Shepherd Pre-trained Language Models to Develop a Train of Thought: An Iterative Prompting Approach

Anonymous ACL submission

Abstract

While Pre-trained Language Models (PLMs) internalize a great amount of world knowledge, they have been shown incapable of recalling these knowledge to solve tasks requiring complex & multi-step inference procedures. Similar to how humans develop a "train of thought" for these tasks, how can we equip PLMs with such abilities? In this work, we explore an iterative prompting framework, a new prompting paradigm which progressively elicits relevant knowledge from PLMs for multi-step inference tasks. We identify key limitations of existing prompting methods, namely they are either restricted to queries with a single identifiable relation/predicate, or being agnostic to input contexts, which makes it difficult to capture variabilities across different inference steps. We propose an iterative context-aware prompter, which addresses these limitations by learning to dynamically synthesize prompts conditioned on the current step's contexts. Experiments on three datasets involving multi-step inference show the effectiveness of the iterative scheme and the context-aware prompter design.[1]

1 Introduction

Humans can develop a "train of thought" for complex decision making. For example, when asked the question (**Q**) shown in Figure 1, which involves composition, an important type of multi-step inference, humans apply two consecutive steps to derive the final answer: 1) find "father" of the topic entity "Gwilym Lloyd George" (**E1**); 2) find "birthplace" of the entity returned in the first step (**E2**).

Recently, large-scale pre-trained language models (PLMs) have been shown capable of internalizing a great amount of simple factual knowledge such as **E1** and **E2**, yielding competitive performance on a range of knowledge-intensive tasks without resorting to any external knowledge source (Petroni et al., 2019; Shin et al., 2020; Zhong et al., 2021; Roberts et al., 2020; Lee et al., 2020). However, work such as (Talmor et al., 2020a; Kassner et al., 2020; Rae et al., 2021) reveals that PLMs face difficulties in *complex, multi-step* inferences. For example, they struggle with answering complex questions like **Q** without using external sources, no matter whether they are fine-tuned based on QA pairs or simply prompted to produce the answer (where even if they have memorized **E1** and **E2**).

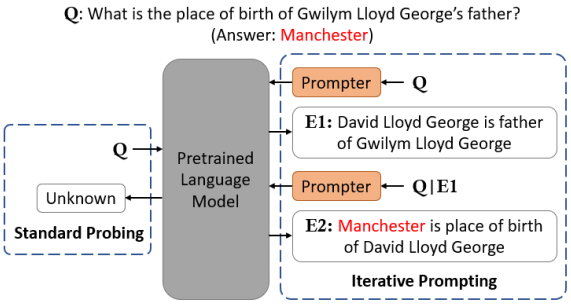

Figure 1: Our Iterative Prompting approach for deriving a "train of thoughts" with a PLM (on the right), compared with standard knowledge probing (on the left).

In this paper, we study the following question: How to shepherd a PLM to *recall a series of stored knowledge* (e.g., **E1** and **E2**) that is necessary for multi-step inference (e.g., answering **Q**), analogous to how humans develop a "train of thought" for complex decision making?

A direct way would be to fine-tune the PLM to generate the series of knowledge all at once (assuming such supervision is available), but soon one realizes the practical issue in this approach: PLMs which internalize a great amount of knowledge are inevitably large in scale, and fine-tuning all their parameters would become more and more costly as they keep scaling up. There's also the potential concern that fine-tuning PLMs may interfere with their implicit knowledge storage, a phenomenon observed in (Wang et al., 2021) which is more generally related to the catastrophic forgetting problem of deep learning models (McCloskey and Cohen,

[1] Our source code will be released upon acceptance.

1989; Kirkpatrick et al., 2017; Howard and Ruder, 2018). Therefore, lightweight methods such as prompting (Liu et al., 2021) which keep a PLM's parameters intact would be more preferable for our purpose of eliciting knowledge. However, we find that no matter whether it is fine-tuned or prompted to generate the series of knowledge all at once, the PLM tends to lose its "train of thought" during the process, generating irrelevant facts or suffering from hallucination.

Motivated by the iterative nature of multi-step inference problems, we explore an *iterative prompting* framework in this paper, which elicits knowledge from PLMs step by step for a given inference task. We have two desiderata in iterative prompting: (1) At different inference steps, the prompts need to focus on different components of the complex query. (2) The prompts should appropriately integrate knowledge gathered in previous steps into the current step; for instance, during the 2nd step in the example in Figure 1, the prompts need to combine the entity "David Lloyd George" (from knowledge recalled in the 1st step) with the unresolved part "What is the place of birth of" in the query.

A natural thought is to directly apply existing prompting methods in an iterative fashion. Unfortunately, their prompts are either restricted to queries with a single, identifiable relation/predicate (Jiang et al., 2020; Petroni et al., 2019; Zhong et al., 2021; Shin et al., 2020; Qin and Eisner, 2021), or being agnostic and insensitive to step-wise inputs (Lester et al., 2021; Li and Liang, 2021; Brown et al., 2020), and hence not ideal for our desiderata.

We design a novel iterative prompting method towards that end. We augment the PLM with an *iterative Context-Aware Prompter*, a model which learns to *dynamically synthesize prompts* based on the current step context. At each step, the Prompter learns to process the query and previously gathered evidence, and composes a prompt which steers the PLM to recall the next piece of knowledge. Like other prompting methods, the PLM is kept fixed throughout the learning process. In addition, as the PLM size increases, the number of trainable parameters in our method scales comparably with or slower than previous prompting methods.

We conduct experiments on three datasets involving multi-step inference, including two recent multi-hop Question Answering datasets: 2WikiMultiHopQA (Ho et al., 2020) and R4C (Inoue et al., 2020), and a scientific dataset (Talmor et al., 2020b) for reasoning over taxonomic relations. Our experimental results show (1) effectiveness of the iterative scheme; (2) our proposed Context-Aware Prompter design outperforms existing prompting methods by notable margins; (3) quantitative and qualitative analysis which reveal the faithfulness of our learned prompter.

2 Methodology

In this section, we first formalize our problem setup (§2.1), and then introduce our iterative prompting framework (§2.2), followed by our context-aware prompter design (§2.3) which addresses key limitations of previous prompting methods when applied in this iterative scheme.

2.1 Problem Setup

Given a complex query q, our goal is to drive a PLM $\mathcal{M}$ to recall a sequence of simple knowledge statements $C_q = [c_1, ..., c_{n_q}]$ which is sufficient for deciding the response to q. In particular, we focus on developing prompting methods, where the parameters of $\mathcal{M}$ are fixed and we aim to construct prompt T which steer $\mathcal{M}$ to recall C_q. Note that here we treat T as a variable, which may or may not depend on other variables based on different modellings. Writing $\mathcal{M}^T$ as $\mathcal{M}$ augmented with prompt T, our training objective is to learn how to find T which could maximize the log likelihood:

$$\mathcal{L}(T) = \sum_{i=1}^{N} \log P(C_{q_i}|q_i; \mathcal{M}^T)$$

with a set of training data $\{q_i, C_{q_i}\}_{i=1}^{N}$.

Our formulation here is general and applicable to all prompting based methods, where the settings in previous work such as (Zhong et al., 2021; Shin et al., 2020; Lester et al., 2021; Li and Liang, 2021; Qin and Eisner, 2021) correspond to the reduced case where $|C_q| = 1$ for any query q. In our experiments we also consider PLM fine-tuning, in which case there's no prompt T in the pipeline and instead the parameters of $\mathcal{M}$ are optimized.

2.2 Iterative Prompting Framework

Inspired by the sequential nature of multi-step inference tasks, we approach the problem in an iterative way:

$$P(C_q|q; \mathcal{M}^T) = \prod_{j=1}^{n_q} P(c_j|q, c_1, ..., c_{j-1}; \mathcal{M}^T)$$

where at each step j, $\mathcal{M}^T$ recalls the next piece of knowledge c_j conditioned on the query q and all previously gathered knowledge $c_1, ..., c_{j-1}$ (concatenated with q).

2.3 Context-Aware Prompter

Previous prompting methods which take single-relation inputs clearly fail to apply in this iterative setting due to the complexity of the input context $q, c_1, ..., c_{j-1}$. Task-level prompting methods such as Prompt-Tuning (Lester et al., 2021) and Prefix-Tuning (Li and Liang, 2021) are applicable here, where T is treated as a *static* parameter. However, as described earlier, this modelling is not ideal for T to fully capture variabilities across different inference steps. In this work, we model T as the output of our *Prompter*, a learnable function mapping f_W which dynamically synthesizes T *w.r.t.* the current step input context:

$$T = f_W(q, c_1, ..., c_{j-1}), \forall j$$

Prompter Instantiation. While there are many plausible design choices for the Prompter f_W, here we instantiate it with a transformer-based language model (shown in Figure 2). The prompts are designed to be contextualizations (by the Prompter) of a set of special tokens *w.r.t.* the current step input context, linearly projected into the PLM's embedding space by a trainable matrix (omitted in the figure due to space limit). In this work, we adopt an Encoder-Decoder PLM and use prefix-prompts in the implementation; hence we have prompts that are prepended to both the PLM's encoder inputs and decoder inputs. Note that our design could be easily adapted to other types of PLMs (e.g., encoder-only/decoder-only models) and different prompt positionings (e.g., infix, postfix).

Comparison with Prompt/Prefix-Tuning. Both Prompt-Tuning (Lester et al., 2021) and Prefix-Tuning (Li and Liang, 2021) model the prompt T as a context-agnostic parameter. In Prompt-Tuning, T has the same identity as in our approach which is a set of virtual input tokens (*Encoder Prompts* & *Decoder Prompts* in Figure 2). In Prefix-Tuning, T is modelled to be the set of activations (keys & values in the transformer attention blocks) of the virtual prompt tokens across all PLM layers. Let D be the embedding dimension of the PLM, h be the number of layers in the PLM, d be the embedding dimension of the Prompter ($d \leq D$) and l be the length of the prompt tokens (both encoder & decoder prompts). Then the number of trainable parameters is $\Theta(d \cdot (D + l))$ for our proposed method, $\Theta(l \cdot D)$ for Prompt-Tuning and $\Theta(l \cdot h \cdot D)$ for Prefix-Tuning. It can thus be seen that our proposed method scales comparatively with Prompt-Tuning, slower than Prefix-Tuning, and overall maintains a manageable amount of trained parameters as the PLM scales up (which increases D and h).

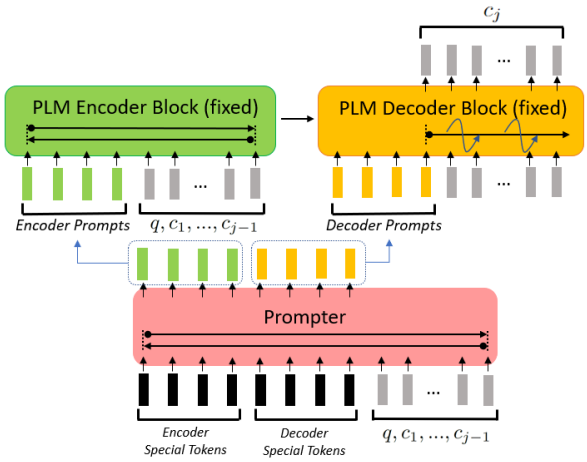

Figure 2: Our context-aware prompter design. The prompter contextualizes a set of special tokens *w.r.t.* the current step context $q, c_1, ..., c_{j-1}$ to get the resulting prompts, which steers the PLM to recall the next piece of knowledge c_j.

Continuous v.s. Discrete Prompts. While modelling T as discrete tokens in the PLM's vocabulary could increase the readability of the prompts, a discrete space is much less expressive than its continuous counterpart, and optimization over a discrete space could be highly inefficient. Also, despite being inside the vocabulary, the searched discrete prompts could still have low interpretabilities as seen by the given examples in (Shin et al., 2020). Hence, we follow prior work (Zhong et al., 2021; Li and Liang, 2021; Lester et al., 2021; Qin and Eisner, 2021) and model the prompts to be continuous virtual tokens instead of discrete tokens.

2.4 Learning and Inference

We use teacher-forcing for model training, namely at each step, the ground truth contexts at that step (query and previous knowledge pieces) are presented to the model. We maximize $\mathcal{L}(T)$ using standard sequence-to-sequence (seq2seq) learning objectives. During inference, we proceed autoregressively by feeding the recalled knowledge at step $t-1$ as the additional context at step t, and execute for some predefined number of steps.

3 Experimental Setup

3.1 Datasets & Preprocessing

We conduct experiments on three datasets involving multi-step reasoning which include annotations for knowledge statements relevant to the queries: 2WikiMultiHopQA (abbreviated as 2Wiki) (Ho et al., 2020), R4C (Inoue et al., 2020), and a scientific commonsense reasoning dataset (abbreviated as LoT[2]) constructed by (Talmor et al., 2020b).

2WikiMultiHopQA (Ho et al., 2020). 2Wiki is a recent large scale multi-hop QA dataset, which contains in total over 192k (167k train, 12.5k development and 12.5k test) samples constructed jointly from Wikipedia and Wikidata. Since the test set is private, we randomly split the original development set into our development & test set (6k samples each). The dataset format largely follows HotpotQA (Yang et al., 2018), but includes more diverse reasoning types of questions and detailed annotations of evidence paths for each question. Here, an evidence path is an ordered list of (subject entity, relation, object entity) knowledge base triplets. We use the question as the query q, and use a simple template to convert each triplet in the evidence path into a natural language statement, forming C_q. Due to the large training set size and limited computing budget, we randomly sample 10% of the training data to form our final training set, which has the side benefit of largely reducing the test/train overlap (more details in §4.2).

R4C (Inoue et al., 2020). R4C is another recent multi-hop QA dataset containing annotated evidence paths. The dataset contains 4.6k examples (2.4k train, 2.2k development) constructed on top of HotpotQA, where the authors used crowdsourcing efforts to collect the evidence paths in the form of simple subject-verb-object natural language sentences. Again, we randomly split the development set (there's no test set given) into our development and test set (1.1k samples each). We use the question as our query q and use the annotated evidence sentences as C_q.

LoT (Talmor et al., 2020b). The dataset involves reasoning over a set of taxonomic relations, constructed from ConceptNet and WordNet. Each example consists of a hypothesis (e.g., "A whale has a belly button") which we treat as query q, and a set of simple facts including hypernym rules (e.g., "A whale is a mammal", "A whale is a vertebrate") and properties (e.g., "A mammal has a belly button", "A vertebrate has a tail"). By reasoning over the facts and selecting the correct chain of hypernym rule & property ("A whale is a mammal", "A mammal has a belly button"), one could verify or deny the given hypothesis. One subtle issue of directly using the gold hypernym rule and property as C_q is, during the first step, it would be difficult to directly identify the correct object entity without looking ahead on the properties in the second step. Therefore, for the first step, we concatenate all the hypernymic objects appearing in the dataset *w.r.t.* to the same subject to form c_1. We drop samples from the original training set where the relevant facts are not (or only partially) provided, and obtain 9.4k/1.2k/1.2k samples for training/development/testing.

For 2Wiki and R4C, the number of steps during inference is set to be 4 since over 99% of the samples have less or equal than 4 inference steps. For LoT, we set the number of inference steps to be 2. Overall, we regard 2Wiki as our "major" evaluation dataset due to its largest scale (despite our downsampling) and diverse types of queries, and use it to conduct a faithfulness study of prompting in §4.2. Some examples of the processed data samples are shown in Appendix A.4.

3.2 Evaluation Metric

We use both intrinsic and extrinsic metrics to evaluate the PLM recalled knowledge.

Intrinsic Evaluation. Here, we directly measure the quality of recalled knowledge. While there are standard metrics for evaluating text generation such as BLEU and ROUGE, these metrics generally fail to capture the entity-centric nature of the recalled knowledge we wish to examine. Therefore, we propose a set of measures which are better suited for the tasks in our experiments. For 2Wiki and R4C, we evaluate the ratio where the recalled knowledge contains the answer entity (**Ans.**R); we also compute the ratio among only those samples where the answer entity does not appear in the query (**Ans.**$\hat{R}$). For 2Wiki, since the annotated evidence are KB triples, we additionally evaluate the entity coverage of recalled contexts by computing the average ratio of gold entities appearing in the recalled contexts (**Ent.**R) and the ratio of samples where *all* gold entities are recalled (**Ent.**R^*) as a more strict measure. For LoT, at the first step, we deem the hypernym rule as correct if the gold object is recalled, and use exact match for the recalled property at the second

[2]The abbreviation here comes from the phrase "Leap-of-Thought" in the paper title of (Talmor et al., 2020b).

step. We compute the ratio of correctly recalled hypernym rule/property (**Cont.**R) and the percentage of samples where the recalled hypernym rule & property are both correct (**Cont.**R^*).

Extrinsic Evaluation. We also conduct extrinsic evaluation by measuring how much the recalled knowledge help find the response to the query. Similar to reading comprehension, we concatenate all recalled knowledge as the contexts, and use a reader which tries to infer the answer given the query and contexts. For 2Wiki and R4C, we first pre-train the reader using the ground truth contexts, and then fine-tune it on the recalled contexts[3]; for LoT, we use a rule-based reader directly[4]. We report Exact Match (EM) and Answer F1 scores for 2Wiki & R4C, and EM score for LoT where the answer is restricted to yes/no.

3.3 Compared Methods

We compare our proposed iterative Context-Aware Prompter (**iCAP**) along with Prompt Tuning (**Prompt-T**), Prefix Tuning (**Prefix-T**) and PLM fine-tuning (**PLM-FT**) under both non-iterative and iterative setting. The iterative setting is described in §2.2 and for the non-iterative setting, we simply concatenate all the knowledge statements in C_q to form one single piece of knowledge for each query. In extrinsic evaluation, we also compare with fine-tuning the PLM on (query, answer) pairs without knowledge recall (**PLM-QA**), which measures how much the PLM can solve these multi-step inference problems directly, a skill which PLMs are poor at as shown by previous work. We additionally report final inference results when feeding ground truth contexts to the reader (**Oracle-RD**) as an upper bound for extrinsic evaluation. Relation-specific prompting methods such as (Shin et al., 2020; Zhong et al., 2021; Petroni et al., 2019) are not included since they're not directly applicable in our problem setup as discussed earlier.

Our focus in this work is on *knowledge elicitation from PLMs*, and hence we do not aim to compare with previous dataset-specific methods which typically have different problem formulations & focus than ours and utilize other attributes in the datasets which we do not use (e.g., gold & distractor evidence paragraphs).

[3]We found in our preliminary experiments that this approach gives the best results across different methods.

[4]LoT is constructed using templates, and therefore a rule-based reader can perfectly solve the inference task (100% accuracy when ground truth contexts are given, see Table 2).

3.4 Implementation Details

Architectures & hyperparameters. We use BART-large (Lewis et al., 2020) for our PLM and RoBERTa-base (Liu et al., 2019) for our prompter, which is several times smaller than the PLM[5]. We also include some results & discussion for different prompter scales in Appendix A.5. We use another BART-large for the reader in extrinsic evaluation[6].

Our implementation is based on the Hugging Face Transformer library (Wolf et al., 2020). We use AdamW optimizer (Loshchilov and Hutter, 2019) and a linear learning rate scheduler with warmup ratio 0.06 for optimization. For hyperparameters, we use a batch size of 32, 128, 32 for 2Wiki, LoT and R4C respectively, and tune the learning rate from {4e-5, 8e-5, 4e-4, 8e-4, 4e-3, 8e-3, 4e-2} & length of encoder/decoder prompts[7] from {15, 30, 45, 60, 80, 100}; more details are included in Appendix A.1. We run most experiments with three random seeds, and report the average scores with standard deviations in the subscripts.

Knowledge Enhancement for PLM. Since our focus is on how to make PLMs better at recalling relevant knowledge for multi-step inference, we need to make sure the PLM actually memorizes all the relevant knowledge in the first place, so that the results can be attributed solely to the effectiveness of knowledge recall. Hence, we conduct knowledge enhancement for the PLM, where we additionally pre-train the PLM to recover separately masked elements in the triplets which form the knowledge statements, a strategy similar to salient span masking (Roberts et al., 2020; Guu et al., 2020). More details could be found in Appendix A.2. Note the same PLM after knowledge enhancement is used across different compared methods.

4 Results

4.1 Effectiveness of iCAP

The results for intrinsic & extrinsic evaluation are summarized in Table 1 and 2 respectively, which are highly consistent. We elaborate the results in what follows.

[5]While our prompter is also initialized using a Pre-trained Language Model, we'll use the term "PLM" to refer only to the larger & more knowledgeable one.

[6]For the reader, we intentionally choose the same architecture with the PLM for fair comparison with **PLM-QA**.

[7]We set the length of encoder & decoder prompts to be the same, as we do not observe improvements otherwise in preliminary experiments.

	2Wiki				LoT		R4C	
	Ent.R^*	**Ent.**R	**Ans.**$\hat{R}$	**Ans.**R	**Cont.**R^*	**Cont.**R	**Ans.**$\hat{R}$	**Ans.**R
PLM-FT	$10.3_{\pm 0.6}$	$33.8_{\pm 1.4}$	$12.3_{\pm 1.4}$	$45.3_{\pm 1.6}$	$41.8_{\pm 1.2}$	$70.8_{\pm 0.8}$	$38.1_{\pm 1.2}$	$43.9_{\pm 1.1}$
PLM-FT (Iter)	$26.3_{\pm 0.7}$	$48.9_{\pm 0.6}$	$35.4_{\pm 0.7}$	$60.6_{\pm 0.3}$	$41.3_{\pm 1.6}$	$70.1_{\pm 1.2}$	$43.1_{\pm 1.2}$	$48.5_{\pm 1.0}$
Prompt-T	$5.5_{\pm 0.1}$	$22.3_{\pm 0.4}$	$6.6_{\pm 0.1}$	$41.3_{\pm 0.6}$	$35.3_{\pm 1.6}$	$62.8_{\pm 1.9}$	$28.2_{\pm 0.9}$	$33.4_{\pm 0.4}$
Prompt-T (Iter)	$10.8_{\pm 0.4}$	$27.5_{\pm 0.5}$	$16.7_{\pm 0.7}$	$46.2_{\pm 0.8}$	$33.3_{\pm 1.5}$	$63.4_{\pm 1.8}$	$30.6_{\pm 2.2}$	$36.0_{\pm 2.4}$
Prefix-T	$6.7_{\pm 0.1}$	$25.9_{\pm 0.2}$	$7.6_{\pm 0.1}$	$44.2_{\pm 0.1}$	$31.8_{\pm 0.2}$	$64.0_{\pm 0.1}$	$27.2_{\pm 0.4}$	$33.9_{\pm 0.3}$
Prefix-T (Iter)	$14.8_{\pm 0.4}$	$33.9_{\pm 0.5}$	$22.5_{\pm 0.4}$	$53.2_{\pm 0.3}$	$31.6_{\pm 0.3}$	$64.9_{\pm 0.3}$	$33.7_{\pm 0.6}$	$39.8_{\pm 0.6}$
iCAP	$22.0_{\pm 1.7}$	$42.1_{\pm 2.2}$	$28.6_{\pm 2.7}$	$54.6_{\pm 1.6}$	$34.1_{\pm 0.7}$	$65.0_{\pm 0.2}$	$36.8_{\pm 0.7}$	$41.5_{\pm 0.6}$

Table 1: Intrinsic Evaluation Results. "(Iter)" indicates the iterative setting of a method. All metrics are defined in §3.2 and overall measure the gold (answer) entity/object coverage of the recalled knowledge from different perspectives.

	2Wiki		LoT	R4C	
	EM	F1	EM	EM	F1
Oracle-RD	97.8	98.9	100.0	75.7	86.8
PLM-QA	$24.1_{\pm 0.2}$	$29.3_{\pm 0.4}$	$68.3_{\pm 0.4}$	$22.6_{\pm 0.7}$	$28.8_{\pm 1.0}$
PLM-FT	$33.6_{\pm 0.6}$	$37.8_{\pm 0.5}$	$76.0_{\pm 0.6}$	$25.3_{\pm 1.2}$	$36.8_{\pm 0.9}$
PLM-FT (Iter)	$45.5_{\pm 1.4}$	$50.9_{\pm 1.7}$	$77.8_{\pm 0.3}$	$32.2_{\pm 0.0}$	$42.5_{\pm 0.5}$
Prompt-T	$26.9_{\pm 0.6}$	$31.0_{\pm 0.6}$	$65.9_{\pm 1.1}$	$16.6_{\pm 2.9}$	$25.9_{\pm 2.8}$
Prompt-T (Iter)	$25.0_{\pm 0.6}$	$30.2_{\pm 0.2}$	$68.8_{\pm 1.1}$	$22.4_{\pm 2.3}$	$30.4_{\pm 2.7}$
Prefix-T	$31.6_{\pm 0.1}$	$35.6_{\pm 0.2}$	$69.0_{\pm 0.2}$	$19.2_{\pm 0.4}$	$29.2_{\pm 0.5}$
Prefix-T (Iter)	$31.1_{\pm 0.7}$	$36.4_{\pm 0.4}$	$72.6_{\pm 0.3}$	$24.0_{\pm 0.3}$	$34.2_{\pm 0.4}$
iCAP	$42.8_{\pm 2.5}$	$47.9_{\pm 2.3}$	$73.8_{\pm 0.3}$	$25.7_{\pm 1.9}$	$35.2_{\pm 1.9}$

Table 2: Results for Extrinsic Evaluation, where the recalled knowledge of each method is used for final inference, except for Oracle-RD and PLM-QA.

Effectiveness of Iterative Scheme & Context-Aware Prompter. Across different datasets, it can be seen that most compared methods benefit from the iterative setting (Iter) over the non-iterative setting. Moreover, our proposed iterative Context-Aware Prompter (**iCAP**) further outperforms Prompt/Prefix Tuning by notable gains across different datasets and metrics, approaching the performance of PLM fine-tuning (**PLM-FT**); in particular, on the 2Wiki dataset which has the largest scale and diversity of reasoning types, **iCAP** achieves more than 15% and 10% absolute gains in F1 over Prompt-Tuning & Prefix-Tuning respectively. Overall, the results clearly show the effectiveness of the iterative scheme and our proposed context-aware prompter design. However, we note that even the best results (prompting based or fine-tuning based) still far lag behind **Oracle-RD** which uses ground truth contexts as input, which suggests a large room for improvements with better methods for knowledge elicitation from PLMs. Some failure cases of iCAP are included in Appendix A.4.

Helpfulness of Knowledge Recall for Multi-step Inference. The result obtained by fine-tuning the PLM on (query, answer) directly without knowledge recall (**PLM-QA**) is outperformed by almost all other compared methods, verifying the previous findings that PLMs face difficulties on using their stored knowledge to perform multi-step inference tasks. The large gain obtained from methods based on knowledge recall shows the helpfulness of deriving a "train of thought" (especially iteratively) from PLMs for multi-step inference.

4.2 Faithfulness of Prompting

(Zhong et al., 2021) raised and studied some important questions in optimization-based prompting methods: *Are the prompts "really" doing prompting? Is it possible that they capture dataset regularities too?* The issue is related to the notion of test-train overlap (Lewis et al., 2021), where the dataset may contain some underlying spurious patterns that the model exploits, and thus standard evaluations could not truthfully measure their generalization behaviors. Here, we take this concern seriously and conduct a series of analysis to more faithfully interpret the results we obtained. We focus on 2Wiki under iterative setting for our analysis.

Test-Train Overlap. For each development & test sample, we compute the ratio of knowledge statements in C_q that also appear in training set, meaning that during certain step for some training sample, the model has "seen" the exact same piece of knowledge. Note that this is a rather strict measure: even if all the knowledge pieces in C_q are seen during training, they may come from completely different samples & steps and hence organized in different ways. We summarize the overlapping ratios of development & test set samples in Table 5 in Appendix A.3. It can be seen that the down-sampling has the side benefit of greatly reducing the test-train overlap; in particular, the percentage of examples where all knowledge statements are seen during training is reduced from almost 30%

	Random Model				Random Embedding			
	Ent.R^*	**Ent.**R	**Ans.**$\hat{R}$	**Ans.**R	**Ent.**R^*	**Ent.**R	**Ans.**$\hat{R}$	**Ans.**R
PLM-FT	1.77	5.20	3.76	37.48	4.10	11.47	6.52	37.18
Prompt-T	0.0	0.0	0.0	0.0	0.006	0.013	0.003	0.002
Prefix-T	0.001	0.0	0.0	0.0	0.009	0.014	0.004	0.002
iCAP	0.001	0.001	0.0	0.0	1.49	2.83	0.98	0.59

Table 3: Intrinsic Evaluation Results on Random Control Experiments. Here we only focus on the iterative setting using the 2Wiki dataset.

to less than 2%, and more importantly, over 70% of the samples have no overlap. This suggests a rather low risk for the existence of strong spurious regularities in our setup.

Random Control Experiments. Examining the data-level statistics are helpful, but still not sufficient in terms of revealing the spurious regularities that different methods may capture. Hence, we follow (Zhong et al., 2021) to conduct two random control experiments. In the *Random Model* experiment, we re-initialize all parameters of the PLM to clean out its internal knowledge, and proceed with the same training procedure as earlier. In this way, any positive signal obtained could only be attributed to dataset regularities captured by the method. In the *Random Embedding* experiment, we re-initialize only the input embeddings of the PLM, a setting analogous to the *control task* introduced in (Hewitt and Liang, 2019) (more discussions can be found in (Zhong et al., 2021)). Here we only proceed with the iterative setting and conduct intrinsic evaluation, where the results are summarized in Table 3. It can be seen that 1) PLM fine-tuning captures significantly more regularities in the dataset than prompting-based methods; 2) While our proposed method captures a bit more regularities than Prompt/Prefix Tuning, they still remain at a very small level. Overall, our random control experiments show that the exploitation of spurious dataset patterns by the evaluated prompting methods is rather mild, and that by PLM fine-tuning could potentially be larger.

Prompter Attention Visualization. To see whether our proposed iCAP behaves in the way we expect, one direct approach is to examine the inner workings of the prompter. Towards this end, we visualize the attentions during the prompter forward pass at different steps. We randomly choose examples in the development/test set, and use BertViz (Vig, 2019) to visualize the attentions within the forward pass of the prompter after the following processing steps: 1) we aggregate the attention weights of different attention heads within the same transformer layer; 2) to better view the prompt tokens as one single unit, we average the attentions across different prompt tokens to form one "master" prompt token; 3) we drop all special tokens (BOS, EOS) for cleaner visualization. One example (the same example which we use in Figure 1) is in Figure 3, and we include more examples in Appendix A.6. As briefly illustrated earlier in §1, during the 1st step towards solving this query, the prompter should focus on the part concerning "father" of "Gwilym Lloyd George"; during the 2nd step, the prompter should integrate the answer "David Lloyd George" from the 1st step evidence and the "place of birth" part in the query to synthesize the prompt. We can see that the attention distributions at different steps accord well with our expectations. However, we note that attention visualization is only a qualitative approach; more systematic ways for examining the inner working behaviors of transformers remains an open challenge.

5 Related Work

Memorization and Reasoning in PLMs. With the recent success of large-scale pre-trained language models (PLMs), there has been growing interest in investigating what is captured by these PLMs during pre-training (Talmor et al., 2020a; Rogers et al., 2020; Kassner et al., 2020). Studies have shown that in addition to learning linguistic knowledge about language use, PLMs are capable of memorizing a great amount of world knowledge (Rogers et al., 2020), yielding competitive performance on knowledge probing (Petroni et al., 2019; Shin et al., 2020; Zhong et al., 2021) and other knowledge-intensive tasks such as question answering (Roberts et al., 2020) and fact checking (Lee et al., 2020), without resorting to any external knowledge source.

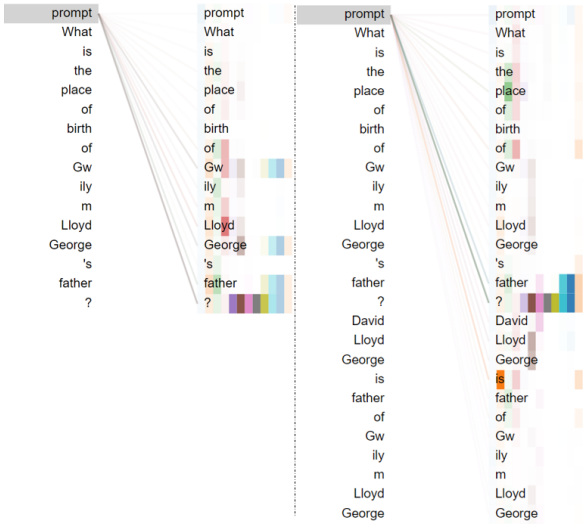

Figure 3: Prompter Attention Visualization. Attentions during the forward pass for the 1st & 2nd step are shown in the left & right respectively. Different colors correspond to different transformer layers. More examples of different reasoning types are included in Appendix A.6.

On the other hand, other work such as (Talmor et al., 2020a; Kassner et al., 2020; Rae et al., 2021) reveals that PLMs face difficulties in recalling their stored knowledge for *multi-step* inferences (such as answering complex, multi-hop questions), which is also verified in our experiments.

Prompt Learning. One type of methods for eliciting knowledge from PLMs is prompting (Liu et al., 2021), which is gaining increasing research interests & potential recently. Prompting methods seek to re-frame queries into prompts which accord with the PLM's input format, and extract useful information from the predicted results. The benefit of not needing to tune PLMs makes prompting especially appealing as PLMs scale up in size. In this work, we are interested in developing prompting methods which could enable PLMs to recall a series of relevant knowledge for multi-step inference. Previous work along this direction mainly use manually designed prompts/templates suited for specific datasets (Paranjape et al., 2021; Mishra et al., 2021; Shwartz et al., 2020; Wei et al., 2022); instead, we seek to develop a general method which can learn to construct appropriate prompts automatically.

For existing work on learning-based prompting, (Shin et al., 2020) proposes to use gradient-guided search to find appropriate discrete prompt tokens in the PLM's vocabulary to form prompt templates. While the resulting prompts are readable, most of them have very low fluency and interpretability. (Zhong et al., 2021; Qin and Eisner, 2021) propose to optimize the prompts in continuous space instead, which shows large benefits in both effectiveness and optimization efficiency. (Zhong et al., 2021) also raises and studies the question about whether learning-based prompting could exploit spurious dataset regularities which would weaken the validity of standard evaluation results, a concern we seriously address in our work. (Lester et al., 2021; Li and Liang, 2021) follow the continuous prompting paradigm, and tune task-level prompts for lightweight adaptation of PLMs. Overall, existing prompt learning methods are either restricted to cases where there exists a single & identifiable relation/predicate within the query (Zhong et al., 2021; Qin and Eisner, 2021; Shin et al., 2020), or being static and not sensitive to sample-wise inputs (Lester et al., 2021; Li and Liang, 2021).

Iterative Knowledge Retrieval. We are also inspired by methods that iteratively retrieve knowledge from explicit knowledge sources for multi-step reasoning, such as (Xiong et al., 2021; Qi et al., 2019; Khattab et al., 2021). Our problem setting could be viewed as iterative retrieval over implicit knowledge in PLMs, instead of from explicit knowledge sources.

6 Conclusion & Future Work

We explore an iterative prompting framework towards driving a "train of thought" from PLMs for multi-step reasoning tasks. We show the superiority of this iterative scheme, and also effectiveness of our proposed context-aware prompter design, which addresses key limitations of previous prompting methods when applied in this new scheme. In addition, we conduct both quantitative & qualitative analysis on the faithfulness of the learned prompting behaviors. In the future, we aim to further extend and apply our ideas to language model pretraining, with the hope that PLMs can be inherently equipped with stronger multi-step reasoning capabilities. The iterative framework we explore here also opens the possibility of human intervention and interaction during inference; namely a human can track along the PLM's train of thought and make edits and corrections at different steps, which improves the transparency and trustworthiness of inference and also helps reduce error propagation along the reasoning process. We leave these investigations as future work.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Naoya Inoue, Pontus Stenetorp, and Kentaro Inui. 2020. R4C: A benchmark for evaluating RC systems to get the right answer for the right reason. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6740–6750, Online. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Nora Kassner, Benno Krojer, and Hinrich Schütze. 2020. Are pretrained language models symbolic reasoners over knowledge? In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 552–564, Online. Association for Computational Linguistics.

Omar Khattab, Christopher Potts, and Matei Zaharia. 2021. Baleen: Robust multi-hop reasoning at scale via condensed retrieval. *Advances in Neural Information Processing Systems*, 34.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Nayeon Lee, Belinda Z. Li, Sinong Wang, Wen-tau Yih, Hao Ma, and Madian Khabsa. 2020. Language models as fact checkers? In *Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER)*, pages 36–41, Online. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1000–1008, Online. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of

prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2021. Reframing instructional prompts to gptk's language. *arXiv preprint arXiv:2109.07830*.

Bhargavi Paranjape, Julian Michael, Marjan Ghazvininejad, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2021. Prompting contrastive explanations for commonsense reasoning tasks. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4179–4192, Online. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. 2019. Answering complex open-domain questions through iterative query generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2590–2602, Hong Kong, China. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629, Online. Association for Computational Linguistics.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020a. olmpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. 2020b. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. In *Advances in Neural Information Processing Systems*, volume 33, pages 20227–20237.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Cunxiang Wang, Pai Liu, and Yue Zhang. 2021. Can generative pre-trained language models serve as knowledge bases for closed-book QA? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3241–3251, Online. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen,

Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online. Association for Computational Linguistics.

A Appendix

A.1 Hyperparameters

We set the batch size to be 32, 128, 32 and train for 70, 50, 40 epochs for 2Wiki, LoT & R4C respectively. Table 4 summarizes other hyperparameters used in our experiments.

	2Wiki		**LoT**		**R4C**	
	lr	pt_len	lr	pt_len	lr	pt_len
Prompt-T	8e-3	80	4e-3	80	4e-3	60
Prefix-T	8e-4	80	4e-4	60	4e-4	80
PLM-FT	4e-5	-	4e-5	-	4e-5	-
PLM-QA	4e-5	-	8e-5	-	4e-5	-
Ours	8e-5	30	8e-5	60	8e-5	30

Table 4: Hyperparameter settings for all compared methods. lr: learning rate, pt_len: prompt length.

A.2 More details on PLM Knowledge Enhancement

To make sure the PLM knows all the relevant knowledge for subsequent recall, we further pretrain the PLM to recover separately masked elements in the triplets which form the knowledge statements. For 2Wiki and LoT, we also additionally include knowledge statements that are not used in the dataset to make the setting more challenging; one can think of these extra knowledge statements as "distractors". For 2Wiki, we filter from the processed Wikidata triples provided by (Agarwal et al., 2021) by keeping those with subject entities appearing in the original knowledge statements, and in the end we obtain 383k extra knowledge statements v.s. 240k original ones (note that while we downsample the training set during our main experiment, the knowledge enhancement step is performed on the full dataset). For LoT, we directly use the provided distractor knowledge in the original dataset. We don't add distractors for R4C because the provided knowledge statements are in natural language and it's hard to retrieve high quality knowledge statements as such. We verified that the PLM after knowledge enhancement can indeed recover the masked elements in the knowledge statements in near-perfect accuracy.

A.3 Test-Train Overlap

Table 5 shows the 2Wiki Test-Train knowledge statement overlap, where **2Wiki (full)** corresponds to the statistics using the full training set, and **2Wiki (down-sampled)** corresponds to the down-sampled training set that we used in our actual experiment. The inference steps in 2Wiki are mostly 2 or 4, so overall there're higher chances for the coverage ratio to be 50%.

A.4 Examples of processed data samples & failure cases of iCAP

Table 7 shows examples of our processed data samples for each dataset and each sub-category, along with some failure cases of our proposed method.

A.5 Variants of Prompter Scales

While we used RoBERTa-base to instantiate the prompter in our main experiments, it is also interesting to see how the performance varies along different scales of the prompter. Towards this end, we conducted experiments on 2Wiki with two smaller scale prompters: BERT-small (28.8 million parameters) & BERT-tiny (4.4 million parameters). The intrinsic evaluation results are shown in Table 6. It can be seen that the performance grows as the prompter scale grows; in addition, BERT-small can also achieve an impressive performance (underperforming RoBERTa-base used in our main experiments by just a small gap) while BERT-tiny basically fails. This suggests that the prompter still needs to be larger than a certain scale for our method to work well.

A.6 More Examples on Prompter Attention Visualizations

Figure 4, 5, 6, 7 show additional example prompter attention visualizations in the 2Wiki dataset, each corresponding to a different reasoning type as indicated in the captions.

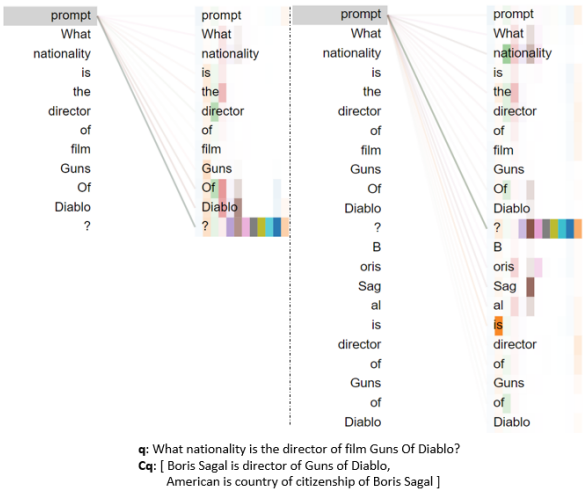

Figure 4: Prompter attention visualization. Reasoning type: Composition.

	0%	1%-20%	21%-40%	41%-60%	61%-80%	81-99%	100%
2wiki (full)	36.0%	0.0%	0.5%	28.4%	5.2%	0.0%	29.8%
2wiki (down-sampled)	71.4%	0.1%	8.1%	16.2%	2.6%	0.0%	1.6%

Table 5: Test/Train simple knowledge overlap on 2Wiki. The horizontal bar represents the percentage range of simple knowledge statements appearing in the training set, and the content values are the percentages of development & test set examples that fall into the corresponding range.

BERT-tiny				**BERT-small**			
Ent.R^*	**Ent.**R	**Ans.**$\hat{R}$	**Ans.**R	**Ent.**R^*	**Ent.**R	**Ans.**$\hat{R}$	**Ans.**R
6.0	17.7	9.0	35.3	21.4	41.2	29.1	54.2

Table 6: 2Wiki intrinsic evaluation results with two smaller-scale prompter instantiations.

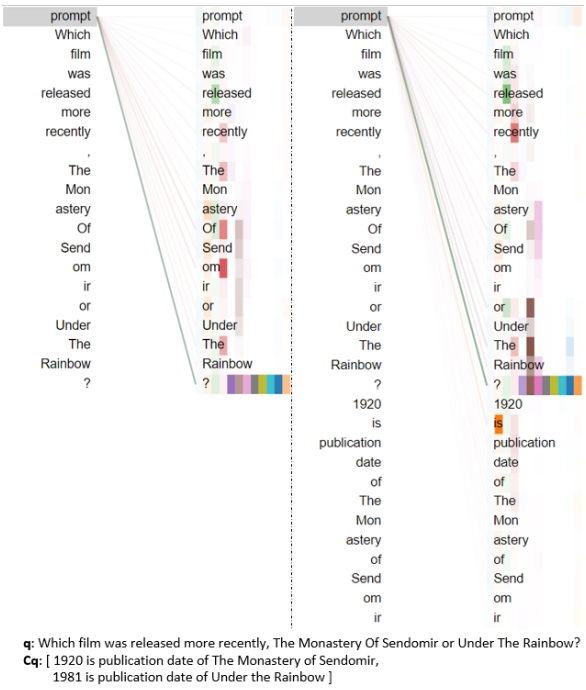

Figure 5: Prompter attention visualization. Reasoning type: Comparison.

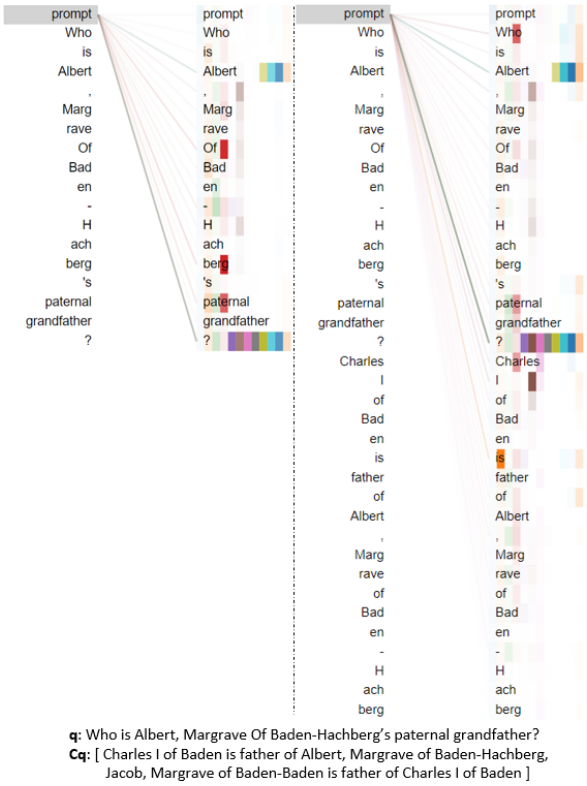

Figure 6: Prompter attention visualization. Reasoning type: Inference.

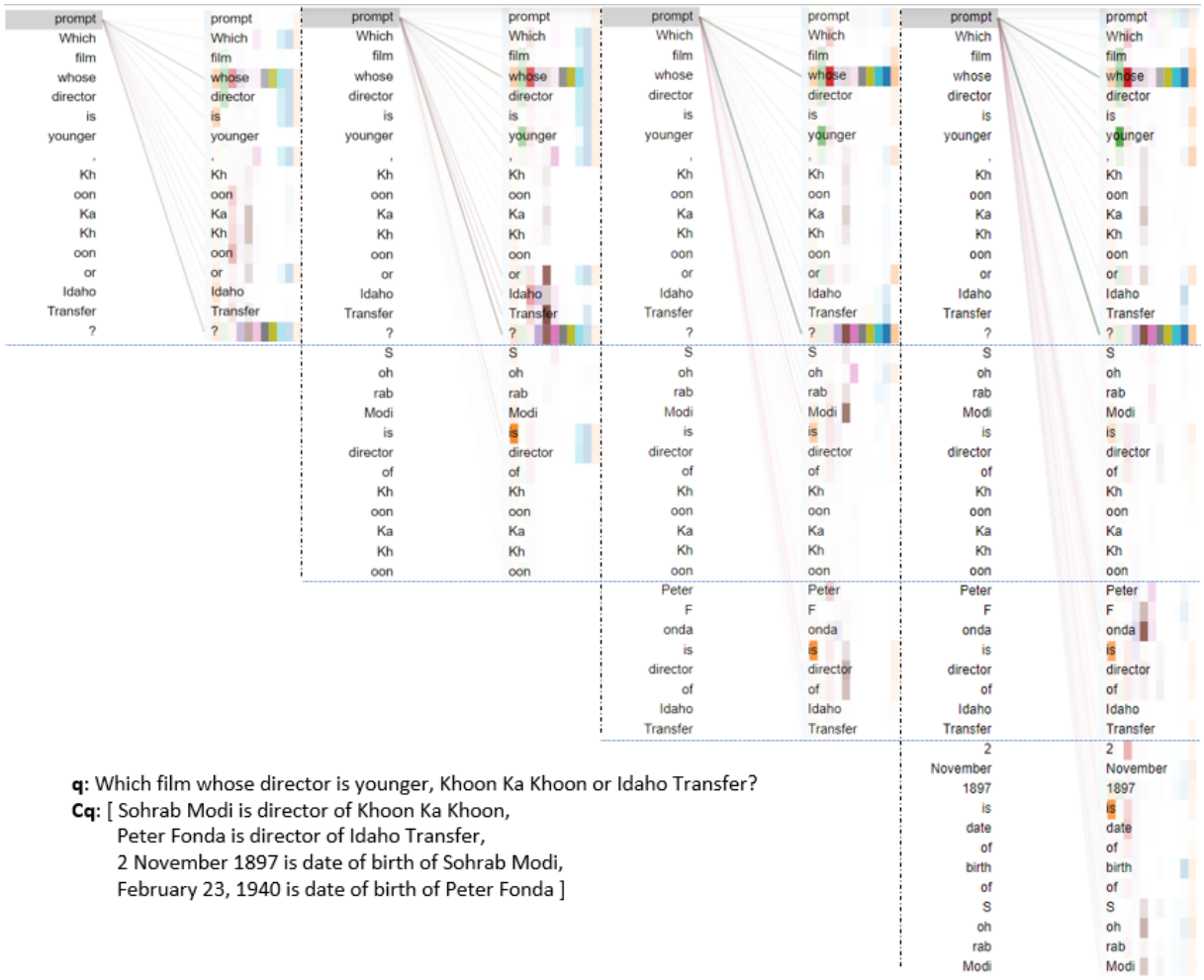

Figure 7: Prompter attention visualization. Reasoning type: Bridge-comparison.

Query (2Wiki[Composition])	What is the place of birth of the performer of song La Terre Est Ronde?
Gold Knowledge	Orelsan is performer of La terre est ronde; Alençon is place of birth of Orelsan
Recalled Knowledge	Basshunter is performer of La Terre est ronde; Havana is place of birth of Basshunter
Query (2Wiki[Comparison])	Who was born first out of Emma Kealy and Viktor Podloucký?
Gold Knowledge	29 May 1977 is date of birth of Emma Kealy; December 3, 1950 is date of birth of Viktor Podloucký
Recalled Knowledge	30 March 1977 is date of birth of Emma Kealy; 9 October 1964 is date of birth of Viktor Podloucký
Query (2Wiki[Inference])	Who is the maternal grandfather of Vyacheslav Yaroslavich?
Gold Knowledge	Ingegerd Olofsdotter of Sweden is mother of Vyacheslav Yaroslavich; Olof Skötkonung is father of Ingegerd Olofsdotter of Sweden
Recalled Knowledge	Yaroslavlava of Avidia is mother of Vyacheslav Yaroslavich; Sovatoslav is father of Yaroslavlava of Avidia
Query (2Wiki[Bridge comparison])	Which film has the director died later, One Day In The Life Of Andrei Arsenevich or Wolves Of The Range?
Gold Knowledge	Chris Marker is director of One Day in the Life of Andrei Arsenevich; Sam Newfield is director of Wolves of the Range; 29 July 2012 is date of death of Chris Marker; November 10, 1964 is date of death of Sam Newfield
Recalled Knowledge	Chris Marker is director of One Day in the Life of Andrei Arsenevich; Wallace Fox is director of Wolves of the Range; 21 January 2013 is date of death of Chris Marker; March 30, 1999 is date of death of Andrei Arsenevich
Query (LoT)	A evergreen is a important food source.
Gold Knowledge	A evergreen is a plant; A plant is not a important food source
Recalled Knowledge	A evergreen is a material, tree; A tree is a important food source
Query (R4C[Comparison])	Which documentary was filmed first, Almost Sunrise or Hail! Hail! Rock 'n' Roll?
Gold Knowledge	Almost Sunrise was filmed in 2016; Hail! Hail! Rock 'n' Roll was filmed in 1986
Recalled Knowledge	Almost Sunrise (album) is credited to American singer-songwriter Taylor Swift; Rock 'n' Roll is filmed in the 1990s
Query (R4C[Bridge])	Who was the chief executive officer of the second largest US car rental company by sales?
Gold Knowledge	The Hertz Corporation is the second-largest US car rental company; Robert L. Stone was chief executive officer of The Hertz Corporation
Recalled Knowledge	The Hertz Corporation is the second-largest US car rental company; Enterprise Rent-A-Car founder Jack Taylor was chief executive officer of Hertz

Table 7: Examples of our processed data samples for each dataset and sub-category (indicated in brackets), along with failure cases of our method.