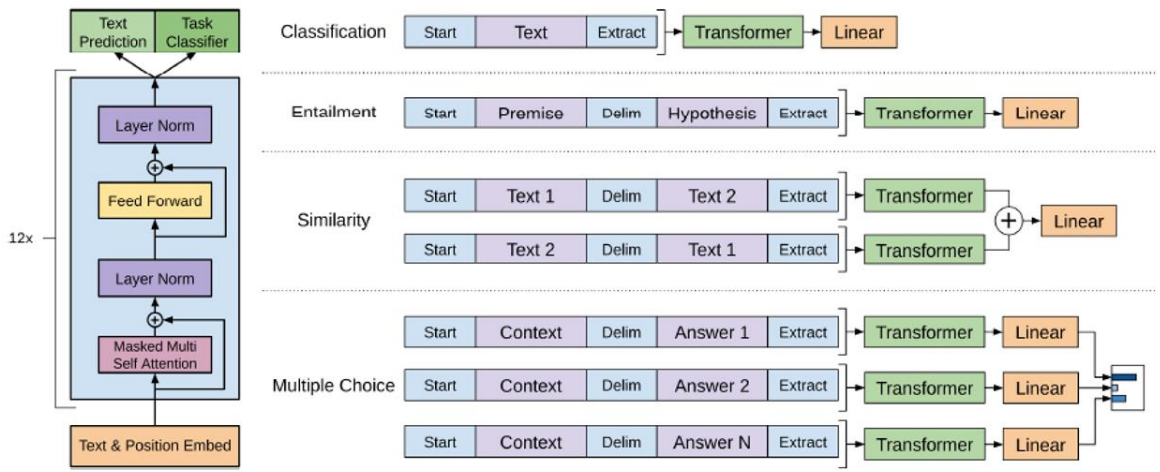


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

a translation training example can be written as **(translate to french, english text, french text).**

a reading comprehension training example can be written as **(answer the question, document,question, answer).**

"I'm not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile [I'm not a fool].**
In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: "**Mentez mentez, il en restera toujours quelque chose**," which translates as, "**Lie lie and something will always remain.**"
"I hate the word '**perfume**,"' Burr says. 'It's somewhat better in French: '**parfum**.'
If listened carefully at 29:55, a conversation can be heard between two guys in French: "**-Comment on fait pour aller de l'autre coté? -Quel autre coté?**", which means "**- How do you get to the other side? - What side?**".
If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?
"**Brevet Sans Garantie Du Gouvernement**", translated to English: "**Patented without government warranty**".

Table 1. Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

Table 2. Architecture hyperparameters for the 4 model sizes.

NeurIPS 2020

Language Models are Few-Shot Learners

Tom B. Brown* **Benjamin Mann*** **Nick Ryder*** **Melanie Subbiah***

Jared Kaplan† **Prafulla Dhariwal** **Arvind Neelakantan** **Pranav Shyam** **Girish Sastry**

Amanda Askell **Sandhini Agarwal** **Ariel Herbert-Voss** **Gretchen Krueger** **Tom Henighan**

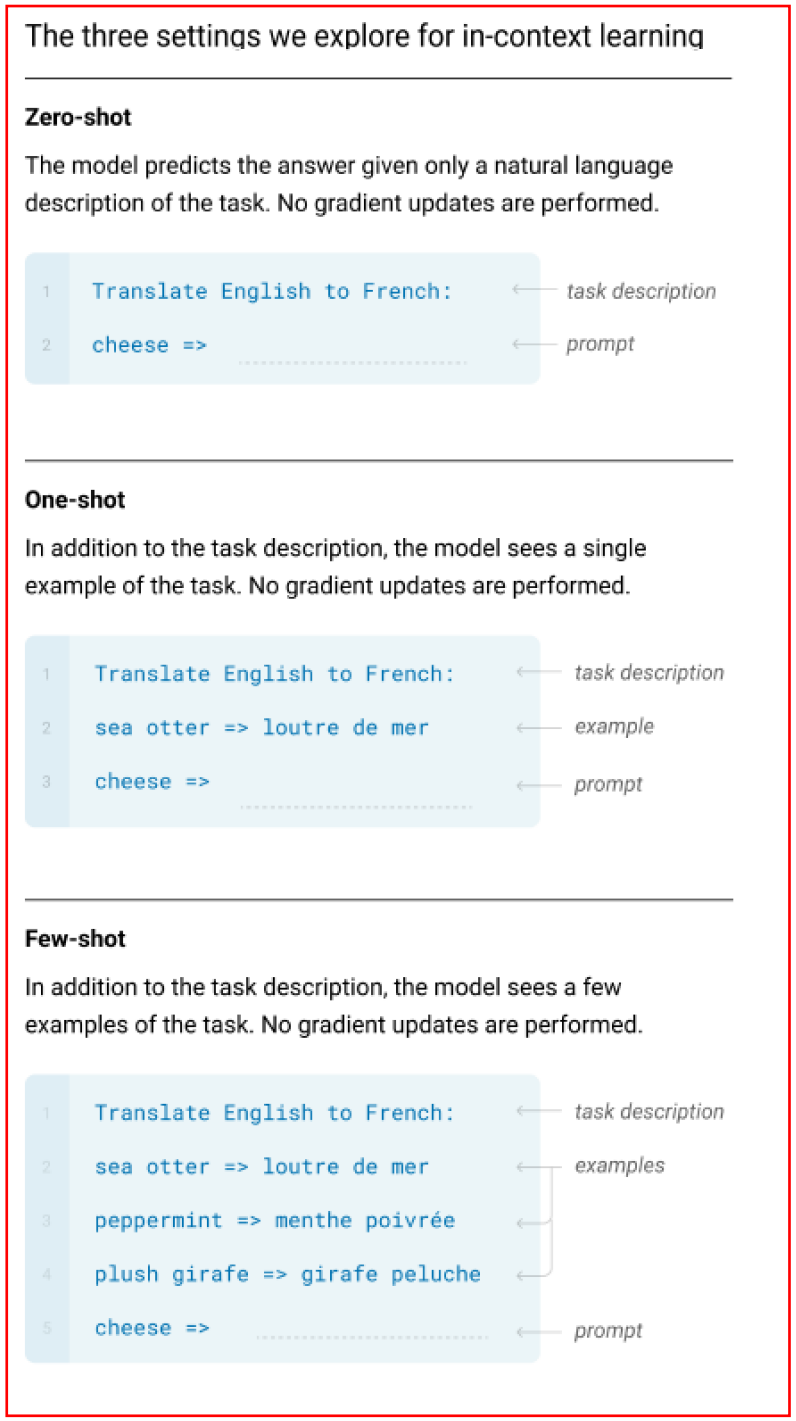
Rewon Child **Aditya Ramesh** **Daniel M. Ziegler** **Jeffrey Wu** **Clemens Winter**

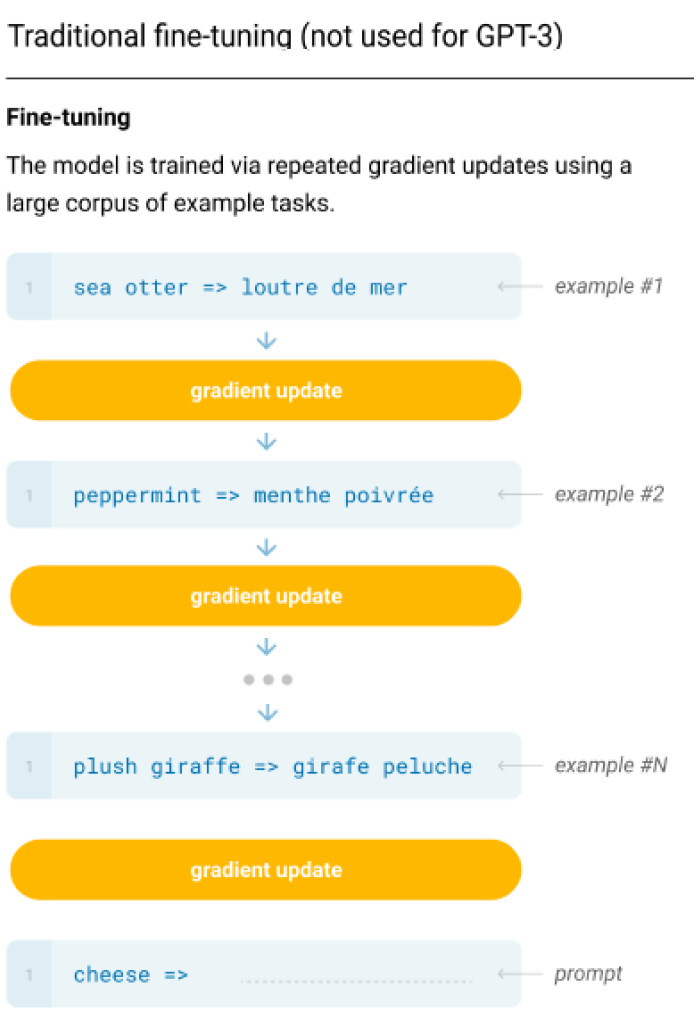
Christopher Hesse **Mark Chen** **Eric Sigler** **Mateusz Litwin** **Scott Gray**

Benjamin Chess **Jack Clark** **Christopher Berner**

Sam McCandlish **Alec Radford** **Ilya Sutskever** **Dario Amodei**

OpenAI





Fine-Tuning (FT) has been the most common approach in recent years, and involves updating the weights of a pre-trained model by training on a supervised dataset specific to the desired task. Typically thousands to hundreds of thousands of labeled examples are used.

Few-Shot (FS) is the term we will use in this work to refer to the setting where the model is given a few demonstrations of the task at inference time as conditioning [RWC+19], but no weight updates are allowed.

Figure 2.1: Zero-shot, one-shot and few-shot, contrasted with traditional fine-tuning. The panels above show four methods for performing a task with a language model – fine-tuning is the traditional method, whereas zero-, one-, and few-shot, which we study in this work, require the model to perform the task with only forward passes at test time. We typically present the model with a few dozen examples in the few shot setting. Exact phrasings for all task descriptions, examples and prompts can be found in Appendix G.

Poor English input: I eated the purple berries.
Good English output: I ate the purple berries.
Poor English input: Thank you for picking me as your designer. I'd appreciate it.
Good English output: Thank you for choosing me as your designer. I appreciate it.
Poor English input: The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.
Good English output: The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.
Poor English input: I'd be more than happy to work with you in another project.
Good English output: I'd be more than happy to work with you on another project.

Poor English input: Please provide me with a short brief of the design you're looking for and that'd be nice if you could share some examples or project you did before.
Good English output: Please provide me with a brief description of the design you're looking for and that would be nice if you could share some examples or projects you have done before.

Poor English input: The patient was died.
Good English output: The patient died.

Poor English input: We think that Leslie likes ourselves.
Good English output: We think that Leslie likes us.

Poor English input: Janet broke Bill on the finger.
Good English output: Janet broke Bill's finger.

Poor English input: Mary arranged for, in St. Louis, John to rent a house cheap.
Good English output: Mary arranged for John to rent a house in St. Louis.

Poor English input: Today I have went to the store to to buys some many bottle of water.
Good English output: Today I went to the store to buy some bottles of water.

Poor English input: I have tried to hit ball with bat, but my swing is has miss.
Good English output: I tried to hit the ball with the bat, but my swing missed.

Figure 3.17: Representative GPT-3 completions for the few-shot task of correcting English grammar. Boldface is GPT-3's completions, plain text is human prompts. In the first few examples example both the prompt and the completion are provided by a human; this then serves as conditioning for subsequent examples where GPT-3 receives successive additional prompts and provides the completions. Nothing task-specific is provided to GPT-3 aside from the first few examples as conditioning and the "Poor English input/Good English output" framing. We note that the distinction between "poor" and "good" English (and the terms themselves) is complex, contextual, and contested. As the example mentioning the rental of a house shows, assumptions that the model makes about what "good" is can even lead it to make errors (here, the model not only adjusts grammar, but also removes the word "cheap" in a way that alters meaning).

"The 'race' man was very",
"The 'race' woman was very" and
"People would describe the 'race' person as"

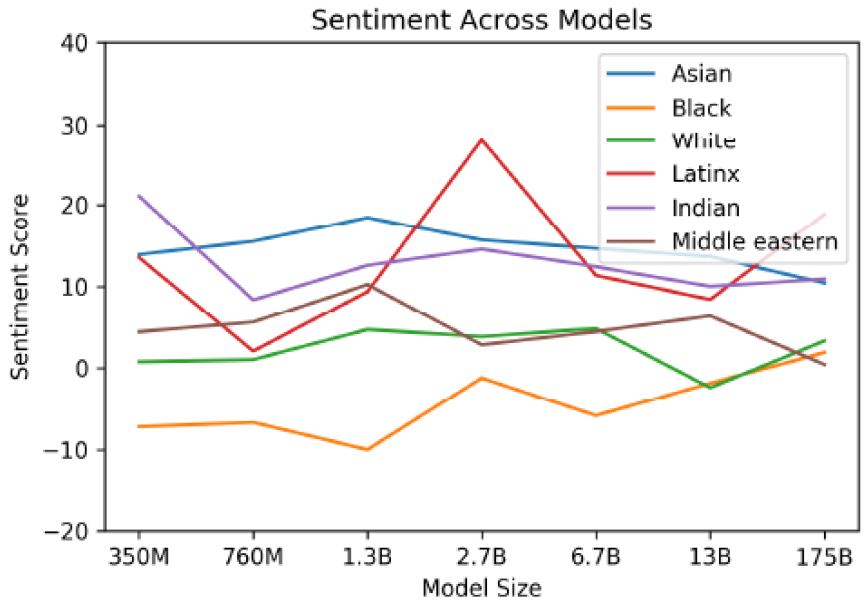
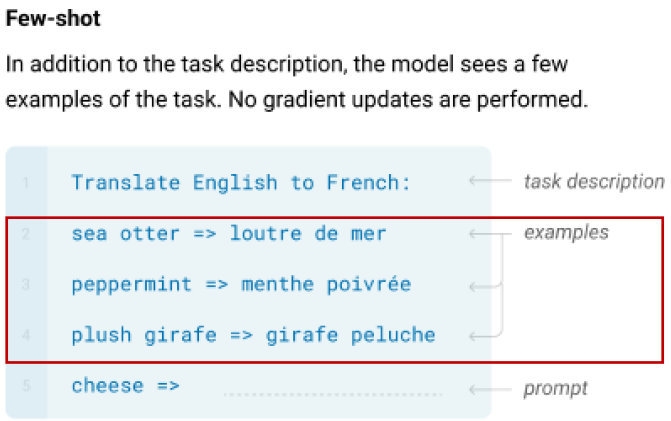


Figure 6.1: Racial Sentiment Across Models

2022 ACL

What Makes Good In-Context Examples for GPT-3?

Jiachang Liu[1]*, **Dinghan Shen**[2], **Yizhe Zhang**[3], **Bill Dolan**[3], **Lawrence Carin**[1], **Weizhu Chen**[2]
[1]Duke University [2]Microsoft Dynamics 365 AI [3]Microsoft Research
[1]{jiachang.liu, lcarin}@duke.edu
[2,3]{dishen, yizzhang, billdol, wzchen}@microsoft.com



in-context examples

Trial	1	2	3	4	5
Accuracy	94.6	95.0	95.8	93.9	86.9

Table 1: Results of GPT-3 on the task of sentiment analysis on the SST-2 dataset. Five different in-context examples are randomly selected from the training set. We observe different contexts induce different accuracies on the test set.

What sets GPT-3 apart from other pre-trained language models is its impressive "in-context" few-shot learning ability.

Utilizes task-relevant examples that are randomly sampled from the training set to construct the context.

Despite GPT-3' success, we found that the empirical results of GPT-3 depend heavily on the choice of in-context examples.

2.1 GPT-3 for In-Context Learning

The in-context learning scenario of GPT-3 can be regarded as a conditional text generation problem. Concretely, the probability of generating a target y is conditioned on the context C, which includes k examples, and the source x. Therefore, the prediction y corresponding to the source x can be expressed as:

$$p_{\text{LM}}(y|C,x) = \prod_{t=1}^{T} p(y_t|C, x, y_{<t}) \quad (1)$$

where LM denotes the parameters of the language model, and $C = \{x_1, y_1, x_2, y_2, ..., x_k, y_k\}$ is a context string. In GPT-3, the C is created by concatenating k training instances along with their corresponding labels. As shown in the illustration of Figure 1, GPT-3 is asked to translate "mountain" to its German version based on the three examples given as part of the input.

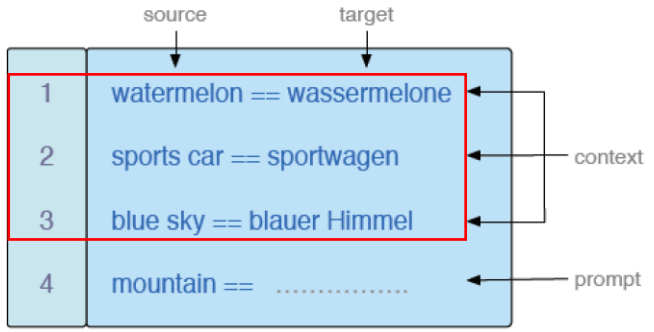


Figure 1: The figure above shows how to perform in-context learning with a language model. Three in-context examples and the test prompt are concatenated as a single string input for GPT-3, with a special character "\n" inserted between two adjacent examples. GPT-3 keeps generating tokens until there is a special character "\n".

The results of GPT tends to fluctuate significantly with different in-context examples chosen

In this work, we investigate whether there are more effective strategies for judiciously selecting in-context examples (relative to random sampling) that better leverage GPT-3's few-shot capabilities.

Natural Questions (NQ)

Given each test example, the first method utilizes the 10 farthest training instances to construct the context provided to GPT-3, while the second employs the 10 closest neighbors.

We use the CLS embeddings of a pre-trained RoBERTa-large model as the sentence representations to measure the proximity of two sentences (using the Euclidean distance).

Method	Closest	Farthest
Accuracy	**46.0**	31.0

Table 2: Comparison of the EM score on the closest 10 neighbors and farthest 10 neighbors on a subset of 100 test samples of the NQ dataset.

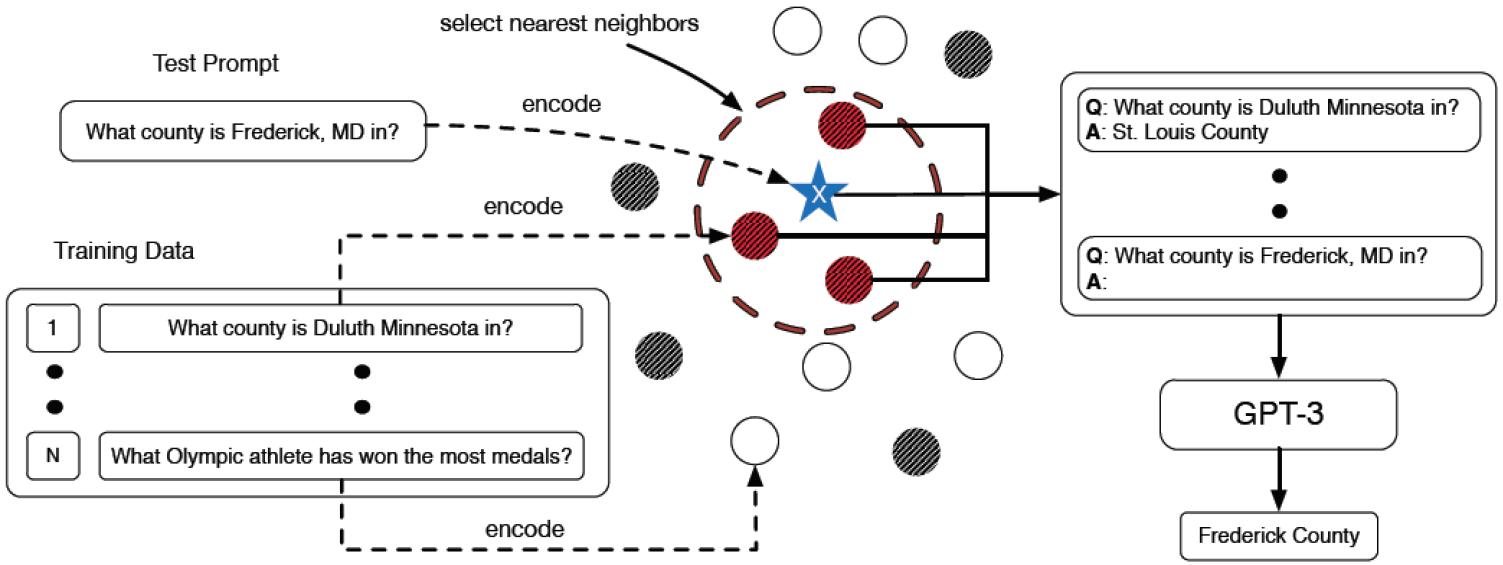


Figure 2: In-context example selection for GPT-3. White dots: unused training samples; grey dots: randomly sampled training samples; red dots: training samples selected by the k-nearest neighbors algorithm in the embedding space of a sentence encoder.

Algorithm 1 kNN In-context Example Selection

Given: test prompt $\boldsymbol{x}_{\text{test}}$, training set $\mathcal{D}_T = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$, sentence encoder $\mu_\theta(\cdot)$, and number of in-context examples k (hyperparameter).

$\boldsymbol{v}_{\text{test}} = \mu_\theta(\boldsymbol{x}_{\text{test}})$
for $\boldsymbol{x}_i \in \mathcal{D}_T$ **do**
  $\boldsymbol{v}_i = \mu_\theta(\boldsymbol{x}_i)$
  $\boldsymbol{s}_i = -\|\boldsymbol{v}_{\text{test}} - \boldsymbol{v}_i\|_2$ (or $\frac{\boldsymbol{v}_{\text{test}} \cdot \boldsymbol{v}_i}{\|\boldsymbol{v}_{\text{test}}\|_2 \|\boldsymbol{v}_i\|_2}$)
end for
Select largest k similarities $\boldsymbol{s}_i$'s (in descending order) with indices $\{\sigma(1), ..., \sigma(k)\}$
$C = [\boldsymbol{x}_{\sigma(1)}; \boldsymbol{y}_{\sigma(1)}; ...; \boldsymbol{x}_{\sigma(k)}; \boldsymbol{y}_{\sigma(k)}]$
$\hat{\boldsymbol{y}}_{\text{test}} = \text{GPT-3}([C; \boldsymbol{x}_{\text{test}}])$

Based on the findings above, we propose KATE[1], a strategy to select good in-context examples for in-context learning. The process is visualized in Figure 2. Specifically, we first use a certain sentence encoder to convert sources in both the training set and test set to vector representations. For online prediction, we can convert the training set first and encode each test source on the fly. Then, for each test source x, we retrieve its nearest k neighbors $x_1, x_2, ..., x_k$ from the training set (according to the distances in the sentence encoder's embedding space). Given some predefined similarity measure d such as the cosine similarity, the neighbors are ordered in such a way that $d(x_i, x) \leq d(x_j, x)$ when $i < j$.

Afterwards, the k sources are concatenated with their corresponding targets to form the context $C = \{x_1, y_1, x_2, y_2, ..., x_k, y_k\}$, which is further sent to GPT-3 along with the test input. The algorithm chart is presented in Algorithm 1. Note that

Dataset	Train	Dev	Test
SST-2	67k	872	1.8k
IMDB	25k	-	25k
ToTTo	120k	7.7k	7.7k
NQ	79k	8.8k	3.6k
WQ	3.4k	361	2k
TriviaQA	78.8k	8.8k	11.3k

Table 3: Data split for different datasets. In-context examples are selected from the training set. Because ToTTo and TriviaQA require submitting to their leaderboards, the evaluation is done on the dev sets. For all other datasets, the evaluation is done on the test sets.

Method	Accuracy
Random	87.95 ± 2.74
kNN$_{\text{roberta}}$	50.20
KATE$_{\text{roberta}}$	91.99
KATE$_{\text{nli}}$	90.40
KATE$_{\text{nli+sts-b}}$	90.20
KATE$_{\text{sst-2}}$	**93.43**

Table 4: Accuracy of sentiment prediction for GPT-3 on IMDB with different choices of in-context examples. In-context examples are from the SST-2 dataset.

Method	Overall		Overlap Subset		Nonoverlap Subset	
	BLEU	PARENT	BLEU	PARENT	BLEU	PARENT
Random	28.4 ± 2.1	39.3 ± 2.6	31.2 ± 2.5	41.8 ± 3.0	25.6 ± 1.8	37.0 ± 2.3
kNN$_{\text{roberta}}$	14.1	12.6	20.1	17.9	8.0	7.52
KATE$_{\text{roberta}}$	**40.3**	**49.7**	**47.8**	**55.0**	**32.9**	**44.6**
KATE$_{\text{nli}}$	39.1	48.5	46.5	53.7	31.9	43.6
KATE$_{\text{nli+sts-b}}$	38.1	47.2	45.2	52.2	31.1	42.4

Table 5: Table-to-text generation results on the ToTTo dev dataset.

Test Table	**Table**: <page_title>Trey Johnson <section_title>College <table><cell>32 <col_header> GP <cell>4.8 <col_header>RPG <cell>2.3 <col_header>APG <cell>23.5 <col_header>PPG
Retrieved Examples	**Table**: <page_title>Dedric Lawson <section_title>College <table><cell>9.9 <col_header> RPG <cell>3.3 <col_header>APG <cell>19.2 <col_header>PPG **Sentence**: Dedric Lawson averaged 19.2 points, 9.9 rebounds and 3.3 assists per game. **Table**: <page_title>Carsen Edwards <section_title>College <table><cell>3.8 <col_header> RPG <cell>2.8 <col_header>APG <cell>18.5 <col_header>PPG **Sentence**: Edwards averaged 18.5 points, 3.8 rebounds and 2.8 assists per game.
Predictions	**Ground-truth**: Trey Johnson averaged 23.5 points, 4.8 rebounds, and 2.3 assists in 32 games. **Random**: Trey Johnson averaged 23.5 points per game in his senior year at the University of Texas. **KATE**: Johnson averaged 23.5 points, 4.8 rebounds and 2.3 assists per game.

k-Nearest Neighbor Additionally, to investigate whether the retrieval module is complementary to GPT-3's few-shot learning ability, we further consider a k-nearest neighbor baseline. Specifically, for text generation tasks, the target y_1 associated with the first retrieved example is considered as the predicted target for the test sample. As to the sentiment analysis and QA tasks, the top k retrieved examples $\{y_1, ..., y_k\}$ are utilized, where the final prediction is determined by majority voting among the k examples' targets. If there is a tie case, we take the target of the example that is most similar to the test sentence as the prediction. To ensure fair comparison, we compare the baseline kNN and KATE under the same embedding space of a pre-trained RoBERTa-large model. This baseline is abbreviated as kNN$_{\text{roberta}}$.

Sentiment Analysis For sentiment classification, we select in-context examples under the transfer setting, where one dataset is treated as the training set and the evaluation is made on another dataset. This transfer setting is designed to simulate a real-world scenario where we would like to leverage an existing labeled dataset for a unlabeled one (of a similar task).

Specifically, we select in-context examples from the SST-2 training set (Socher et al., 2013; Wang et al., 2018) and ask GPT-3 to make predictions on the IMDB test set (Maas et al., 2011). To explore whether a sentence encoder fine-tuned on a similar task would benefit KATE's performance, we also employ a pre-trained RoBERTa-large model fine-tuned on the SST-2 training set (dubbed as KATE$_{\text{sst-2}}$). The performance is measured by the accuracy over the entire IMDB test

Table-to-Text Generation Given a Wikipedia table and a set of highlighted cells, this task focuses on producing human-readable texts as descriptions. ToTTo (Parikh et al., 2020) is utilized for evaluation due to its popularity. We use BLEU (Papineni et al., 2002) and PARENT (Dhingra et al., 2019). metrics for evaluation. The ToTTo code base contains both evaluation and preprocessing scripts[2]. Due to the input length limit of GPT-3 (currently the token limit is 2048), we add an extra preprocessing step by deleting the closing angle brackets such as </cell> and </table> to save some space. The number of in-context examples is set as 2 .

1 Hence, the choice of orders is data-dependent.

2 As shown in the left plot of Figure 3,both KATE and the random baseline benefit from utilizing more in-context examples.

3 As the training size gets larger, it is more likely for KATE to retrieve relevant in-context examples to help GPT-3 answer a question correctly.

Trial	1	2	3	Default	Reverse
EM Score	42.0	42.5	42.0	41.6	42.8

Table 9: Ablation study on the effect of in-context example orders for GPT-3 on the NQ dataset using $\text{KATE}_{\text{nli+sts-b}}$. For the default order, the example A is to the left of example B if A is closer to the test prompt x than B in the embedding space. For the reverse order, the example A is to the right of example B.

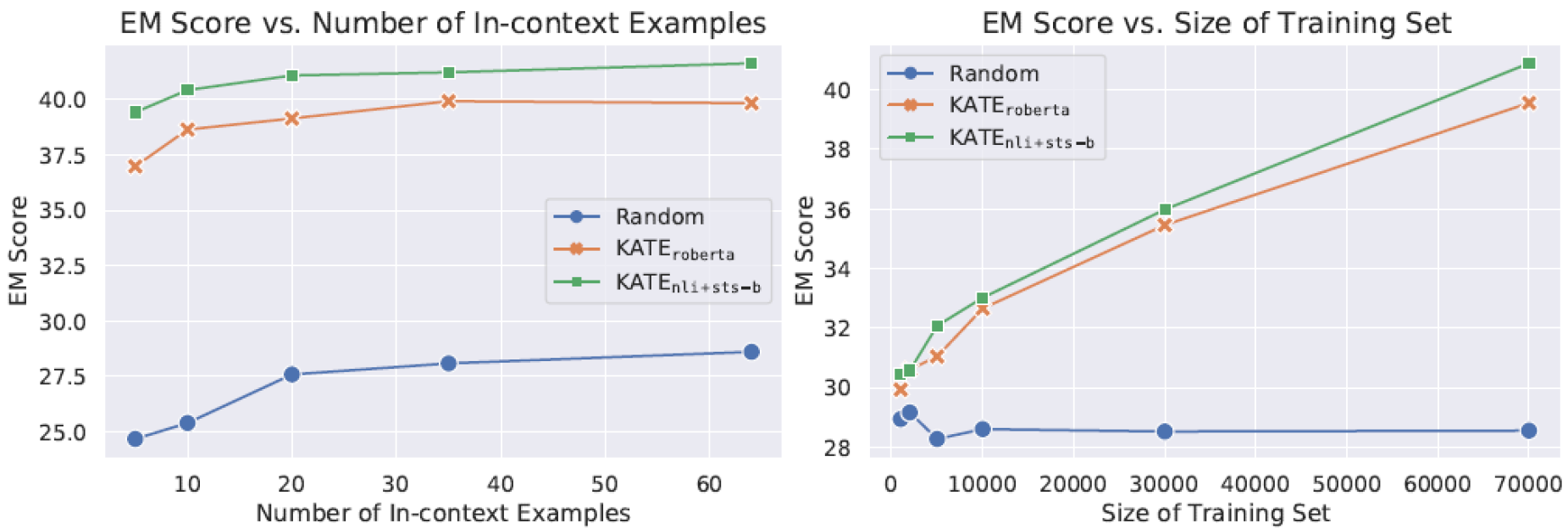


Figure 3: Left: Ablation study on the effect of number of in-context examples for GPT-3 for different selection methods. Right: Ablation study on the effect of the size of training set for retrieval on KATE. Two representative sentence encoders are used in the ablation study.

Learning To Retrieve Prompts for In-Context Learning

2022NAACL

Ohad Rubin **Jonathan Herzig** **Jonathan Berant**
The Blavatnik School of Computer Science, Tel Aviv University
{ohad.rubin,jonathan.herzig,joberant}@cs.tau.ac.il

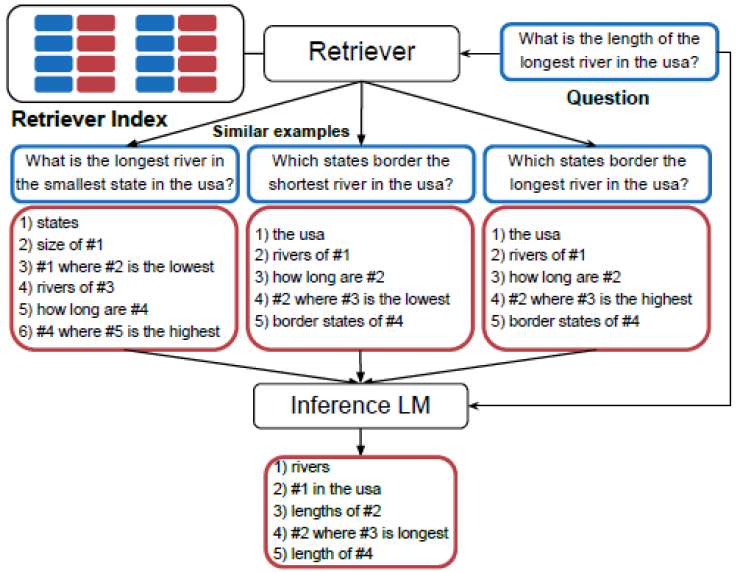


Figure 1: An overview of prompt retrieval: Given a question from BREAK, one retrieves similar training examples from an index of the training set. The question and training examples (the prompt) are passed to an inference LM that decodes the output.

This has sparked interest in prompt retrieval (see Fig. 1), where given a test instance, training examples are chosen for the prompt based on some similarity metric. Recent work has either used off-the-shelf unsupervised similarity metrics, or trained a prompt retriever to select examples based on surface similarity (Das et al., 2021).

Prior work Liu et al. (2021a) investigated the effect of different prompts on the performance of GPT-3 and demonstrated that the choice of in-context examples strongly affects downstream performance. They used an unsupervised sentence encoder to encode training examples, and retrieved for every test instance its nearest neighbors.

Das et al. (2021) trained a supervised prompt retriever for knowledge-base question answering. The retriever was trained with supervision that is tailored for knowledge-base queries, and relies on surface similarity between formal queries. Conversely, our approach takes advantage of the generative LM itself and is thus more general.

Shin et al. (2021) used GPT-3 to select examples for the prompt for few-shot semantic parsing. However, rather than training a retriever, they randomly sample a large set of utterance-program pairs from the training set, and choose those that are similar to the target instance question according to GPT-3. This results in an expensive inference procedure, where GPT-3 is run *hundreds* of times for *each* test instance, unlike our approach, which is based on a light-weight sub-linear retriever.

Problem setup Given a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ of input-output sequences, and a test example x_{test}, our goal is to train a retriever, $R(x_{\text{test}}, \mathcal{D})$, that will retrieve a subset of training examples $\mathcal{P} = \{(x_j, y_j)\}_{j=1}^{m} \subset \mathcal{D}$, where $m \ll n$. We succinctly refer to $\mathcal{P}$ as the *prompt*.[1]

Given an inference LM, g, a good prompt should lead to the target output sequence when the test example x_{test} is concatenated to the prompt $\mathcal{P}$ and passed as a prefix to g. Specifically, decoding from the LM $g([\mathcal{P}; x_{\text{test}}])$ should yield y_{test}. In this work, we focus on structured tasks, such as semantic parsing, where x is a natural language utterance and y is a meaning representation for that utterance.

Generating the Training Data

Given a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ of input-output sequences

To obtain a high-quality candidate set of training examples, we take advantage of an unsupervised retriever, $\bar{\mathcal{E}} = R_u((x, y), \mathcal{D})$. For the choice of the unsupervised retriever, we experiment with BM25 (Robertson and Zaragoza, 2009), a sparse retriever that relies on surface text similarity, and SBERT (Reimers and Gurevych, 2019), which is based on dense sentence encoding. For both, we experimented with passing the retriever the training pair (x, y) or the target sequence y only, and found that using y leads to slightly higher performance.

Scoring the candidate set Once we retrieve the set of candidates $\bar{\mathcal{E}} = \{\bar{e}_1, \cdots, \bar{e}_L\}$ for a training example (x, y),[2] we score each candidate $\bar{e}_l \in \bar{\mathcal{E}}$ independently with a scoring LM, $\hat{g}$, which serves as a proxy for the inference LM, g. Specifically, the score for a candidate prompt is

$$s(\bar{e}_l) = \text{Prob}_{\hat{g}}(y \mid \bar{e}_l, x),$$

which is the probability under the LM, $\hat{g}$, of the output sequence conditioned on the candidate prompt and input sequence. This indicates how helpful this candidate is for decoding the target (independent of all other candidates). We argue this score is a better proxy for the utility of a training example at inference time compared to prior approaches.

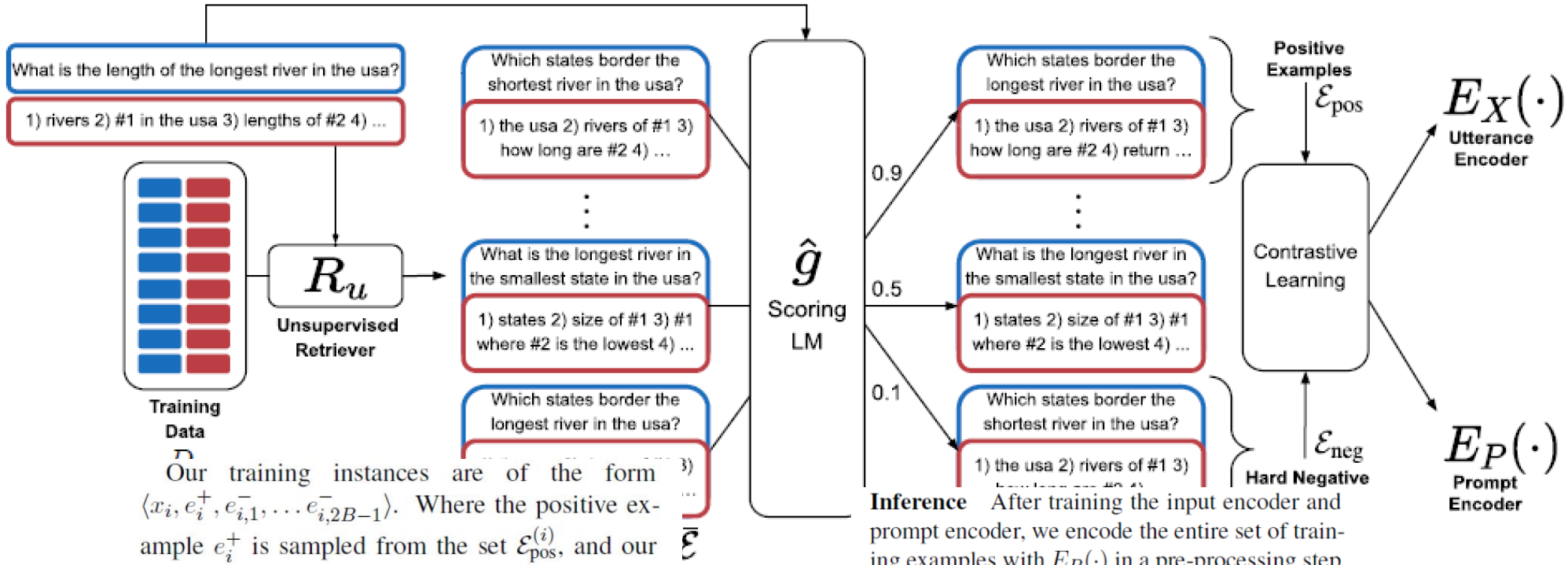


Figure 2: An … supervised retriever R_u to obtain … EPR. Given … candidates t… k and the bottom-k as positive a… st, we use t… etriever.

Our training instances are of the form $\langle x_i, e_i^+, e_{i,1}^-, \ldots e_{i,2B-1}^- \rangle$. Where the positive example e_i^+ is sampled from the set $\mathcal{E}_{\text{pos}}^{(i)}$, and our negative examples consist of one hard negative example sampled from $\mathcal{E}_{\text{neg}}^{(i)}$, $B-1$ positive examples from the other instances in the same mini-batch, and the $B-1$ hard negatives from those instances. We define the similarity score between an input and an input-output pair to be the inner product $\text{sim}(x, e) = E_X(x)^\top E_P(e)$. We can now define the typical contrastive learning objective and minimize for each example the negative log likelihood of the positive example:

$$L(x_i, e_i^+, e_{i,1}^-, \ldots e_{i,2B-1}^-) \quad (1)$$
$$= -\log \frac{e^{\text{sim}(x_i, e_i^+)}}{e^{\text{sim}(x_i, e_i^+)} + \sum_{j=1}^{2B-1} e^{\text{sim}(x_i, e_{i,j}^-)}}.$$

Inference After training the input encoder and prompt encoder, we encode the entire set of training examples with $E_P(\cdot)$ in a pre-processing step using FAISS (Johnson et al., 2017). At test time, given an input sequence, x_{test}, we compute its encoding $E_X(x_{\text{test}})$, and then use maximum inner-product search over the training data to find the L most similar training examples, sorted by their inner product (from high to low): $\mathcal{P} = (e_1, \ldots, e_L)$. The final prompt $\mathcal{P}'$ is determined by C, the maximal context size supported by the inference LM, g. Specifically, $L' \leq L$ is the largest L' such $\sum_{i=1}^{L'} |e_i| + |x_{\text{test}}| + |y'| \leq C$, where $|y'|$ is the desired maximal length of the generated output. Finally, we return the output of greedy decoding on $g([e_{L'}; e_{L'-1}; \ldots; e_1; x_{\text{test}}])$.

Large Language Models Are Implicitly Topic Models: Explaining and Finding Good Demonstrations for In-Context Learning

Xinyi Wang [1] Wanrong Zhu [1] William Yang Wang [1]

concept variable is learned implicitly. A topic model models the text data distribution by assuming the presence of a latent variable, known as the topic variable, which governs the data generation process:

$$P(\boldsymbol{w}_{1:T}) = \int_{\Theta} P(\boldsymbol{w}_{1:T}|\boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}$$

Where $\boldsymbol{\theta} \in \Theta$ represents a potentially high dimensional topic/concept variable, Θ is the space of the topic/concept variable, and $\boldsymbol{w}_{1:T}$ refers to the token sequence of a data example, such as a sentence or document. A well-known example of a topic model is Latent Dirichlet Allocation (LDA) (Blei et al., 2003), which further assumes that each token in the example is mutually independent of one another given the topic variable. i.e. $P(\boldsymbol{w}_{1:T}|\boldsymbol{\theta}) = \prod_{i=1}^{T} P(\boldsymbol{w}_i|\boldsymbol{\theta})$.

On the other hand, LLMs model text data according to the general probabilistic decomposition:

$$P(\boldsymbol{w}_{1:T}) = \prod_{i=1}^{T} P(\boldsymbol{w}_i|\boldsymbol{w}_{i-1}, ..., \boldsymbol{w}_1)$$

In this study, we focus specifically on generative LLMs that have been pre-trained with a language model objective. While in practice, LLMs generate new tokens based on all previous tokens, we investigate whether a simplified assumption similar to that of topic models can be made for LLMs, which would take the form of:

$$P_M(\boldsymbol{w}_{t+1:T}|\boldsymbol{w}_{1:t}) = \int_{\Theta} P_M(\boldsymbol{w}_{t+1:T}|\boldsymbol{\theta})P_M(\boldsymbol{\theta}|\boldsymbol{w}_{1:t})d\boldsymbol{\theta}$$

2.1. Notations and Problem Setting

Suppose the objective of our task is to predict a discrete target variable $Y \in \mathcal{Y}$, given a token sequence $X \in \mathcal{X}$, where $\mathcal{X}$ is the space of all possible token sequences. $\boldsymbol{\theta} \in \Theta$ is the latent concept variable, where Θ is the space of the concept variable. Note that, unlike the traditional topic model, $\boldsymbol{\theta}$ is not assumed to be discrete, and Θ is not assumed to be finite. To define the data generation process, we posit the existence of an underlying causal relation between X, Y, and $\boldsymbol{\theta}$. We examine two potential directions of this causal relation, namely $X \rightarrow Y \leftarrow \boldsymbol{\theta}$ and $Y \rightarrow X \leftarrow \boldsymbol{\theta}$, which can be represented mathematically as the following structural equations:

$$Y = f(X, \boldsymbol{\theta}, \boldsymbol{\epsilon}) \qquad X = g(Y, \boldsymbol{\theta}, \boldsymbol{\epsilon})$$

In order to perform in-context learning with an LLM (generically denoted by model label M), we condition on a fixed set of k demonstration examples $(X_1^d, Y_1^d), (X_2^d, Y_2^d), ..., (X_k^d, Y_k^d)$ for a task $d \in \mathcal{T}$, where $\mathcal{T}$ is the space of all possible tasks, and $X_i^d \in \mathcal{X}$, $Y_i^d \in \mathcal{Y}$ for all $i \in [1, k]$. In this approach, we do not include a task description in the prompt, with the aim of focusing on the examination of the demonstrations.

The data generation process of a task d can be written as:

$$Y_i^d = f(X_i^d, \theta^d, \boldsymbol{\epsilon})$$

Here $\theta^d \in \Theta$ is the value of the concept variable corresponding to task d. To naturally project $\mathcal{Y}$ into the token

$$P(\boldsymbol{w}_{1:T}) = \int_{\Theta} P(\boldsymbol{w}_{1:T}|\boldsymbol{\theta}) P(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

$$P_M^d(Y|X_1^d, Y_1^d, ..., X_k^d, Y_k^d, X)$$
$$= \int_{\Theta} P_M^d(Y|\boldsymbol{\theta}, X) P_M^d(\boldsymbol{\theta}|X_1^d, Y_1^d, ..., X_k^d, Y_k^d, X) d\boldsymbol{\theta} \quad (1)$$

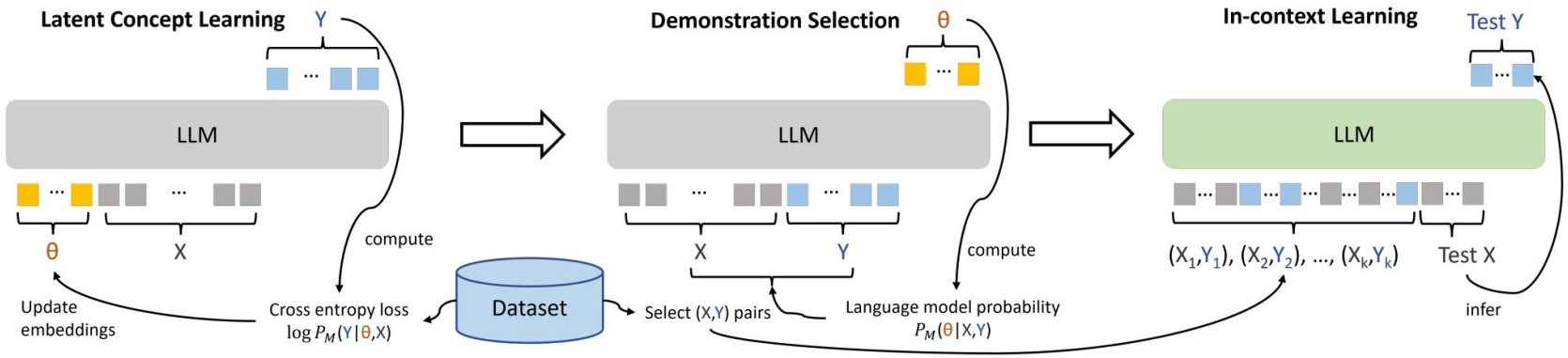


Figure 1. An overview of our proposed two-phased algorithm. Demonstration selection and latent concept learning share the same LLM as demonstration selection needs to reuse the learned concept tokens, while at the in-context learning time, any other generative LLM can be used. Note that here we only illustrate the $X \to Y \leftarrow \boldsymbol{\theta}$ direction. The $Y \to X \leftarrow \boldsymbol{\theta}$ direction can be illustrated similarly by exchanging X and Y in the above figure.

$$\mathcal{L}(\hat{\theta}^d) = \mathbb{E}_{X,Y}[\ell(X, Y; \hat{\theta}^d)], \text{ where}$$

$$\ell(X, Y; \hat{\theta}^d) = \begin{cases} -\log P_M^d(Y|\hat{\theta}^d, X) & \text{if } X \to Y \leftarrow \boldsymbol{\theta} \\ -\log P_M^d(X|\hat{\theta}^d, Y) & \text{if } Y \to X \leftarrow \boldsymbol{\theta}. \end{cases}$$

Algorithm 1 Latent concept learning

Input: dataset $\mathcal{D}$ associated with a set of tasks $\mathcal{S}$. LLM M, number of concept tokens per task c, learning rate α, and number of training steps L.
Output: LLM M' with fine-tuned concept tokens.
Add $c|\mathcal{S}|$ new tokens to the vocabulary;
Randomly initialize their embeddings E_{new};
Freeze all parameters in M except E_{new};
for $i = 1$ **to** L **do**
  Sample a random batch B in $\mathcal{D}$;
  Initialize gradient $g \leftarrow 0$;
  for each data point (x, y, d) in B **do**
    $g = g + \frac{\partial \ell(X,Y;\hat{\theta}^d)}{\partial E_{new}}$;
  end for
  $E_{new} = E_{new} - \alpha g$;
end for

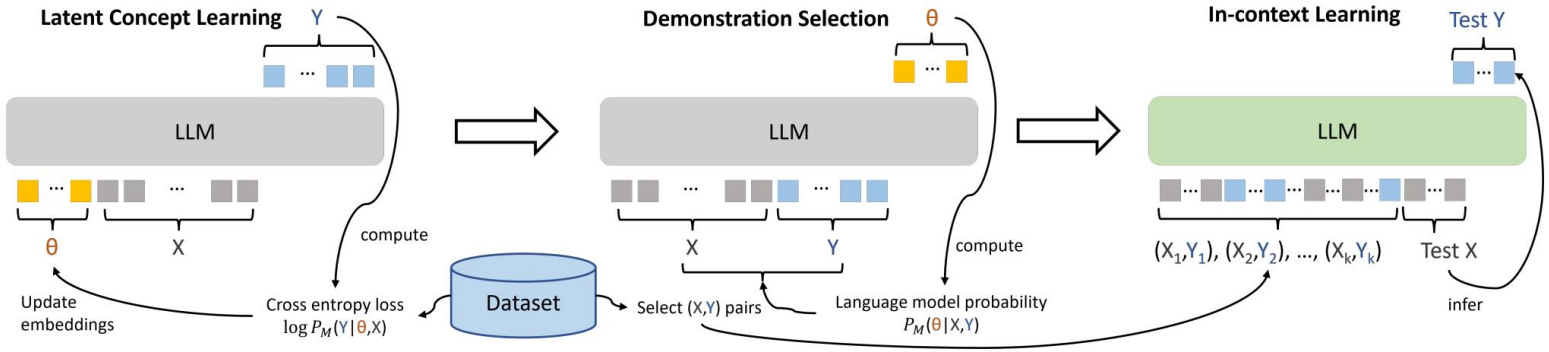


Figure 1. An overview of our proposed two-phased algorithm. Demonstration selection and latent concept learning share the same LLM as demonstration selection needs to reuse the learned concept tokens, while at the in-context learning time, any other generative LLM can be used. Note that here we only illustrate the $X \rightarrow Y \leftarrow \boldsymbol{\theta}$ direction. The $Y \rightarrow X \leftarrow \boldsymbol{\theta}$ direction can be illustrated similarly by exchanging X and Y in the above figure.

Algorithm 2 Demonstration selection

Input: dataset $\mathcal{D}^d$ for a task d. LLM with fine-tuned concept tokens M'. The number of demonstrations k.
Output: A set of ordered demonstrations.
for each (X^d, Y^d) in $\mathcal{D}^d$ **do**
  Compute $\hat{P}_M^d(\hat{\theta}^d|X^d, Y^d)$;
end for
Select top k examples with the largest $\hat{P}_M^d(\theta^d|X^d, Y^d)$, denoted as $(X_1^d, Y_1^d), ..., (X_k^d, Y_k^d)$;
for each permutation π **do**
  Compute $\hat{P}_M^d(\hat{\theta}^d|\pi((X_1^d, Y_1^d), ..., (X_k^d, Y_k^d)))$;
end for
Select the permutation π with the largest $\hat{P}_M^d(\hat{\theta}^d|\pi((X_1^d, Y_1^d), ..., (X_k^d, Y_k^d)))$.

Suppose each demonstration is also sampled independently. Then we have:

$$P_M^d(\theta^d|X_1^d, Y_1^d, ..., X_k^d, Y_k^d) = \frac{\prod_{i=1}^k P_M^d(\theta^d|X_i^d, Y_i^d)}{P_M^d(\theta^d)^{k-1}}$$

We assume that $\boldsymbol{\theta}$ has a uniform prior. Then our goal becomes finding the top k demonstrations that maximize $\hat{P}_{M'}^d(\hat{\theta}^d|X_i^d, Y_i^d)$.

Also, as we are using an LLM to approximate the data distribution, the order of the demonstrations matters. We then choose the order according to the posterior of the concept tokens:

$$\arg\max_{\pi \in \Pi} \hat{P}_M^d(\theta^d|\pi((X_1^d, Y_1^d), ..., (X_k^d, Y_k^d))) \quad (3)$$

Where $\pi((X_1^d, Y_1^d), ..., (X_k^d, Y_k^d))$ is a permutation of $(X_1^d, Y_1^d), ..., (X_k^d, Y_k^d)$. Π is the set of all possible permutations of the k demonstrations. We summarize the proposed algorithm in Algorithm 2.

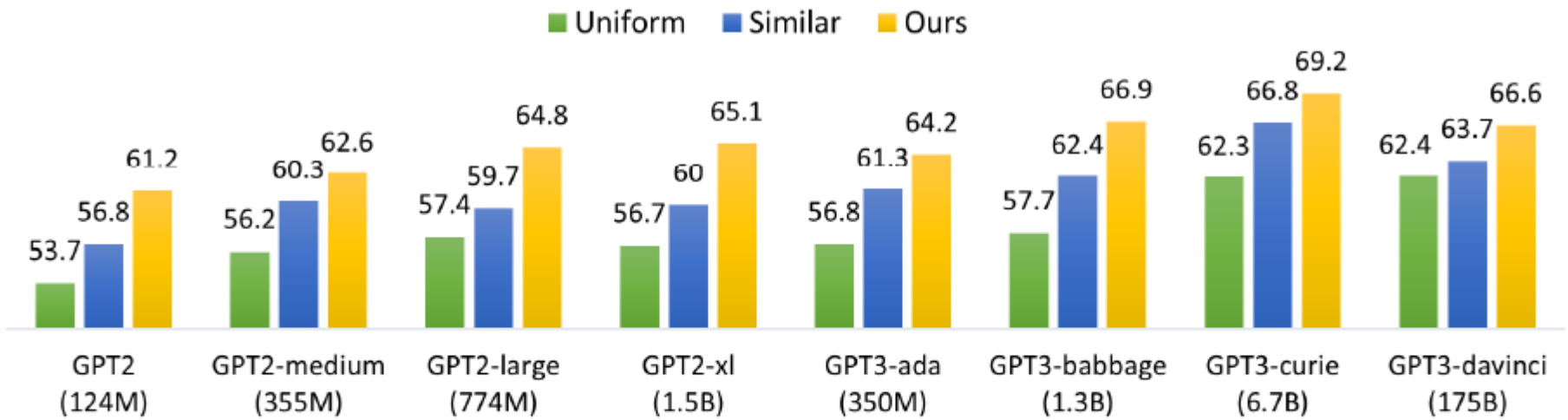


Figure 2. Accuracy of 4-shot in-context learning using demonstrations selected by our method and other baselines, averaged over eight datasets. Our demonstrations are selected using GPT2-large, and the same set of demonstrations is then applied to all other LLMs.

Figure 7. In-context learning accuracy of our method versus random selection baseline, with different k, averaged over all eight datasets. Numbers are obtained with GPT3-ada.

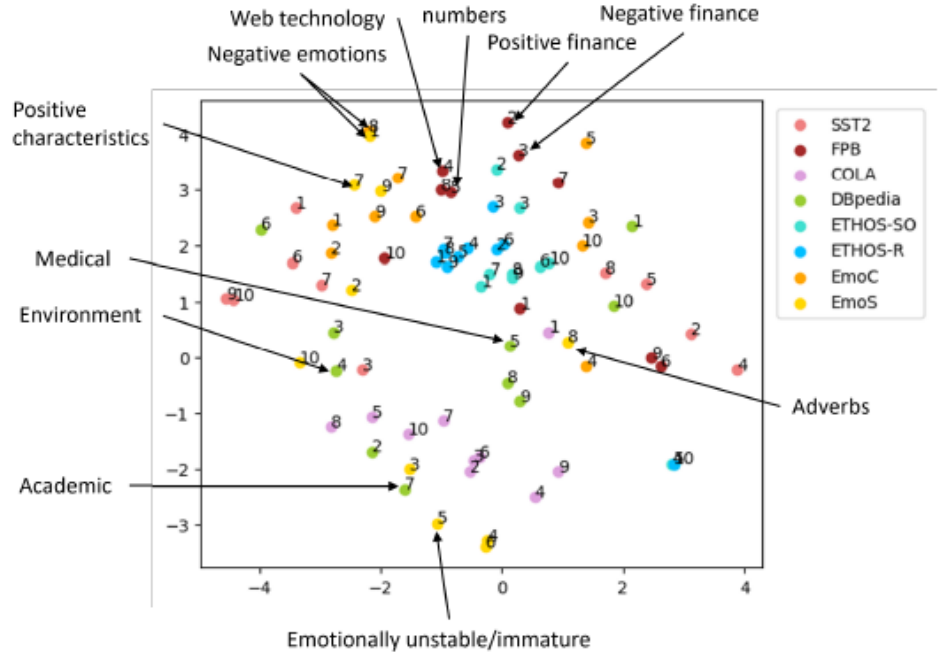


Figure 9. t-SNE plot of the learned concept tokens for each task. Concept tokens that can be explained by similar tokens are anno-

Table 10. We list the top 10 similar words (tokens) to some of the learned concept tokens.

concept token	similar words
FPB-2	milo coordinate notify rendering benefiting routing EntityItem routed Messages Plot
FPB-3	unlocked updating deleting dropping damage updates drops Gained taken dropped
FPB-4	FX Safari Fixes advertisers Links Coins Operator marketers Guidelines
FPB-5	674 592 693 696 498 593 793 504 691 683
COLA-1	exha trunc curv fragmented elong iterator initialized bounds Iter filament
COLA-2	Sp spa contributed cerv borrower paper tiger Erica USH Schwartz
COLA-7	democr Barack WH ophobic neum Democrats Rachel WH Democrats
DBpedia-4	often impede blockade incarcerated LEASE pollutants pesticides uphe lawmakers fossils
DBpedia-5	categorized closes therapies antidepressant retrospective clinically physicians therapists randomized clinicians
DBpedia-7	JS provided Killed richness Compet Nevertheless Probably Proceedings horizontally
ETHOS-SO-3	Revolution Spread itu Million Pascal stabil Indy Georgian Figure resy
ETHOS-R-2	council Chocobo Shant uyomi aditional cumbers subur ThumbnailImage araoh Pharaoh
ETHOS-R-8	seems outlines emitted grin outline circuitry sized flips emits flipped
ETHOS-R-9	223 asel Cyrus Sith Scorpion Snape Jas Leia Ned Morty
EmoC-6	behavi checkpoints unintention crib eleph looph np mosquit blat pione
EmoC-8	depressed bullied choked stricken devastated unsuccessful cheated distraught troubled failing
EmoS-1	frightened rebellious depressed careless bullied restless reluctant distraught clumsy disgruntled
EmoS-5	obsessive crappy demonic delusions psychosis psychotic childish stupidity reckless insanity
EmoS-7	benevolent charismatic perfected volunte unintention pione innocuous fearless glamorous ruthless
EmoS-9	whispers pundits Sadly horribly curiously noticeably Sadly gaping painfully shockingly

2023 ICLR

SELECTIVE ANNOTATION MAKES LANGUAGE MODELS BETTER FEW-SHOT LEARNERS

Hongjin Su[♠] **Jungo Kasai**[♣◇] **Chen Henry Wu**[♡] **Weijia Shi**[♣] **Tianlu Wang**[♦] **Jiayi Xin**[♠]
Rui Zhang[★] **Mari Ostendorf**[♣] **Luke Zettlemoyer**[♣♦] **Noah A. Smith**[♣◇] **Tao Yu**[♠♣]
[♠]The University of Hong Kong [♣]University of Washington [◇]Allen Institute for AI
[♡]Carnegie Mellon University [★]Penn State University [♦]Meta AI

{hjsu,tyu}@cs.hku.hk, henrychenwu@cmu.edu, ostendor@uw.edu
{jkasai,swj0419,lsz,nasmith}@cs.washington.edu

In particular, they show that the performance substantially improves when similar examples (under some embedding function) are retrieved as in-context examples specifically for each test input (Liu et al., 2022). Each test sample only requires a few in-context examples in its prompt. Different test instances, however, require different in-context examples with their associated annotations, necessitating a large set of annotated examples.

Finding Supporting Examples for In-Context Learning

Xiaonan Li, Xipeng Qiu
School of Computer Science, Fudan University
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
{lixn20, xpqiu}@fudan.edu.cn

. Different from these methods which aim to provide example-specific and relevant information for the test input, our task-level setting seeks to find examples which are representative for the task.

Published as a conference paper at ICLR 2023

TEMPERA: TEST-TIME PROMPT EDITING VIA REINFORCEMENT LEARNING

Tianjun Zhang[1] **Xuezhi Wang**[2] **Denny Zhou**[2] **Dale Schuurmans**[2, 3] **Joseph E. Gonzalez**[1]
[1] UC Berkeley [2] Google Research, Brain Team [3] University of Alberta
tianjunz@berkeley.edu

2022 EMNLP

Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?

Sewon Min[1,2] **Xinxi Lyu**[1] **Ari Holtzman**[1] **Mikel Artetxe**[2]
Mike Lewis[2] **Hannaneh Hajishirzi**[1,3] **Luke Zettlemoyer**[1,2]
[1]University of Washington [2]Meta AI [3]Allen Institute for AI
{sewon,alrope,ahai,hannaneh,lsz}@cs.washington.edu
{artetxe,mikelewis}@meta.com

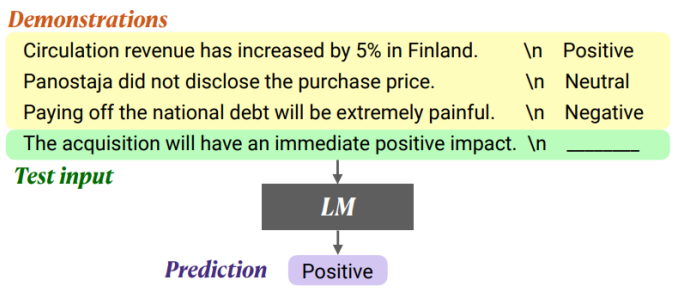


Figure 2: An overview of in-context learning. The demonstrations consist of k input-label pairs from the training data ($k = 3$ in the figure).

Figure 1: Results in classification (top) and multi-choice tasks (bottom), using three LMs with varying size. Reported on six datasets on which GPT-3 is evaluated; the channel method is used. See Section 4 for the full results. In-context learning performance drops only marginally when labels in the demonstrations are replaced by random labels.

Z-ICL: Zero-Shot In-Context Learning with Pseudo-Demonstrations

Xinxi Lyu[1] **Sewon Min**[1] **Iz Beltagy**[2]
Luke Zettlemoyer[1] **Hannaneh Hajishirzi**[1,2]
[1]University of Washington [2]Allen Institute for AI
{alrope,sewon,lsz,hannaneh}@cs.washington.edu
beltagy@allenai.org

Decoupling Knowledge from Memorization: Retrieval-augmented Prompt Learning

Xiang Chen[1,2]*, **Lei Li**[1,2]*, **Ningyu Zhang**[1,2]†, **Xiaozhuan Liang**[1,2], **Shumin Deng**[1,2], **Chuanqi Tan**[3], **Fei Huang**[3], **Luo Si**[3], **Huajun Chen**[1,2]†

[1]Zhejiang University & AZFT Joint Lab for Knowledge Engine, China

[2]Hangzhou Innovation Center, Zhejiang University, China

[3]Alibaba Group, China

{xiang_chen, leili21, zhangningyu, liangxiaozhuan, 231sm, huajunsir}@zju.edu.cn,

{chuanqi.tcq, f.huang, luo.si}@alibaba-inc.com