P-ADAPTERS: ROBUSTLY EXTRACTING FACTUAL INFORMATION FROM LANGUAGE MODELS WITH DIVERSE PROMPTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work (e.g. LAMA (Petroni et al., 2019)) has found that the quality of the factual information extracted from Large Language Models (LLMs) depends on the prompts used to query them. This inconsistency is problematic because different users will query LLMs for the same information using different wording, but should receive the same, accurate responses regardless. In this work we aim to address this shortcoming by introducing *P-Adapters*: lightweight models that sit between the embedding layer and first attention layer of LLMs. They take LLM embeddings as input and output continuous prompts that are used to query the LLM. Additionally, we investigate Mixture of Experts (MoE) models that learn a set of continuous prompts (the "experts") and select one to query the LLM. They require a separate classifier trained on human-annotated data to map natural language prompts to the continuous ones. P-Adapters perform comparably to the more complex MoE models in extracting factual information from BERT and RoBERTa while eliminating the need for additional annotations. P-Adapters show between 12-26% absolute improvement in precision and 36-50% absolute improvement in consistency over a baseline of just using natural language queries alone. Finally, we investigate what makes a P-Adapter successful and conclude that access to the LLM's embeddings of the original natural language prompt, particularly the subject of the entity pair being asked about, is a significant factor.

1 INTRODUCTION

Recently, there has been an interest in meeting users' factual information needs using large language models (LLMs) as knowledge bases in place of traditional, index-based IR methods (Petroni et al., 2019; Metzler et al., 2021). LLMs do have some similarities to traditional IR methods. Both process large sets of documents: IR methods to create an index, and LLMs during pretraining. Both also can be queried by users with natural language: IR methods returning a ranked list of relevant documents leveraging their index, and LLMs returning a ranked list of vocabulary items leveraging their pretraining. However, LLM-knowledge bases have the benefit that they can interactively generate language to respond to users needs, rather than merely pointing them in to a resource.

In this conception of LLM-based knowledge bases, LLM parameters are frozen and end users can query them using their own handwritten, natural language prompts. For example, if one user wants to know what the capital of Canada is, they might query a model with the prompt, "The capital of Canada is `[MASK]`.", while another might use different wording: "Canada, which has the capital city `[MASK]`." In order for LLMs to be effective knowledge bases, they have to be robust to different queries users could provide. Unfortunately, prior work has shown that LLMs are not robust: the queries they are prompted with greatly impact the model's predictions, and semantically equivalent prompts may lead to inconsistent predictions (Jiang et al., 2020; Elazar et al., 2021). In the above example, the first prompt does extract the correct answer "Ottawa" from BERT Base, but the second one extracts the incorrect "Winnipeg". This observation has led many works to try to find the optimal prompt or set of prompts for a given relation: the one(s) that will allow models to extract factual information the best (e.g., for the relation `capital_of`). These works have proposed ensembling sets of prompts (Jiang et al., 2020), optimizing which tokens are included in the prompt (Haviv et al., 2021; Shin et al., 2020), or forgoing the difficult discrete optimization problem entirely, and instead

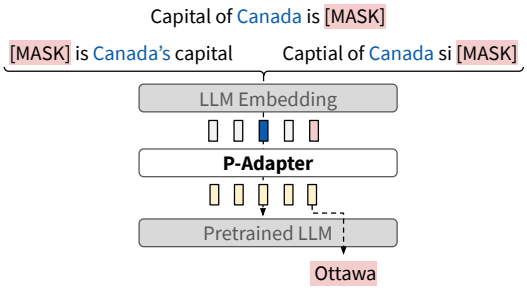


Figure 1: The P-Adapter Framework: P-Adapters sit between the embedding layer and the first attention layer of the LLM. They take as input the LLM embeddings and output continuous prompts that are fed into the LLM. The LLM is frozen (embeddings included), while the parameters of the P-Adapter are trained. The adapter helps mitigate variability among different phrasings and typographic errors in the prompts.

optimizing the continuous embeddings that are input to the LLM (Li & Liang, 2021; Qin & Eisner, 2021; Zhong et al., 2021).

In our work, rather than focus on finding a single optimal prompt, we aim to help LLMs overcome this variability by *adapting* natural language prompts into continuous representations that allow LLMs to accurately predict factual information. Because we are motivated by a user-focused setting, we want our adaptation method to require only a natural language prompt (e.g. "The capital of Canada is `[MASK]`") at inference time with no additional annotation for the relation (`capital_of`) between the subject ("Canada") and object ("Ottawa") of the entity pair in the prompt or even the identity of the subject separate from the prompt. At training time, our ideal method would only rely on (prompt, object) pairs (e.g. ("The capital of Canada is `[MASK]`", "Ottawa")), so collecting new data would not require additional annotation. We introduce a class of models called P-Adapters for performing this adaptation, which satisfy our inference and training desiderata. P-Adapters sit between the embedding layer and the first attention layer of the frozen LLM, and modify the LLM's embeddings so the factual information can be more effectively predicted (Figure 1). They are optimized end-to-end, only requiring (prompt, object) pairs at training time. We also investigate other models that could increase the consistency of natural language prompts: Mixture of Experts (MoE) models. Rather than adapt a natural language prompt, a MoE model maps it to a canonical continuous prompt (an "expert") based on the relation between the entities in the prompt, and queries the LLM with that expert. These methods require additional annotations during training and inference (namely the relation between the entities and the identity of the entity subject), and therefore do not conform to our desiderata. However, we include them in our discussion because they are useful for comparing to previous work and for understanding why P-Adapters are successful. We find that P-Adapter and MoE methods both strongly outperform a baseline of using the natural language prompts themselves by between 12-28% on precision and 36-86% on consistency. While MoE models usually perform better, the small increase comes at the cost of requiring relation annotations at training time and specifying the subject of the entity at inference time.

To simulate the variable kinds of natural language prompts we expect our methods to handle, we evaluate them in three out-of-distribution (OOD) settings: on different set of natural language prompts, on a different distribution of entities (uniform rather than the distribution of Wikidata), and on natural language prompts with typographical errors. These OOD evaluations are also important because recent work has found that models that appear to have knowledge of factual information are often relying on surface-level feature associations. For example, models ignore negation, utilize stereotypical associations when making predictions (Kassner & Schütze, 2020; Poerner et al., 2020), and pick up on the distribution of objects from their training corpus rather than learn how to access arbitrary facts (Zhong et al., 2021; Cao et al., 2021). We find that the OOD distribution of entities is the most challenging setting, and while typographic errors reduce model performance, they do so by less than might be expected.

Finally, we take a deep dive into what makes a P-Adapter effective, finding that it is important to keep some of the original natural language prompt embeddings available to the LLM. In particular,

the availability of the subject of the prompt is the most important factor, in contrast with some other work suggesting the subject matters less (Cao et al., 2021).

Overall, we conclude that our method is helpful in reducing the impact of variable prompts, and is able to do so without requiring additional annotations. We hope P-Adapters are taken up by the community as effective ways to extract factual information from LLMs.

2 Related Work

Prompting Language Models for Factual Knowledge. Prompting has taken off as a low-parameter alternative to finetuning LLMs. In prompting, the LLMs are usually frozen and their masked language model pre-training task is used to fill in the relevant information, which is usually a token (Trinh & Le, 2018; Davison et al., 2019; Petroni et al., 2019). Cao et al. (2021) outlines three main approaches for prompting LLMs for factual information. The first involves querying them zero-shot with a prompt that contains the single entity of interest (like in Figure 1) (Petroni et al., 2019). The second consists of querying them in a few-shot setting, where a few completed example prompts are provided in addition to the query of interest (e.g. "The capital of France is Paris. The capital of Canada is `[MASK]`") (Lu et al., 2021; Gao et al., 2021). The third involves querying LLMs with additional context that likely contains the answer to the query (e.g. "Ottawa sits on the Ottawa River at the border between Quebec and Ontario. The capital of Canada is `[MASK]`.") (Petroni et al., 2020; Bian et al., 2021). In this work, we focus on the first approach because it is similar to how IR systems are currently used in practice.

Much previous work in this setting endeavors to understand how much factual information LLMs store. (Petroni et al., 2019) use one natural language prompt for each relation that they evaluate in the LAMA dataset. Subsequent work argues that this approach of using a single natural language prompt per relation underestimates the ability of language models to predict factual information given language models are very sensitive to the input prompts. To this end, Jiang et al. (2020) and Elazar et al. (2021) generate paraphrases of prompts in the LAMA dataset and ensemble them to get a score for each relation. Other methods argue that natural language prompts might not be the best, and instead optimize a discrete (Shin et al., 2020; Haviv et al., 2021) or continuous (Qin & Eisner, 2021; Zhong et al., 2021; Liu et al., 2021b) prompt for each relation.

Challenges with Prompting. While the idea of querying LLMs for factual information has received recent interest, there are notable shortcomings such as source attribution, updating missing or outdated knowledge, and even the consistency of model predictions (Metzler et al., 2021). We focus on the issue posed by consistency. Elazar et al. (2021) thoroughly investigated this issue, finding models often predict different entities for semantically equivalent prompts. Further, Elazar et al. (2021) proposed to continue training LLMs with a consistency loss function to improve their robustness. In this work, however, we focus on a user-centric and lightweight approach that does not require updating LLM parameters.

Robustness and Extracting Facts. Poerner et al. (2020) and Kassner & Schütze (2020) argue that LLMs exploit surface form regularities when making predictions. Zhong et al. (2021) and Dufter et al. (2021) make a related observation, concluding that simpler models like randomly initialized LLMs, static embeddings, and even a Naive Bayes model can achieve an accuracy better than a majority baseline. Cao et al. (2021) argue that prompts, that were found to do well in previous work, overfit the distribution of objects in the training data rather than enabling knowledge extraction. They show that prompting with different sets of entities leads to very similar language model predictions. Therefore, we adopt an out-of-distribution evaluation setting to mitigate these concerns. Low parameter finetuning methods (Houlsby et al., 2019; Ben Zaken et al., 2021; Liu et al., 2021a) have also been shown to have better OOD performance for generation and classification tasks (Li & Liang, 2021; Lester et al., 2021), which we hope to leverage for factual extraction here.

3 Terminology

Taking inspiration from Liu et al. (2021a), we use the following terminology throughout this work. Each fact we want to extract consists of two parts: a pair of entities (a *subject* (x) and an *object* (y))

and the *relation* (r) between them. We use the 41 relations proposed by the LAMA dataset (Petroni et al., 2019). Each relation is expressed through a number of *templates* (e.g., one might be "The capital of [X] is [Y]."). To query a language model for factual information, we substitute the subject into its place in the template, and a `[MASK]` token into the object's place to form a *prompt* (x'). We refer to this as a *natural language prompt* to emphasize that it consists of natural language tokens. Our prediction problem then consists of mapping x' to y.

They way LLMs perform this mapping is by using their embedding layer, e, to embed x' into a sequence of continuous vectors, and then input this sequence to the first attention layer of the LLM. We refer to this sequence of continuous vectors that are input to the LLM as a *continuous prompt* (x'_{cont}). We make this distinction because our P-Adapters sit between the embedding layer and the first attention layer of the LLM, so they take as input $e(x')$ and output a continuous prompt x'_{cont} that is different from $e(x')$. After passing x'_{cont} into the rest of the LLM, we take the LLM's prediction (y') to be the token assigned the highest probability in the location of the `[MASK]` token in x', and compare it to y to assess if the model knows the fact.

4 METHODS

4.1 DATA

We use the entity pairs and relations from the T-Rex split of the LAMA work (Elsahar et al., 2018; Petroni et al., 2019) in our experiments. This data includes 41 relations and approximately 1000 entity pairs for each relation. We pool the templates from the LAMA, LPAQA, and ParaRel datasets (Jiang et al., 2020; Elazar et al., 2021). LPAQA includes templates created automatically with a paraphrase model, mined from Wikipedia, and written by annotators, and ParaRel contains high-quality human-written prompts. Additionally, we use the BERT lexical substitution system from Lee et al. (2021) to augment our pool of ParaRel templates. This gives us on average 81.4 prompt templates per relation (More statistics available in Figure 6 in the Appendix).

For training and validation, we use the entity pairs from the training and validation splits of Shin et al. (2020) respectively. We randomly divide the templates into two groups: one for training and one for OOD natural language prompt evaluation.

To test extracting arbitrary factual knowledge (rather than overfitting to training templates and entity pairs distributions), we perform evaluations in four different settings: The first setting is on ID templates and objects. We use the entity pairs from the LAMA dataset and the templates that were seen during training time. The entity pairs are different from what was seen at training-time, but they still follow the distribution of objects in Wikidata—the source of the training entity pairs.

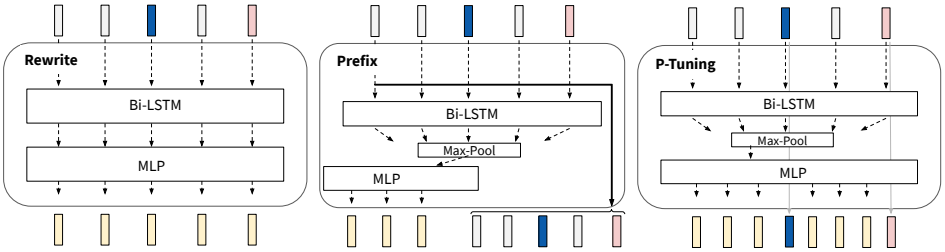
Second, we evaluate on OOD natural language prompts (**OOD Prompts**). This tests whether our P-Adapters generate effective continuous prompts for novel templates. We use the same entity pairs as in the ID evaluation, but use the templates we did not train with.

The next evaluation setting is on an OOD distribution of objects (**OOD Objects**). This tests whether our P-Adapter models learn to match the distribution of objects in the training set rather than predict arbitrary object entities. To evaluate this, we use the uniform-wikidata dataset from Cao et al. (2021). This dataset was constructed so that each object appears the same number of times for each relation, in contrast to the ID evaluation set which contains a more uneven distribution of objects from Wikidata.

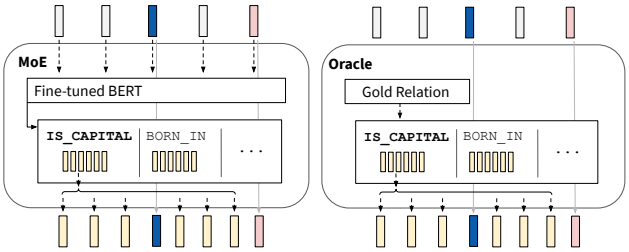
Finally, we evaluate on templates with typographic errors (**OOD Keyboard Errors**). This assesses the robustness of our P-Adapters to potential typos in the natural language prompts. We use the entity pairs and the templates from the ID evaluation, except we introduce at least one typographic error in each template using the nlpaug package (Ma, 2019).

4.2 MODELS

First we introduce our P-Adapter models, and then we introduce our MoE models, and we finish up with our oracle and baseline models. Recall that P-Adapters intercede between the LLM's embedding layer, e, and the first attention layer (Figure 1). The LLMs that we extract factual information



(a) P-Adapter Proposals



(b) Mixture of Experts and Oracle

Figure 2: P-Adapters lie in between the LLM Embeddings and the rest of the model (1). We propose end-to-end P-Adapters (2a) as well as a Mixture of Experts model (2b). Figures 2b and 2a are substituted into the "P-Adapter" block in 1.

from are BERT Base Cased, BERT Large Cased, and RoBERTa Large (Devlin et al., 2018; Liu et al., 2019).

P-Adapter. Our P-Adapter models take as input the embeddings of the natural language prompt, $e(x')$, and output a new sequence of continuous embeddings that are used as input to the LLM in place of the original natural language prompt, x'_{cont}. Formally, a P-Adapter is a function $f_{\text{prompt}} : e(x') \rightarrow x'_{\text{cont}}$ trained to maximize $P_{\text{LM}}(y \mid x'_{\text{cont}})$. The LLM's prediction would then be:

$$\arg\max_{v \in \mathcal{V}} P_{\text{LM}}\left(v \mid f_{\text{prompt}}(e(x'))\right),$$

where $\mathcal{V}$ is the LLM vocabulary. There are many different possible parameterizations of f_{prompt}, and we describe the three here. Two of these (Rewrite P-Adapter and Prefix P-Adapter) fit the training and inference data criteria put forth in the introduction. The third (P-Tuning P-Adapter) does not, but we include it because it provides insight into what makes a P-Adapter effective (See Figure 2a for an illustration of each.)

The simplest adapter we investigate is the **Rewrite P-Adapter**. The Rewrite P-Adapter is parameterized by a bidirectional LSTM followed by an MLP layer. The output of the MLP is x'_{cont}, and is input to the LLM. We use LSTMs to better compare our results with those of Liu et al. (2021b) who use also use them, and because we want to use only a small number of parameters. Additionally, the prompts tend to be short and we do not do any pre-training, so LSTMs are effective.

Next we look at the **Prefix P-Adapter**. The Prefix P-Adapter is parameterized similarly to the Rewrite P-Adapter; however, it prepends the output of its MLP to $e(x')$. Additionally, the prefix is set to be a fixed length, as is common in many works that learn a prompt prefix (Li & Liang, 2021; Lester et al., 2021). We choose a prefix length of nine to compare with later models. Formally, using the Prefix P-Adapter, the input to the LLM is $[g(e(x')); e(x')]$, where g is the application of a Bi-LSTM, a max-pool, and an MLP. Unlike the Rewrite P-Adapter, the Prefix P-Adapter keeps all of the embeddings $e(x')$ accessible to the LLM's first attention layer.

Finally, we investigate the **P-Tuning P-Adapter**. This P-Adapter is based on work by Liu et al. (2021b) in which they present a method called P-Tuning that learns a continuous prompt for each relation in the LAMA dataset. They learn a function $g : r \rightarrow \mathbb{R}^{d \times 9}$, where d is the hidden dimension of the LLM. The x'_{cont} consists of the output of g, the embedding of the subject, and the embedding

of the mask token. It has the following form:

$$\left[g(r)_{[0:3]}; e(x); g(r)_{[3:6]}; e(\texttt{[MASK]}); g(r)_{[6:9]}\right],$$

where bracket notation represents python-style indexing.

Our P-Tuning P-Adapter is implemented in a similar fashion, except our g takes as input x', and is parameterized similarly to the Prefix P-Adapter. Like the Prefix P-Adapter, a fixed number of embeddings are output; however, unlike the Prefix P-Adapter, only the embeddings for the subject and `[MASK]` token are kept unchanged, rather than all of $e(x')$. Note that this P-Adapter uses more annotations during training and inference than the other two: specifically, it requires knowing the identity of the subject. This means that it does not fit our desiderata laid out in the introduction, but we include because it allows us to investigate what makes a P-Adapter successful.

All of these adapters were trained using the hyperparameters from Liu et al. (2021b): Adam optimizer with a learning rater of 0.00001, weight decay of 0.0005, a batch size of 128, and an exponential learning rate decay schedule with a decay rate of 0.98 (Kingma & Ba, 2015).

Enforcing consistency does not require modifying a natural language prompt like the P-Adapters do. Another option is to map each prompt to a canonical prompt, and query the LLM with this prompt. We explore this next with our MoE and Oracle methods. In these, the canonical prompt is a continuous prompt for the relation between the entities in the natural prompt learned with P-Tuning. In contrast to the P-Adapter methods, these require additional annotations with the relation of the prompt (See Figure 2b).

Mixture of Experts. Our Mixture of Experts (MoE) model consists of two components. The first is a classifier that predicts the relation between the entities of a natural language prompt. This classifier is a BERT Base Cased model fully fine-tuned on a 41-way classification task using an AdamW optimizer with a learning rate of 0.001 and linear learning rate decay (Loshchilov & Hutter, 2018). We use HuggingFace Transformers to train the model for 3 epochs on the same training data used to train the P-Adapter models (Wolf et al., 2020).

The second component is a look-up table to map the predicted relations to the cannonical continuous prompts. We train these using the same hyperparamters reported in Liu et al. (2021b) with one prompt for each of the 41 relations.

Note that this is similar to a traditional mixture of experts model approach except that we do not use a weighted combination of prompts from different relations and instead just use the single prompt from the predicted relation.

Oracle. Our oracle method is similar to the MoE approach, except rather than using a classifier to predict the relation, we use the gold relation at test time. We call this method the oracle because it achieves perfect consistency (not precision). The oracle also serves as a direct comparison to P-Tuning because P-Tuning assumes access to the gold relation.

Baseline. Our baseline is a model that takes as input the natural language prompt without any prefixes or optimization. Any effective and useful P-Adapter network would have to perform better than just using the natural language text itself.

4.3 METRICS.

We report two metrics: precision@1 (P@1) and consistency. P@1 is a stringent measure of whether a prompt can extract a fact. It is defined as the proportion of prompts where the correct object is the LLM's top prediction. Consistency measures whether a model's predictions for a given entity pair match across the different prompts x''s that contain the entities. We calculated consistency using the method from Elazar et al. (2021): the consistency of knowledge of a fact is the proportion of pairs of prompts for the fact where the model makes the same prediction. Formally, given a set of unordered pairs of prompts, X', where there are n unique prompts, and letting $v(x') = \arg\max_{v \in \mathcal{V}} P_{\text{LM}}\left(v \mid f_{\text{prompt}}(e(x'))\right)$ return the model's top prediction given a natural language prompt x', the consistency is defined as:

$$\text{consistency} = \frac{\sum_{(x'_i, x'_j) \in X'} \mathbb{1}[v(x'_i) = v(x'_j)]}{\frac{1}{2}n(n-1)}.$$

Note a model can be consistent but is not accurate (e.g., by always predicting the same incorrect object).

5 RESULTS

Metric	P-Adapter	ID	OOD Prompts	OOD Objects	OOD KE
P@1	Baseline	0.157	0.157	0.069	0.092
	Rewrite	0.258 $\pm$ 0.00	0.247 $\pm$ 0.00	0.078 $\pm$ 0.00	0.167 $\pm$ 0.00
	Prefix	**0.416** $\pm$ 0.01	**0.407** $\pm$ 0.01	**0.192** $\pm$ 0.00	**0.320** $\pm$ 0.01
	P-Tuning	0.442 $\pm$ 0.00	0.422 $\pm$ 0.00	0.203 $\pm$ 0.00	0.325 $\pm$ 0.00
	MoE	0.486 $\pm$ 0.00	0.418 $\pm$ 0.00	0.233 $\pm$ 0.01	0.330 $\pm$ 0.00
	Oracle	0.496 $\pm$ 0.01	0.496 $\pm$ 0.01	0.238 $\pm$ 0.02	0.496 $\pm$ 0.01
Consistency	Baseline	0.126	0.133	0.097	0.068
	Rewrite	0.476 $\pm$ 0.01	0.448 $\pm$ 0.02	0.456 $\pm$ 0.01	0.223 $\pm$ 0.01
	Prefix	**0.629** $\pm$ 0.02	**0.591** $\pm$ 0.01	**0.572** $\pm$ 0.02	**0.437** $\pm$ 0.02
	P-Tuning	0.730 $\pm$ 0.00	0.646 $\pm$ 0.01	0.656 $\pm$ 0.01	0.476 $\pm$ 0.01
	MoE	0.953 $\pm$ 0.02	0.664 $\pm$ 0.02	0.925 $\pm$ 0.03	0.441 $\pm$ 0.01
	Oracle	1.000 $\pm$ 0.00	1.000 $\pm$ 0.00	1.000 $\pm$ 0.00	1.000 $\pm$ 0.00

Table 1: P@1 and Consistency for BERT Base across all our settings. P-Adapters are separated based on whether they require additional training/inference data. Note that the Baseline does not have any standard deviation because there is no optimization across runs, and that the consistency of the Oracle is 1.0. (OOD KE: OOD Keyboard Errors)

We present our main results for BERT Base in Table 1. The results for BERT Large and RoBERTA Large show similar trends and are available in the Appendix (In Tables 2, 3 as well as Figures 4 and 5). Comparing the different evaluation sets, we observe the following. First, the OOD Objects was the most challenging setting, on average 20.41% lower precision than the ID setting, even for the oracle. However, the models performed similarly on the OOD Prompts as they did on the in domain ones. At first this might seem to conflict with previous work that finds that the prompt has a large impact on performance (Jiang et al., 2020), but these results make claims about average prompt performance rather than individual prompt performance. That said, precision is still higher ID than OOD, particularly for the MoE model. While this suggests some overfitting, our methods still outperform the baseline, so what they learn is still useful for the OOD natural language prompts. Finally, when evaluated on the OOD Keyboard Errors, we find that the models still do well compared to the baseline. However, precision on the corrupted prompts is lower than the uncorrupted ones.

Across all evaluation sets, we find that optimized prompts lead to higher precision than natural language ones, as the low performance of the baseline indicates.

Additionally, across all variation types we can see that optimized prompts lead to more consistency among the models' predictions. The consistencies dramatically increase from less than 0.2 for the baseline to over 0.4 (Figure 5). Models were the least consistent on the OOD Keyboard Errors, and interestingly, were similarly consistent between the OOD Prompts and OOD Objects evaluations despite having a much higher precision for the OOD Prompts. The oracle has a perfect consistency of 1.0, because it uses the same continuous prompt for all facts with the same relation, so the predictions for that fact are all the same. The MoE model has a high consistency for a similar reason: as long as the classifier predicts two prompts for the same entity pair come from the same relation, the predictions will be the same, and therefore consistent. The P-Tuning P-Adapter often has a a high consistency as well, sometimes even higher than the MoE. This suggests that in P-Tuning P-Adapter, the main factor driving the model's prediction is the subject of the entity pair, because the unmodified embedding of the subject is shared among all of the prompts for the same object.

This last points contributes to a more general observation: we found that giving the LLM access to the embeddings of the natural language prompt was helpful. The Prefix P-Adapter and the P-Tuning P-Adapter, which have access to some unmodified LLM embeddings, achieve higher precisions

than the Rewrite P-Adapter, which alters all of the LLM embeddings. Because of this, we perform additional experiments to see what part of natural language prompt was important to keep.

One hypothesis is that the [MASK] token embedding has to remain unmodified, and the Rewrite P-Adapter performs poorly because it corrupts this embedding. We investigate this in two ways. First, we train a version of the Rewrite P-Adapter that replaces the P-Adapter's output in the [MASK] token position with the LLM's [MASK] token embedding. Second, we train a version of the P-Tuning P-Adapter whose output only has the LLM's [MASK] embedding, not the subject embedding. However, we find these do not increase performance much compared to the Rewrite P-Adapter.

Another hypothesis is that the subject embeddings must remain unmodified, and that the Rewrite P-Adapter corrupts the subject, leading to lower precision. To address this, we train a version of the Rewrite P-Adapter model that replaces the embeddings for the subject in the P-Adapter output with the LLM's embeddings for the subject. We find this is much more effective, increasing precision by 13%. For comparison with the P-Tuning P-Adapter, which keeps the subject and [MASK] embeddings, we also train a version of the Rewrite P-Adapter that replaces the embeddings at both the subject and [MASK] token positions with the LLM's embeddings and find that this performs only 4% worse than the P-Tuning P-Adapter. The results are visible in Figure 3. From these experiments, we conclude that the most important aspect of the prompt is the subject token. This also helps explain why the Prefix P-Adapter does well: it has access to the LLM's embeddings for the entire natural language prompt, which includes the subject.

Unfortunately, while these changes to the Rewrite P-Adapter models show positive results, they require knowing the index of the subject tokens , which contradicts our requirement to not use extra annotations. Fortunately, there is another way to incorporate the unmodified LLM's embeddings of the natural language prompt into the output of the P-Adapter: we can interpolate between the P-Adapter output and the unmodified embeddings. To do this, we train a third type of Rewrite P-Adapter. If $f_{\text{rewrite-prompt}}$ is the original Rewrite P-Adapter, the equation for the new Rewrite P-Adapter is:

$$f_{\text{prompt}}(x') = \alpha e(x') + (1 - \alpha) f_{\text{rewrite-prompt}}(e(x')).$$

where α is a hyperparameter. When $\alpha = 0$, this P-Adapter is equivalent to the original Rewrite P-Adapter, and when $\alpha = 1$ it is equivalent to the baseline. We test $\alpha \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$, and find $\alpha = 0.5$ to perform the best. It outperforms the Rewrite P-Adapter when the subject and [MASK] tokens are substituted in, though barely underperforms compared to the P-Tuning P-Adapter (See Figure 3).

6 Discussion

Our goal was to create models to adapt arbitrary user queries so predictions from frozen LLMs were accurate and consistent. One of our desiderata was to train with only (prompt, object) pairs, so that a user's experience would match current IR system experiences. Despite this requirement, some of the methods we introduce do use additional information—for comparison to other methods and understanding our own. For example, the P-Tuning P-Adapter and the MoE models both need the identity of the subject of the user's prompt, and requiring a user to provide this is feels cumbersome and redundant. One could obviate this need using an off-the shelf NER system as well, and we leave this to future work.

Additionally, the MoE models train their classifiers using annotations for the relation between the entities in the prompts. We use these annotations to compare to previous work: previous work assumes that relation is the most useful intermediate variable for factual extraction, but this does not have to be the case. For example, good prompts across different relations like "Ottawa is the capital of [MASK]" and "The second largest country by land area is [MASK]" might share components even though they do not share a relation. Our P-Adapters allow for greater flexibility by conditioning on the natural language prompt itself rather than the relation. Importantly, while using extra information led to better results in some cases, the Prefix P-Adapter was usually able to achieve comparable precision. And while the Rewrite P-Adapter floundered, ensembling its outputs with the LLM embeddings led to greater success.

There are two likely reasons for the P-Adapters' success. First, not updating the LLM parameters likely maintains all of factual knowledge learned during pretraining. Second, the natural language

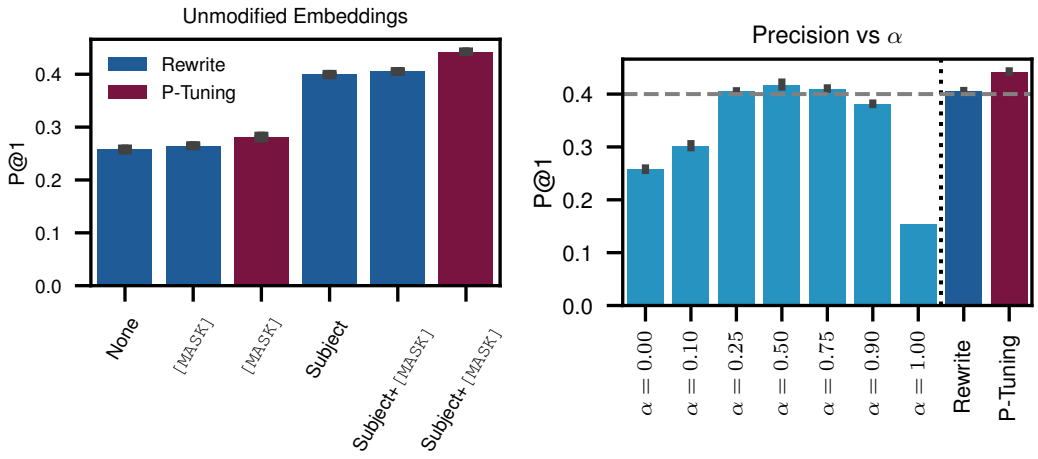


Figure 3: The above results show which of the LLM's embeddings are important to keep unmodified. On the left, the x-axis has the embeddings that are not modified by the P-Adapter (e.g. "None" is the Rewrite P-Adapter, where all embeddings are modified). These results are computed with BERT Base and the ID data. In the plot on the right, we compare different α's to the P-Tuning P-Adapter and the version of the "Rewrite" P-Adapter with the unmodified subject and `[MASK]` embeddings, finding the $\alpha = 0.5$ performs as well as the other adapters that require additional annotation.

prompts are likely close to being effective, so they do not require large changes to elicit the correct objects. The small pushes from the P-Adapters are sufficient.

It is also worth noting that while the Prefix P-Adapter was able to approach the performance of the techniques with more information, this performance is still quite modest. The oracle was able to achieve only a precision of 50% in the cases with ID objects, missing half of the facts. For the OOD Objects evaluation set, it performed much worse, only correctly predicting about 25% of the facts. While the P@1 metric is quite strict, these numbers do not bode well for this approach. Cao et al. (2021) blame this poor performance on the optimized prompts overfitting to the training object distribution, and they observe that when the subjects are changed, the LLM's distributions do not change all that much. They even observe that the LLM's distributions when prompts have subjects are similar to those when the subjects are completely omitted. Our findings point to a more nuanced result. While it might be true that the LLM's distributions in these cases are correlated, the subject does matter when considering the top prediction. We find that without the LLM's embeddings of the subject accessible, the P-Adapters performs even more poorly.

7 Conclusion

LLMs implicitly learn factual information during pretraining, but accurately and consistently extracting this information presents a challenge. Our work complements previous work on extracting factual information from LLMs by introducing a more user-oriented setting: we want our models to only take in varied natural language prompts, and return the objects that meet information needs. To do this, we propose P-Adapter models, which sit between the embedding layer and the first attention layer of LLMs. These are effective compared to other methods that require additional annotations (e.g., the relation between the entities in the prompt), and we perform ablations determining that their success is due to keeping the LLM's embeddings of the subject available to the LLM. While we focus on the task of extracting factual information from LLMs, P-Adapters potentially provide a general framework for adapting to variable inputs, for example in reading comprehension or dialogue. While there is still room for the further improvement of using LLM's as knowledge bases, and we see P-Adapters as an important part of the future of this endeavor.

8 ETHICS STATEMENT

While our work is inspired by how users interact with current IR systems, we do not perform any experiments with human subjects. Additionally, our method improves the ability to extract factual information from Pretraining corpora, so this means it may be possible to exploit our method to elicit the biased or harmful from these corpora as well, though we do not study this. Finally, improving the ability to accurately and consistently extract information from text sources has privacy concerns if LLMs are pretrained on private or otherwise sensitive data. For the LLMs we study, this is not a concern as they were trained on Wikipedia and Books Corpus data rather than Web data.

9 REPRODUCIBILITY STATEMENT

The descriptions of our models and training procedure in Section 4 and the code uploaded in the supplementary material is sufficient for reproducing our results. The data we use is available in the Github repositories of the work we cite that creates these resources. For running experiments, see the README.md file. We will upload our code to Github and place a link in this section upon acceptance.

REFERENCES

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. arXiv e-prints, pp. arXiv–2106, 2021.

Ning Bian, Xianpei Han, Bo Chen, and Le Sun. Benchmarking knowledge-enhanced commonsense question answering via knowledge-to-text transformation. Proceedings of the AAAI Conference on Artificial Intelligence, 35(14):12574–12582, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17490.

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. Knowledgeable or educated guess? revisiting language models as knowledge bases. 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), pp. 1860–1874, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.146. URL https://aclanthology.org/2021.acl-long.146.

Joe Davison, Joshua Feldman, and Alexander M Rush. Commonsense knowledge mining from pretrained models. 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), pp. 1173–1178, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

Philipp Dufter, Nora Kassner, and Hinrich Schütze. Static embeddings as efficient knowledge bases? In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 2353–2363, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.186. URL https://aclanthology.org/2021.naacl-main.186.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. arXiv preprint arXiv:2102.01017, 2021.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-REx: A large scale alignment of natural language with knowledge base triples. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1544.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 3816–3830, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.295. URL https://aclanthology.org/2021.acl-long.295.

Adi Haviv, Jonathan Berant, and Amir Globerson. BERTese: Learning to speak to BERT. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 3618–3623, Online, April 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.eacl-main.316.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How Can We Know What Language Models Know? Transactions of the Association for Computational Linguistics, 8:423–438, 07 2020. ISSN 2307-387X. doi: 10.1162/tacl_a_00324. URL https://doi.org/10.1162/tacl_a_00324.

Nora Kassner and Hinrich Schütze. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 7811–7818, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.698. URL https://aclanthology.org/2020.acl-main.698.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR (Poster), 2015. URL http://arxiv.org/abs/1412.6980.

Mina Lee, Chris Donahue, Robin Jia, Alexander Iyabor, and Percy Liang. Swords: A benchmark for lexical substitution with improved data coverage and quality. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4362–4379, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.345. URL https://aclanthology.org/2021.naacl-main.345.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. ArXiv, abs/2104.08691, 2021.

Xiang Lisa Li and Percy Liang. 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), pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353.

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, 2021a.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. arXiv preprint arXiv:2103.10385, 2021b.

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. arXiv preprint arXiv:1907.11692, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In International Conference on Learning Representations, 2018.

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. arXiv preprint arXiv:2104.08786, 2021.

Edward Ma. Nlp augmentation. https://github.com/makcedward/nlpaug, 2019.

Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. Rethinking search: making domain experts out of dilettantes. In ACM SIGIR Forum, volume 55, pp. 1–27. ACM New York, NY, USA, 2021.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 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), pp. 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL https://aclanthology.org/D19-1250.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. How context affects language models' factual predictions. In Automated Knowledge Base Construction, 2020. URL https://openreview.net/forum?id=025X0zPfn.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. E-BERT: Efficient-yet-effective entity embeddings for BERT. In Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 803–818, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.71. URL https://aclanthology.org/2020.findings-emnlp.71.

Guanghui Qin and Jason Eisner. 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, pp. 5203–5212, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.410. URL https://aclanthology.org/2021.naacl-main.410.

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 the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 4222–4235, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.346. URL https://aclanthology.org/2020.emnlp-main.346.

Trieu H Trinh and Quoc V Le. A simple method for commonsense reasoning. arXiv preprint arXiv:1806.02847, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 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, pp. 5017–5033, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.398. URL https://aclanthology.org/2021.naacl-main.398.

A All Results

B Templates

LLM	P-Adapter	ID	OOD Prompts	OOD Objects	OOD KE
BERT Base	Baseline	0.157	0.157	0.069	0.092
	Rewrite	0.258 ± 0.00	0.247 ± 0.00	0.078 ± 0.00	0.167 ± 0.00
	P-Tuning	0.442 ± 0.00	0.422 ± 0.00	0.203 ± 0.00	0.325 ± 0.00
	Prefix	0.416 ± 0.01	0.407 ± 0.01	0.192 ± 0.00	0.320 ± 0.01
	MoE	0.486 ± 0.00	0.418 ± 0.00	0.233 ± 0.01	0.330 ± 0.00
	Oracle	0.496 ± 0.01	0.496 ± 0.01	0.238 ± 0.02	0.496 ± 0.01
BERT Large	Baseline	0.163	0.160	0.077	0.099
	Rewrite	0.117 ± 0.06	0.115 ± 0.06	0.042 ± 0.02	0.072 ± 0.03
	P-Tuning	0.470 ± 0.00	0.443 ± 0.00	0.222 ± 0.00	0.346 ± 0.01
	Prefix	0.394 ± 0.01	0.398 ± 0.01	0.194 ± 0.01	0.315 ± 0.01
	MoE	0.498 ± 0.01	0.431 ± 0.01	0.250 ± 0.02	0.339 ± 0.01
	Oracle	0.510 ± 0.02	0.509 ± 0.02	0.258 ± 0.03	0.510 ± 0.02
RoBERTa Large	Baseline	0.122	0.121	0.054	0.071
	Rewrite	0.222 ± 0.04	0.214 ± 0.04	0.068 ± 0.01	0.148 ± 0.03
	P-Tuning	0.360 ± 0.07	0.359 ± 0.06	0.167 ± 0.03	0.306 ± 0.03
	Prefix	0.338 ± 0.02	0.343 ± 0.01	0.157 ± 0.01	0.280 ± 0.01
	MoE	0.335 ± 0.09	0.290 ± 0.07	0.156 ± 0.05	0.231 ± 0.05
	Oracle	0.340 ± 0.11	0.340 ± 0.11	0.157 ± 0.06	0.340 ± 0.11

Table 2: Tabular format for the P@1 from Figure 4. The Baseline does not have any standard deviation because there is no randomness across runs. (OOD KE: OOD Keyboard Errors)

LLM	P-Adapter	ID	OOD Prompts	OOD Objects	OOD KE
BERT Base	Baseline	0.126	0.133	0.097	0.068
	Rewrite	0.476 ± 0.01	0.448 ± 0.02	0.456 ± 0.01	0.223 ± 0.01
	P-Tuning	0.730 ± 0.00	0.646 ± 0.01	0.656 ± 0.01	0.476 ± 0.01
	Prefix	0.629 ± 0.02	0.591 ± 0.01	0.572 ± 0.02	0.437 ± 0.02
	MoE	0.953 ± 0.02	0.664 ± 0.02	0.925 ± 0.03	0.441 ± 0.01
	Oracle	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00
BERT Large	Baseline	0.122	0.123	0.094	0.069
	Rewrite	0.550 ± 0.22	0.538 ± 0.23	0.571 ± 0.22	0.399 ± 0.29
	P-Tuning	0.759 ± 0.01	0.656 ± 0.01	0.694 ± 0.02	0.484 ± 0.02
	Prefix	0.542 ± 0.03	0.538 ± 0.02	0.495 ± 0.03	0.435 ± 0.01
	MoE	0.955 ± 0.02	0.680 ± 0.04	0.933 ± 0.04	0.456 ± 0.02
	Oracle	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00
RoBERTa Large	Baseline	0.093	0.096	0.076	0.050
	Rewrite	0.438 ± 0.03	0.403 ± 0.02	0.421 ± 0.02	0.217 ± 0.01
	P-Tuning	0.731 ± 0.13	0.707 ± 0.15	0.702 ± 0.18	0.600 ± 0.21
	Prefix	0.445 ± 0.03	0.450 ± 0.02	0.387 ± 0.04	0.374 ± 0.02
	MoE	0.974 ± 0.01	0.661 ± 0.03	0.970 ± 0.01	0.436 ± 0.02
	Oracle	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00	1.000 ± 0.00

Table 3: Tabular format for the consistency from Figure 5. The Baseline does not have any standard deviation because there is no randomness across runs. Note that the Oracle models achieves a consistency of 1. (OOD KE: OOD Keyboard Errors)

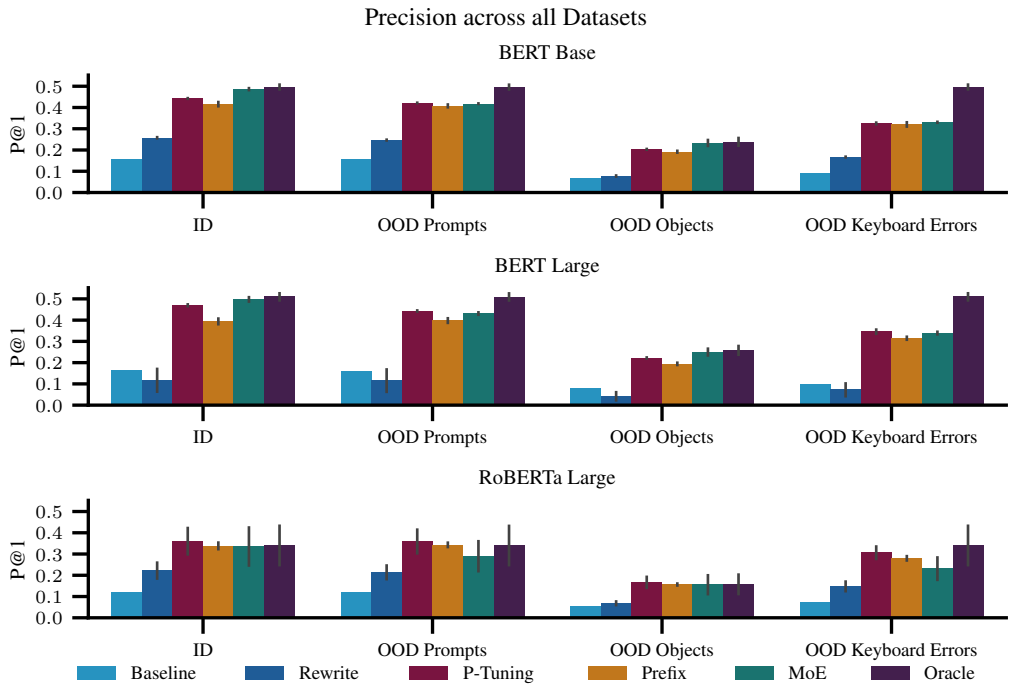


Figure 4: Results across our ID and OOD datasets across three LLMs. Each row is a different model, the x-axis is the dataset and error bars show standard deviation across either 3 runs. We can see that the oracle method outperformed all others, while the baseline and rewriter did the worst.

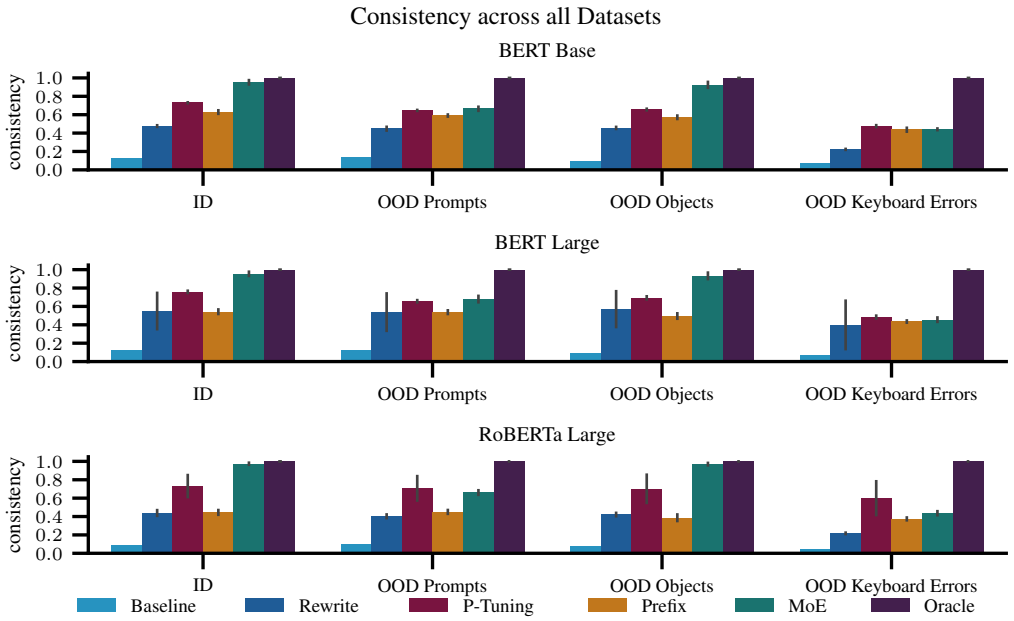


Figure 5: Consistency across all LLMs and evaluation sets.

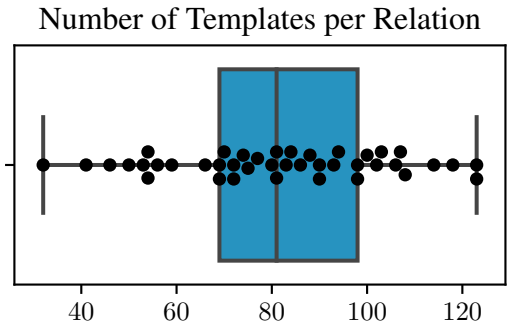


Figure 6: Distribution of number of templates across the 41 relations we tested. Minimum is 32, maximum is 123 and mean is 81.44.

subject	Canada
object	Ottawa
relation	`IS_CAPITAL`
template	The capital of [X] is [Y].
natural language prompt	The capital of Canada is `[MASK]`.
prediction	Toronto

Table 4: Here the terminology is applied to the example from the introduction. Note that the model is incorrect in this example.