Decoupling Knowledge from Memorization: Retrieval-augmented Prompt Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Prompt learning approaches have made waves in natural language processing by inducing better few-shot performance while they still follow a parametric-based learning paradigm; the oblivion and rote memorization problems in learning may encounter unstable generalization issues. Specifically, vanilla prompt learning may struggle to utilize atypical instances by rote during fully-supervised training or overfit shallow patterns with low-shot data. To alleviate such limitations, we develop RETROPROMPT with the motivation of decoupling knowledge from memorization to help the model strike a balance between generalization and memorization. In contrast with vanilla prompt learning, RETROPROMPT constructs an open-book knowledge-store from training instances and implements a retrieval mechanism during the process of input, training and inference, thus equipping the model with the ability to retrieve related contexts from the training corpus as cues for enhancement. Extensive experiments demonstrate that RETROPROMPT can obtain better performance in both few-shot and zero-shot settings. Besides, we further illustrate that our proposed RETROPROMPT can yield better generalization abilities with new datasets. Detailed analysis of memorization indeed reveals RETROPROMPT can reduce the reliance of language models on memorization; thus, improving generalization for downstream tasks[1].

1 Introduction

Large parametric language models [42, 6, 19, 28] have achieved dramatic empirical success in natural language processing (NLP). Notably, pre-trained language models (PLMs) have learned a substantial amount of in-depth knowledge from data, and have archived tremendous promise in few-shot/zero-shot learning ability with the natural language prompts [11, 47, 52]. However, Recent studies [34, 36, 54] observe that prompt learning with PLMs usually generalizes unstably in an extremely low-resource setting or emerging domains. One potential reason is that, it is non-trivial for parametric models to *learn rare or hard patterns well with rote memorization*, thus, resulting in inefficient generalizable performance.

Intuitively, if we regard the whole training data as a *book* and the test phase as the *examination*, the current training-test procedure of prompt learning (based on batch data training) can be viewed as *page-by-page memorization* and *closed-book examination* [39]. During training, vanilla prompt learning may struggle to memorize atypical instances in a fully-supervised setting or overfit shallow patterns with low-shot data [56, 8]. Specifically, recent studies[9, 10] have proposed a long-tail theory,

[1]Code and datasets are in the supplementary materials and will be released for reproducibility.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

which states that if training data form a long-tail distribution and have small "sub-populations" with atypical instances, then PLMs indeed predict on the test data through rote memorizing these atypical instances rather than learning the common patterns [56, 51].

The limitations of rote memorization remind us of the human learning process of *"learn by analogy"* and the proverb that *"the palest ink is better than the best memory"*. Note that humans can perform associative learning to recall relevant skills in deep memories for reinforcing each other, thus, owning the extraordinary abilities to solve few-shot and zero-shot tasks. Motivated by these, we endeavor to improve the generalization ability of prompt learning with retrieval and association. Our intuition is that the difficulty of resolving the above limitations can be substantially alleviated if we can decouple the knowledge from memorization by constructing *an open-book knowledge-store* from the training data; thus, referring to related knowledge could provide a strong enhancement signal to help the model strike a balance between generalization and memorization.

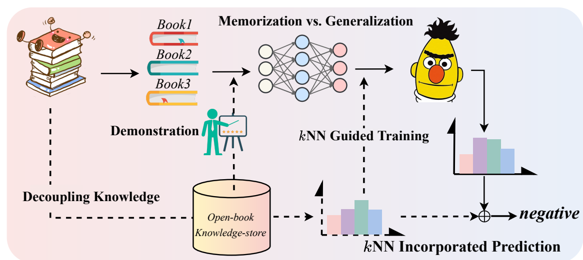


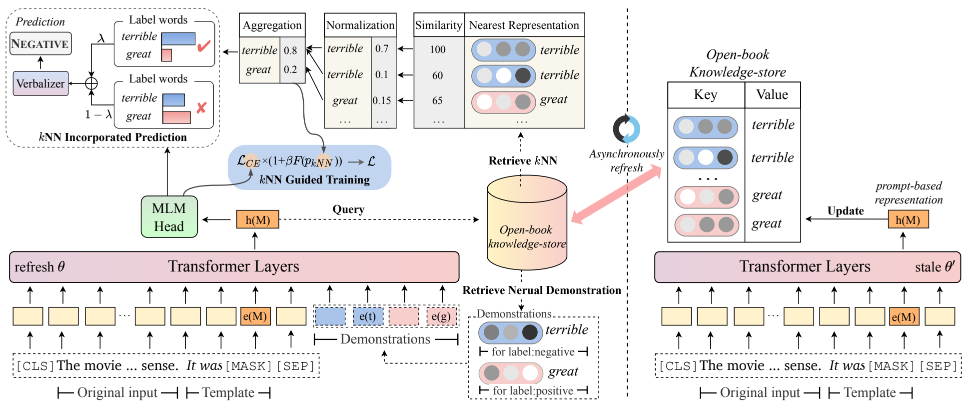
Figure 1: Decoupling knowledge from memorization.

Specifically, we introduce a novel retrieval-augmented framework based on prompt learning (**RetroPrompt**) as shown in Figure 1. The open-book knowledge store $(\mathcal{K}, \mathcal{V})$, defined as the set of *key: prompt-based example embeddings* and *value: corresponding label words* constructed from the training data, are served as additional references for the model to decouple knowledge from pure memorization to some extent. Specifically, to integrate retrieved knowledge into the input, **Firstly**, we design to incorporate neural demonstrations into the input sequences as in-context augmentation, where the demonstration is retrieved from the knowledge-store. **Then**, we apply a non-parametric algorithm kNN over the input query and knowledge store, and regard kNN results as an indication of easy vs. hard examples in the training set. More specifically, we automatically force the model to focus on the hard examples identified by kNN by assigning a scaling during training. **Lastly**, the kNN results are further employed at the output of the PLM head to participate in masked prediction during inference. The model retrieves Top-k nearest reference instances as cues from $(\mathcal{K}, \mathcal{V})$ and makes inference by linearly interpolating the output of prompt learning with a non-parametric nearest neighbor distribution.

The considerable performance gains on nine tasks in few-shot and zero-shot settings demonstrate that our systemic retrieval mechanism helps the model generalize better with scarce data. Experiments in the fully-supervised setting with long-tail distribution illustrate that our RetroPrompt can deal with atypical instances more robustly. We further adopt self-influence [24] as our memorization scoring function to analyze the memorization process between fine-tuning, prompt learning and our RetroPrompt. The final analysis results show that 1) the training instances with the highest memorization scores tend to be atypical, 2) RetroPrompt generalize better than fine-tuning and convention prompt-tuning with decoupling knowledge from memorization to alleviate the rote of PLMs. In a nutshell, our work may open up new avenues to improve the generalization of prompting PLMs by retrieving knowledge from memorization.

2 Preliminaries of Prompt Learning

Assuming that $\mathcal{M}$, $\mathcal{T}$ respectively denotes the PLM and the template function for prompt tuning. Formally, the text classification task takes a query sentence $\boldsymbol{x} = (x_0, x_1, ..., x_n)$ as input, and classify it into a class label $y \in \mathcal{Y}$. While prompt learning converts classification task into a masked language modeling problem with *cloze-style* objectives. Specifically, the template function $\mathcal{T}$ inserts pieces of texts into $\boldsymbol{x}$ as $\hat{\boldsymbol{x}} = \mathcal{T}(\boldsymbol{x})$, where $\hat{\boldsymbol{x}}$ is the corresponding input of $\mathcal{M}$ with a `[MASK]` token in it. For example, assuming we need to classify the sentence $\boldsymbol{x}$ ="The movie makes absolutely no sense." into



a. Retrieval-augmented prompt learning

b. Creation and refresh of open-book knowledge-store

Figure 2: Overview of RETROPROMPT. Note that $e(\cdot)$ denotes word embedding function in the PLM $\mathcal{M}$, while "M","t" and "g" in $e(\cdot)$ specifically refers to "[MASK]", "terrible" and "great".

label NEGATIVE (labeled as 0) or POSITIVE (labeled as 1), we wrap it into

$$\hat{\boldsymbol{x}} = \texttt{[CLS]}\,\boldsymbol{x}\text{ It was }\texttt{[MASK]}\,\texttt{[SEP]} \tag{1}$$

The verbalizer $f : \mathcal{Y} \mapsto \mathcal{V}$ is defined as a mapping from the label space $\mathcal{Y}$ to a few words in the vocabulary, which form the *label word* set $\mathcal{V}$. The base component of $\mathcal{M}$ produces the sequence representation over $\hat{\boldsymbol{x}}$, and we choose the hidden vector at the `[MASK]` position as the contextual representation $\boldsymbol{h}_{\hat{\boldsymbol{x}}} \in \mathbb{R}^d$, where d is the dimension of hidden states. Then the MLM head of $\mathcal{M}$ can operate on $\boldsymbol{h}_{\hat{\boldsymbol{x}}}$ to calculate the probability of each word v in the vocabulary being filled in `[MASK]` token $P_{\mathcal{M}}(\texttt{[MASK]} = v|\hat{\boldsymbol{x}})$. We let $\mathcal{V}_y$ to represent the subset of $\mathcal{V}$ that is connected with a specific label y, $\cup_{y\in\mathcal{Y}}\mathcal{V}_y = \mathcal{V}$. Then the probability distribution over the label y is calculated as:

$$P(y|\boldsymbol{x}){=}g\left(P_{\mathcal{M}}(\texttt{[MASK]}{=}v|\mathcal{T}(\boldsymbol{x}))|v \in \mathcal{V}_y\right), \tag{2}$$

where g is a function transforming the probability of label words into the probability of the classes.

3 RETROPROMPT: Retrieval-augmented Prompt Learning

We introduce a simple and general retrieval-augmented framework for prompt learning, named RETROPROMPT, whose basis is the dense retriever (§3.1) with an open-book knowledge-store to decouple knowledge from memorization. As shown in Figure 2, RETROPROMPT consists of three components: retrieval of neural demonstration for enhancing input (§3.2), the kNN guided training (§3.3) and the kNN-based probability for *cloze-style* prediction (§3.4).

3.1 Dense Retriever

Open-book Knowledge-store The first step of our proposed framework is to build a knowledge-store for retrieval that can decouple from memorization and captures the semantics of the instance from the training set $\mathcal{C}$. Specifically, we utilize the encoder to embed prompt-based instance representation over the $\mathcal{C}$ to construct the knowledge-store. Given the i-th example $(\boldsymbol{c}_i, y_i)$ in the training data $\mathcal{C}$, we compute the key-value pair $(\boldsymbol{h}_{\hat{\boldsymbol{c}}_i}, v_i)$, in which $\hat{\boldsymbol{c}}_i = \mathcal{T}(\boldsymbol{c}_i)$, $\boldsymbol{h}_{\hat{\boldsymbol{c}}_i} \in \mathbb{R}^d$ is the embedding of the `[MASK]` token in the last layer of the underlying PLM, and $v_i = f(y_i)$ denotes the label word of the i-th example. We store all pairs $(\boldsymbol{h}_{\hat{\boldsymbol{c}}}, v)$ in a key-value datastore $(\mathcal{K}, \mathcal{V})$ where $\boldsymbol{h}_{\hat{\boldsymbol{c}}}$ serves as *key* and v as *value* as follows:

$$(\mathcal{K}, \mathcal{V}) = \{(\boldsymbol{h}_{\hat{\boldsymbol{c}}_i}, v_i) \mid (\boldsymbol{c}_i, y_i) \in \mathcal{C}\} \tag{3}$$

The knowledge-store is flexible to add, edit or delete any instances and can be asynchronously updated during the training procedure. Note that our knowledge-store is constructed from few-shot trainsets in the corresponding few-shot settings rather than the whole available training data.

Efficient Searching Considering that the size of the training data $\mathcal{C}$ can be enormous, we must ensure an efficient retrieval process. As shown in the above creation of open-book knowledge-store, we can build the matrix $\mathbf{D} \in \mathbb{R}^{|\mathcal{C}| \times d}$ as the index of training examples. Given a query set Q, we first encode each query example with template mapping function $\mathcal{T}(\cdot)$ to get a set of prompt-based query vectors $\boldsymbol{h}_{\hat{q}}$ for retrieval augmentation on the fly. Then, we utilize query vectors to search for the closest examples over the index $\mathbf{D}$ via maximum inner product search (MIPS). For the retrieval process, we choose FAISS [18] to query the open-book knowledge-store efficiently. FAISS is an excellent open-source library for fast nearest neighbor retrieval in high-dimensional spaces.

Asynchronous Refresh of the Knowledge-store Since the neural demonstration may lead to the variable contextual representation of instance as the parameters of the PLM are continually updated, we thus propose to "refresh" the index of retrieval by asynchronously re-embedding and re-indexing all embeddings in an open-book knowledge-store every j training epochs [2]. In § 4.6, we empirically demonstrate that this procedure results in performance improvement.

3.2 Retrieval of Neural Demonstration

To enhance the PLMs with the ability to learn by analogy through the knowledge-store, we further combine RetroPrompt with neural demonstrations, an orthogonal technique enhancing language models, to improve the generalization ability of our model. For the t-th query instance $\boldsymbol{q}_t$, we first utilize prompt-based representation $\boldsymbol{h}_{\hat{q}_t}$ to query the cached representations of open-book knowledge-store. Then we retrieve m nearest neighbors $\{\{\boldsymbol{c}_1^{(1)}, ..., \boldsymbol{c}_m^{(1)}\}, ..., \{\boldsymbol{c}_1^{(L)}, ..., \boldsymbol{c}_m^{(L)}\}\}$ of $\boldsymbol{q}_t$ for each class, where the superscript L denotes the total number of the classes and the $\boldsymbol{c}_i^{(l)}$ is retrieved as the i-th nearest neighbor in the l-th class. After the model retrieves the Top-m candidates for each class, their corresponding representation $\boldsymbol{h}_{\hat{\boldsymbol{c}}_i}^{(l)}$ and label word $v^{(l)}$ from knowledge-store will be incorporated into the encoder to act as a demonstration learning. Since the $\boldsymbol{h}_{\hat{\boldsymbol{c}}_i}^{(l)}$ is already vector, we intuitively aggregate the m neighbor vectors for each class according to their similarity and incorporate the demonstration into the input representation of $\hat{\boldsymbol{x}}$ after the word embedding layer of the $\mathcal{M}$ as follows:

$$\mathcal{I} = e(\hat{\boldsymbol{x}}) \oplus [\sum_{i \in [1:m]} \alpha_i^{(1)} \boldsymbol{h}_{\hat{\boldsymbol{c}}_i}^{(1)}, e(v^{(1)})] \oplus ... \oplus [\sum_{i \in [1:m]} \alpha_i^{(L)} \boldsymbol{h}_{\hat{\boldsymbol{c}}_i}^{(L)}, e(v^{(L)})]; \alpha_i^{(l)} = \frac{e^{\boldsymbol{h}_{\hat{\boldsymbol{q}}} \cdot \boldsymbol{h}_{\hat{\boldsymbol{c}}_i}^{(l)}}}{\sum_{i \in [1:m]} e^{\boldsymbol{h}_{\hat{\boldsymbol{q}}} \cdot \boldsymbol{h}_{\hat{\boldsymbol{c}}_i}^{(l)}}} \tag{4}$$

where $e(\cdot)$ represents the word embedding layer of $\mathcal{M}$, $\oplus$ denotes the concatenation of input sequences, $\alpha_i^{(l)}$ is the softmax score for the i-th retrieval belonging to l-th class label to denote their relevance with $\hat{\boldsymbol{q}}$, and $\mathcal{I}$ is the sequence features for inputting the next layer of PLM. As shown in the above equation, we encode demonstration representation with the weighted sum of the retrieval representation. Thus, retrieval scores are directly used in the final representation, making the framework differentiable. To this end, we denote this style of demonstration as *neural demonstration*, significantly different from prior work of *discrete demonstration* [11].

Neural vs. Discrete Demonstration Compared with prior discrete demonstrations described in [11, 32, 46, 25], retrieving weighted neural demonstrations from the knowledge-store to augment prompt learning has advantages in the following three major aspects: (1) neural demonstrations could be more tolerant of the model's maximum input length than discrete demonstrations, while the discrete demonstration is usually not suitable for multi-class classification tasks due to the limitation of input length, such as relation extraction, etc. (2) the model needs to deal with large retrieval tokens for discrete demonstration, making it time-consuming and computationally intensive to perform cross-attention operations due to the quadratic attention complexity. In contrast, dealing with much shorter instance representations as neural demonstrations unleashes the potential of cross-attention and accelerates the inference. (3) when sampling examples based on the similarity between instances, our *cloze-style* contextual representation is more informative and consistent than the contextual representation from `[CLS]` of Sentence-BERT [44] (adopted in LM-BFF).

[2] Specifically, we refresh the knowledge-store for each epoch in our experiments.

3.3 Retrieve kNN for Guiding Training

Eager learners, such as PLMs, are trained to provide a global approximating function that maps from input to output space. Lazy learners such as k-nearest neighbor classifiers, on the contrary, focus on approximating the neighborhoods around test examples [2]. Since kNN can easily predict for each encountered query instance based on pre-trained representation without an extra classifier, it is intuitively to leverage the kNN's classification results as the **prior external knowledge** to guide the PLMs' parameters attending to hard examples (hard samples usually refer to atypical samples) during the training process (also referred as kNN-train for the abbreviation). Particularly, our intuition is to differentiate between easy and hard examples according to the prediction of kNN. Given the t-th query instance $\boldsymbol{q}_t$, we leverage the $\boldsymbol{h}_{q_t}$ querying the open-book knowledge-store $(\mathcal{K}, \mathcal{V})$ to retrieve the k-nearest neighbors $\mathcal{N}$ of $\boldsymbol{q}_t$ according to a similarity function $d(\cdot,\cdot)$, where $d(\cdot,\cdot)$ typically adopt the inner product similarity. Then, we compute a distribution over neighbors based on a softmax of their similarities and aggregate probability mass for each label word across all its occurrences in the retrieved targets:

$$P_{k\mathrm{NN}}(y \mid \boldsymbol{q}_t) \propto \sum_{(\boldsymbol{c}_i, y_i) \in \mathcal{N}} \mathbb{1}_{y=y_i} \exp\left(d\left(\boldsymbol{h}_{\hat{\boldsymbol{q}}_t}, \boldsymbol{h}_{\hat{\boldsymbol{c}}_i}\right)\right). \tag{5}$$

Given the probability $p_{k\mathrm{NN}}$ of the query instance $\boldsymbol{q}_t$ being predicted as the **gold class**, we propose to retrieve the kNN for guiding the training process of prompt learning. The kNN guider reweights the cross-entropy loss $\mathcal{L}_{CE}$ by adjusting the relative loss for the correctly-classified or misclassified instances identified by kNN, respectively. Specifically, we apply the negative log-likelihood as the modulating factor $F(p_{k\mathrm{NN}})$. The final loss $\mathcal{L}$ is defined as:

$$F(p_{k\mathrm{NN}}) = -\log(p_{k\mathrm{NN}}), \quad \mathcal{L} = (1 + \beta F(p_{k\mathrm{NN}}))\, \mathcal{L}_{CE}, \tag{6}$$

where β denotes a scalar to determine the proportion of each loss term. Note that $p_{k\mathrm{NN}}$ is computed using the *leave-one-out* distribution on the training set due to the fact that each example in the training set cannot retrieve itself. The motivation of modulating factor here is similar to Focal-loss [31], while we focus on exploit the application of kNN in tuning PLMs.

3.4 kNN based probability for *Cloze-style* Prediction

Apart from the neural demonstration on the input side and kNN guided training process (also referred as kNN-test for the abbreviation), we further present kNN based probability for *Cloze-style* prediction on the inference process, providing the PLM ability to retrieve nearest neighbors for decisions rather than making predictions only based on memorized parameters. Given the non-parametric k nearest neighbor distribution $P_{k\mathrm{NN}}$ of the query instance $\boldsymbol{q}_t$ being predicted as y, the $P(y \mid \boldsymbol{q}_t)$ is reformulated by interpolating the $P_{k\mathrm{NN}}$ with the already-trained base PLM's MLM prediction $P_{\mathcal{M}}$ using parameter λ to produce the final probability of the label:

$$P(y \mid \boldsymbol{q}_t) = \lambda P_{k\mathrm{NN}}(y \mid \boldsymbol{q}_t) + (1-\lambda) g\left(P_{\mathcal{M}}(\texttt{[MASK]} = v \mid \mathcal{T}(\boldsymbol{q}_t))\right). \tag{7}$$

Different from kNN-LM [14] that uses tokens to augment the language modeling directly, we explicitly take advantage of prompt-based instance representation for classification tasks, which is more deeply rooted in prompt learning. In this way, we can unlock the model prediction process as an *open-book* examination.

4 Experiments

4.1 Datasets and Baselines

Datasets We evaluate RetroPrompt on several types of natural language understanding tasks, including single sentence classification tasks (SST-2 [50], MR [40], and CR [16]) and sentence pair classification tasks (MNLI [53], QNLI [43], and QQP[3]). To further evaluate the effectiveness of the proposed approach with multi-class classification, we also conduct experiments on the information extraction tasks, including FewNERD [7], SemEval 2010 Task 8 (SemEval) [15], and TACRED [55].

[3] `https://www.quora.com/q/quoradata/`.

Table 1: Results across 9 NLU datasets in the few-shot and zero-shot setting. We report mean (and standard deviation) results over five different few-shot splits. “D-demo” refers to discrete demonstration, and “KnPr” is the abbreviation of KnowPrompt. LOTClass [38] is the SOTA model in unsupervised text classification with self-training. † donates the model uses **extra knowledge** and ♣ means they **train** the PLM on the whole unlabeled trainset, while we and the other baselines only leverage the vanilla PLM to test without training. The average scores with * denote that we reuse the results of the “non-demo” version of the related model to fill in the default values.

St.	Model	Single Sentence			Sentence Pair			Model	Information Extraction			Avg.
		SST-2 (acc)	MR (acc)	CR (acc)	MNLI (acc)	QNLI (acc)	QQP (F1)		FewN (acc)	SemEval (acc)	TACRED (F1)	
16	FT	81.4 (3.8)	76.9 (5.9)	75.8 (3.2)	45.8 (6.4)	60.2 (6.5)	60.7 (4.3)	FT	52.7 (2.2)	66.1 (1.2)	25.8 (2.8)	60.6
16	LM-BFF (man)	91.6 (1.2)	87.0 (2.0)	90.3 (1.6)	64.3 (2.5)	64.6 (5.4)	65.4 (5.3)	KnPr	65.3 (1.1)	80.9 (2.5)	33.2 (2.0)	71.4
16	LM-BFF (D-demo)	91.8 (1.2)	86.6 (1.8)	90.2 (1.4)	64.8 (2.3)	69.2 (5.4)	68.2 (3.2)	KnPr (D-demo)	—	—	—	72.2*
16	KPT †	90.3 (1.6)	86.8 (1.8)	88.8 (3.7)	61.4 (2.1)	61.5 (2.8)	71.6 (2.7)	KPT †	65.9 (1.5)	78.8 (2.1)	32.8 (1.7)	70.9
16	**Ours**	**93.9** (0.4)	**88.0** (0.8)	**91.9** (0.7)	**71.1** (1.8)	**71.6** (1.8)	**74.0** (2.0)	**Ours**	**67.3** (0.9)	**81.5** (1.3)	**40.7** (0.7)	**75.6**
4	FT	60.2 (2.8)	57.6 (1.4)	66.4 (5.5)	35.0 (0.3)	54.2 (3.9)	52.8 (4.7)	FT	32.7 (2.9)	38.8 (2.0)	14.7 (2.8)	45.8
4	LM-BFF (man)	90.7 (0.8)	85.2 (2.8)	89.9 (1.8)	51.0 (2.5)	61.1 (6.1)	48.0 (4.9)	KnPr	52.5 (1.5)	58.4 (3.7)	28.8 (2.5)	62.8
4	LM-BFF (D-demo)	90.2 (1.5)	85.5 (2.1)	89.7 (0.6)	56.1 (1.0)	61.7 (7.6)	63.2 (5.6)	KnPr (D-demo)	—	—	—	65.1*
4	KPT †	88.2 (5.7)	83.4 (1.5)	87.2 (2.5)	53.7 (2.7)	59.2 (2.8)	54.9 (7.9)	KPT †	58.8 (2.2)	57.2 (3.2)	27.5 (2.2)	63.3
4	**Ours**	**91.5** (0.4)	**87.4** (0.5)	**91.4** (0.6)	**57.6** (5.5)	**62.8** (4.5)	**66.1** (4.1)	**Ours**	**60.9** (1.9)	**59.9** (1.9)	**32.1** (2.0)	**67.7**
0	LOTClass♣	71.8	81.7	50.1	50.4	36.5	55.9	LOTClass♣	11.5	9.8	2.5	41.1
0	FT	49.1	50.0	49.8	34.4	49.5	31.6	FT	10.0	6.2	0.5	31.2
0	LM-BFF (man)	83.5	80.3	78.4	49.7	50.5	49.7	KnPr	15.9	10.3	2.3	46.7
0	LM-BFF (D-demo)	82.9	80.7	**81.4**	52.2	53.5	44.0	KnPr (D-demo)	—	—	—	47.0*
0	KPT †	78.4	81.9	71.4	37.1	58.4	47.5	KPT †	24.6	11.6	0.8	45.7
0	**Ours**	**89.1**	**86.1**	79.7	**53.7**	**60.1**	**65.1**	**Ours**	**41.3**	**12.2**	**3.6**	**54.5**

Baselines We compare with LM-BFF [11] for single sentence and sentence pair classification tasks and adopt SOTA prompt learning model KnowPrompt [5] as the baseline for information extraction tasks. Note that the discrete demonstration method cannot be applied to multi-class classification tasks due to the input length limitations; thus, we leave out the experimental table about the results of KnPr (D-demo). We also compare our RETROPROMPT with the knowledge-enhanced prompt learning method KPT [17] since KPT leverages the external knowledge base for enhancing prompt learning while we focus on utilizing internal trainsets as a knowledge-store.

4.2 Evaluation protocols and details

The experiments are implemented on 1 NVIDIA V100 and utilize Pytorch [41] as the base library. We adopt $\text{RoBERTa}_{\text{large}}$ [35] as the PLM and employ AdamW as the optimizer for all experiments. To mitigate the influence of diverse templates, we conduct baselines and RETROPROMPT with the same templates for each dataset. The specific templates we use for each dataset are in Appendix. As for few-shot and zero-shot experiments, we leverage different settings, respectively.

Few-shot Setting. We follow the few-shot setting of LM-BFF [11] to conduct 4-shot and 16-shot experiments and evaluate the average performance with a fixed set of seeds, $\mathcal{S}_{\text{seed}}$, across five different sampled $\mathcal{D}_{\text{train}}$ for each task. Note that our knowledge-store is constructed with the **few-shot training set** in this setting.

Zero-shot Setting. We leverage vanilla $\text{RoBERTa}_{\text{large}}$ for all baselines (except LOTClass [38]) to directly inference on the test set. To take advantage of retrieval mechanism, RETROPROMPT follows LOTClass [38] to utilize **unlabeled** trainsets for retrieval. Specifically, we take the vanilla $\text{RoBERTa}_{\text{large}}$ to tag the pseudo labels on unlabeled trainset and create the open-book knowledge-store with the unlabeled trainsets and pseudo labels. Lastly, RETROPROMPT make predictions on the test set based on the constructed datastore **without tuning any of the model parameters**.

4.3 Experimental Results

Few-shot Results. As shown in Table 1, we find RETROPROMPT consistently outperforms baseline method LM-BFF and KnowPrompt, both in 4-shot and 16-shot experiments. Especially for information extraction tasks with multiple classes, discrete demonstrations cannot be applied to the input due to the limited input sequence length, while our neural demonstration can also work and achieves improvement on these multi-class datasets. Moreover, RETROPROMPT obtain better performance compared with KPT. Compared with KPT with external knowledge, we only focus on referencing the internal few-shot trainsets without visiting the external knowledge base. Besides, we observe that RETROPROMPT has a relatively lower standard deviation than the baselines. The reason may lie that the retrieval mechanism can compensate for instabilities in parametric predictions.

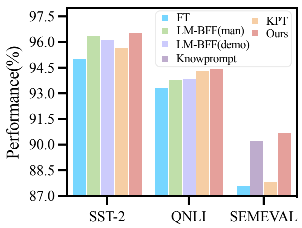


Figure 3: Performance on fully-supervised datasets.

Zero-shot Results. From Table 1, we also observe that RETROPROMPT achieves improvements in the zero-shot setting. Another notable point is that RETROPROMPT performs even better than KPT in the zero-shot setting, revealing that exploring own data to decouple knowledge from memorization has more potential than leveraging external knowledge. Moreover, we achieve superior performance to LOTClass even though we utilize the vanilla RoBERTa$_{\text{large}}$ without any training.

fully-supervised Results. The experiments in fully-supervised settings with long-tail distribution illustrate that RETROPROMPT achieves improvement compared with baselines. This indicates that our retrieval mechanism extends the LM's ability to learn hard examples in the fully-supervised datasets.

4.4 Model Generalization to New Domains

The scarce data may bring the overfitting problem for the lots of memory parameters of PLMs, even though prompt learning. Thus, we conduct cross-domain experiments to validate the generalization of our RETROPROMPT. Specifically, we utilize the model trained on the source datasets and directly test on the other target datasets. From Table 2, we can find that our method consistently outperforms baselines. This finding illustrates that RETROPROMPT achieves great model generalization to new domains.

Table 2: Results of model generalization to new domains.

Model	Source	Target Domain	
	16-shot MR	SST-2	CR
FT	76.9	71.4	64.7
LM-BFF (man)	87.0	88.9	86.9
LM-BFF (D-demo)	86.6	89.3	87.5
KPT	86.8	89.1	86.7
RETROPROMPT	**88.0**	**91.4**	**88.8**
	16-shot QQP	MRPC	RTE
FT	60.7	43.7	48.0
LM-BFF (man)	65.4	20.9	65.5
LM-BFF (D-demo)	68.2	38.8	66.2
KPT	71.6	42.3	65.8
RETROPROMPT	**74.0**	**49.4**	**67.3**

4.5 Analysis of Memorization

It is necessary and interesting to further explore the memorization mechanism to help us better understand the utility of retrieval for memorization in NLP.

Definition of Memorization Measurement. Inspired by the idea of [9] in the computer vision area, we define *memorization measures* as to how the classification varies when a training instance $\boldsymbol{z}$ is deleted from the trainset. We follow [24, 56] to define and derive the memorization score for a training instance $\boldsymbol{z}$ as follows:

$$\mathcal{S}_{\text{delate}}(\boldsymbol{z}) \stackrel{\text{def}}{=} -\left.\frac{dP(y|\boldsymbol{x};\hat{\theta}_{\xi,-\boldsymbol{z}})}{d\xi}\right|_{\xi=0} = -\nabla_\theta P(y|\boldsymbol{x};\hat{\theta})^\top \left.\frac{d\hat{\theta}_{\xi,-\boldsymbol{z}}}{d\xi}\right|_{\xi=0} = -\nabla_\theta P(y|\boldsymbol{x};\hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(\boldsymbol{z},\hat{\theta}), \tag{8}$$

where $\hat{\theta}_{\xi,-\boldsymbol{z}}$ denotes the parameters of the model trained with the instance $\boldsymbol{z}$ down-weighted by ξ, $\hat{\theta}$ is the parameters of the model trained with all instances and $H_{\hat{\theta}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 \mathcal{L}(z_i, \hat{\theta})$. Thus $\mathcal{S}_{\text{delate}}(\boldsymbol{z})$ is the amount of change of $P(y|x;\theta)$ when the instance $\boldsymbol{z}$ is down-weighted by a small amount ξ.

Top-memorized Instances: Typical or Atypical? Since the SST-2 dataset provides the annotations of phrase-level sentiment polarity labels, we adopt SST-2 to analyze the memorization by judging the atypical of an instance by checking the percentage of positive phrases. We collect such statistics from SST-2 and find that a typical positive instance has a relatively high percentage of positive phrases, and a typical negative instance should have a relatively low percentage of positive phrases. Based on the above observation, we apply the memorization score defined in Eq. 8 to select Top-10% and

Bottom-10% memorized instances from the trainset and collect the average percentage of positive phrases in these instances.

As shown in Table 3,we can conclude following findings: (1) **The PLM tends to give atypical samples deeper memory attention.** Specifically, no matter LM-BFF or our method, the top-10% memorized negative instances have a higher percentage of positive phrases than the average percentage of positive phrases of all negative instances. 2) LM-BFF has lower memorization scores on hard samples than fine-tuning. We think it owns to **prompt learning can help PLMs recall what they learned from pre-training without strengthening memory for downstream data.** 3) RETROPROMPT further has lower average memorization scores than fine-tuning and LM-BFF, which illustrates that our method is less memory dependent. This result may be attributed to **decoupling knowledge from memorization through retrieval to alleviating the rote of PLMs.**

Table 3: The upper part shows the average percentage of *positive phrases* over different memory groups of positive/negative instances. The lower part denotes the mean values of memorization score on the SST-2 dataset.

Mem Group	Negative			Postive		
	FT	LM-BFF	OURS	FT	LM-BFF	OURS
Top-10%	34.29	32.78	30.23	68.75	69.71	75.67
ALL		23.40			86.39	
Bottom-10%	17.63	16.25	14.42	95.92	95.08	94.53

	FT	LM-BFF	OURS
MEM SCORE	4.597	0.121	0.032

Case Analysis. As shown in Table 6, we manually list the top-ranked and bottom-ranked training instances of SST-2 according to our model. It reveals that the top-ranked memorized instances seem to show universal opinions indirectly. Thus, we inspect them as atypical/hard for sentiment classification. While those instances with 0 memorization scores are straightforward to show their opinion for sentiment classification, representing the typical instance. Note that $F(p_{kNN})$ is defined to represent the difficulty of the sample discriminated by kNN distribution. And the Table 6 also shows that $F(p_{kNN})$ indeed reflect atypicality of examples, which validate the effectiveness of the kNN guided training.

Table 4: Detailed ablation experiments in few-shot settings. "N-demo" donates the neural demonstration, and "refresh" refers to the asynchronous refresh of the knowledge-tore.

Model	16-shot				
	SST-2	CR	MNLI	QQP	TACRED
OURS	**93.9**	**91.9**	**71.1**	**74.0**	**40.7**
w/o kNN-test	93.2	91.2	70.4	73.0	38.2
w/o kNN-train	92.0	91.2	68.8	71.3	36.5
w/o N-demo	92.4	90.8	69.1	72.0	37.6
w/o refresh	93.5	91.5	70.7	73.6	39.9

4.6 Ablation Study

Component Ablation. As shown in Table 4, the performance of component ablation experiments with four variants has a clear drop, which proves the effectiveness of our retrieval component. We also find that neural demonstration and kNN-train have more improvement in the few-shot setting than kNN-test. Note that kNN-test is similar to kNN-LM [23, 14] and the results reveals that simply incorporate kNN in the test process of prompt learning has little influence in a few-shot setting.

Key Representation and kNN Acquisition. We study the effect of using different representations of the key in the knowledge-store. We experiment with two types of representations: (1) prompt-based representation, which is the default setting, and (2) [CLS] based representation of current LM. We also experiment with two types of calculation of kNN distribution: (1) representation based similarity score (refer as rep-similar), which is the default setting, and (2) BM25 based score , which calculates the correlation score between the query and each key examples with BM25 [45] algorithm. Results in Table 5 show that using prompt-based representations for key and representation based similarity scores for kNN leads to the best performance. It suggests that prompt learn better representations for context similarity and the representation similarity based kNN distribution is better than BM25 based scores.

Table 5: Performance on 16-shot CR and TACRED with different representations of key and calculate function of kNN distribution.

Key Repres.	kNN Acq.	CR	TAC.
Prompt	Rep-similar	91.9	40.7
`[CLS]`	Rep-similar	89.0	37.2
Prompt	BM25	89.5	38.8
`[CLS]`	BM25	88.7	36.1

Table 6: Case examples of Top-3 and Bottom-3 memorized instance of ours from trainset of SST-2.

Negative			Positive		
Content	**Mem**	$F(p_{k\text{NN}})$	**Content**	**Mem**	$F(p_{k\text{NN}})$
Although god is great addressed interesting matters of identity and heritage, it's hard to shake the feeling that it was intend to be a different kind of film.	0.066	1.17	A b-movie you can sit through, enjoy on a certain level and then forget.	0.020	0.18
A standard police-oriented drama that, were it not for deniro's participation, would have likely wound up a tnt original.	0.011	1.48	A film that will be best appreciated by those willing to endure its extremely languorous rhythms, waiting for happiness is ultimately thoughtful without having much dramatic impact.	0.010	0.43
A hit and miss affair, consistently amusing but not as outrageous or funny as cho may have intended or as imaginative as one might have hoped.	0.010	2.74	What's invigorating about is that it doesn't give a damn.	0.003	0.06
It's a loathsome movie, it really is and it makes absolutely no sense.	0.00	0.00	A fun family movie that's suitable for all ages– a movie that will make you laugh, cry and realize, 'it's never too late to believe in your dreams.'	0.00	0.00
It is that rare combination of bad writing, bad direction and bad acting – the trifecta of badness.	0.00	0.00	It's a cool event for the whole family.	0.00	0.00
This thing is virtually unwatchable.	0.00	0.00	Good fun, good action, good acting, good dialogue, good pace, good cinematography.	0.00	0.00

5 Related Work

Retrieval-enhanced PLMs. Our pipeline is partly inspired by discrete demonstration methods such as [11, 32, 46, 25, 26] that retrieves few training examples in a natural language prompt, while we propose neural demonstration for enhancing the input to alleviate the limitations of input length. Another line researches of retrieval augmentation [12, 20, 29] retrieve useful information from a external knowledge corpus (e.g., Wikipedia) for a particular task (e.g., an open-domain question). Unlike these works, we focus on retrieving examples from the internal training data. Besides, semi-parametric methods [23, 14, 22, 21, 1, 39] have risen to leverage k-nearest neighbor classifier that makes the prediction based on representation similarities, to enhance pre-trained language models. However, unlike these models using nearest neighbors only for augmenting the process of prediction, we aim to develop a comprehensive retrieval mechanism for input, training and test process.

Prompt learning for PLMs. With the birth of GPT-3 [3], prompt learning [33] has recently arisen to fill the gap between masked LM objective of PLMs and downstream fine-tuning objective. Prompt learning has achieves very impressive performance on various tasks [48, 49, 4, 37, 13, 5], especially under the setting of few-shot learning. Moreover, continuous prompts have also been proposed [30, 27, 34] to reduce prompt engineering, which directly appends a series of learnable continuous embeddings as prompts into the input sequence. Our work is orthogonal to previous prompt learning approaches, which aim to optimize prompts, while we focus on the systematic study of retrieving related examples from training data to enhance prompt learning.

6 Conclusion and Future Work

We propose RetroPrompt that decouples knowledge from memorization by introducing retrieval augmentation to further improve the generalization ability of prompt learning on the input side and the whole process of model training and prediction. RetroPrompt, is a straightforward yet effective retrieval method that combines both neural demonstrations, kNN guider for training and prediction. Our extensive results show that it outperforms other demonstration-enhanced prompt methods and knowledge-enhanced prompt methods in few-shot, zero-shot and fully-supervised settings. Analyzing the essence of memorization validates the effectiveness of decoupling knowledge from memorization. Interesting future directions include: 1) apply to other tasks, such as QA and NLG, 2) explore the noise data mining for unsupervised learning, 3) further improve the retrieve efficiency for large datasets, etc.

References

[1] Uri Alon, Frank F. Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. Neurosymbolic language modeling with automaton-augmented retrieval, 2022.

[2] Gianluca Bontempi, Hugues Bersini, and Mauro Birattari. The local paradigm for modeling and control: from neuro-fuzzy to lazy learning. *Fuzzy sets and systems*, 121(1):59–72, 2001.

[3] 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, and Dario Amodei. Language models are few-shot learners. In *Proceedings of NeurIPS 2020*, 2020.

[4] Xiang Chen, Ningyu Zhang, Lei Li, Xin Xie, Shumin Deng, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. Lightner: A lightweight generative framework with prompt-guided attention for low-resource NER. *CoRR*, abs/2109.00720, 2021.

[5] Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. *CoRR*, abs/2104.07650, 2021.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[7] Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. Few-nerd: A few-shot named entity recognition dataset. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3198–3213. Association for Computational Linguistics, 2021.

[8] Aparna Elangovan, Jiayuan He, and Karin Verspoor. Memorization vs. generalization : Quantifying data leakage in NLP performance evaluation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1325–1335, Online, April 2021. Association for Computational Linguistics.

[9] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 954–959. ACM, 2020.

[10] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[11] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of ACL*, 2021.

[12] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020.

[13] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. PTR: prompt tuning with rules for text classification. *CoRR*, abs/2105.11259, 2021.

[14] Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Efficient nearest neighbor language models. In *Proc. of EMNLP*, 2021.

[15] Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SemEval*, pages 33–38, 2010.

[16] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 168–177. ACM, 2004.

[17] Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. *CoRR*, abs/2108.02035, 2021.

[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547, 2021.

[19] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77, 2020.

[20] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics, 2020.

[21] Nora Kassner and Hinrich Schütze. Bert-knn: Adding a knn search component to pretrained language models for better QA. In *Findings of EMNLP*, 2020.

[22] Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[23] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[24] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017.

[25] Sawan Kumar and Partha Talukdar. Reordering examples helps during priming-based few-shot learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4507–4518, Online, 2021. Association for Computational Linguistics.

[26] Dong-Ho Lee, Mahak Agarwal, Akshen Kadakia, Jay Pujara, and Xiang Ren. Good examples make A faster learner: Simple demonstration-based learning for low-resource NER. *CoRR*, abs/2110.08454, 2021.

[27] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[28] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of ACL 2020*, 2020.

[29] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[30] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of ACL/IJCNLP 2021*, 2021.

[31] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 42, pages 318–327, 2020.

[32] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *CoRR*, abs/2101.06804, 2021.

[33] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*, 2021.

[34] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *CoRR*, abs/2103.10385, 2021.

[35] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[36] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *CoRR*, abs/2104.08786, 2021.

[37] Ruotian Ma, Xin Zhou, Tao Gui, Yiding Tan, Qi Zhang, and Xuanjing Huang. Template-free prompt tuning for few-shot NER. *CoRR*, abs/2109.13532, 2021.

[38] Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. Text classification using label names only: A language model self-training approach. In *Proceedings of EMNLP*, 2020.

[39] Yuxian Meng, Shi Zong, Xiaoya Li, Xiaofei Sun, Tianwei Zhang, Fei Wu, and Jiwei Li. GNN-LM: language modeling based on global contexts via GNN. *CoRR*, abs/2110.08743, 2021.

[40] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Kevin Knight, Hwee Tou Ng, and Kemal Oflazer, editors, *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 115–124. The Association for Computer Linguistics, 2005.

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style,

high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[42] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI*, 2018.

[43] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics, 2016.

[44] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics, 2019.

[45] Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, 2009.

[46] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. *CoRR*, abs/2112.08633, 2021.

[47] 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, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. 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 Févry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. *CoRR*, abs/2110.08207, 2021.

[48] Timo Schick, Helmut Schmid, and Hinrich Schütze. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of COLING*, December 2020.

[49] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of EMNLP 2020*, 2020.

[50] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL, 2013.

[51] Michael Tänzer, Sebastian Ruder, and Marek Rei. Memorisation versus generalisation in pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7564–7578, 2022.

[52] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652, 2021.

[53] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[54] Sen Yang, Yunchen Zhang, Leyang Cui, and Yue Zhang. Do prompts solve NLP tasks using natural language? *CoRR*, abs/2203.00902, 2022.

[55] Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of EMNLP 2017*, 2017.

[56] Xiaosen Zheng and Jing Jiang. An empirical study of memorization in NLP. *CoRR*, abs/2203.12171, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include the source code and data in our supplemental material submission, and we outline the data generation procedure, the evaluation protocol, the training regime, and everything else necessary for reproduction either in the main body of the paper or in the appendix.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Subsection 4.2 and Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We list the standard deviation for few-shot setting.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We introduce type of resources in Section 4.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] The code and the data are proprietary.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] The code and the data are proprietary.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]