ORCA: INTERPRETING PROMPTED LANGUAGE MODELS VIA LOCATING SUPPORTING EVIDENCE IN THE OCEAN OF PRETRAINING DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Prompting large pretrained language models leads to strong performance in a variety of downstream tasks. However, it is still unclear *from where* the model learns task-specific knowledge, especially in zero-shot setups. In this work, we propose a novel method ORCA to identify evidence of the model's task-specific competence in prompt-based learning. Through an instance attribution approach to model interpretability, by iteratively using gradient information related to the downstream task, ORCA locates a very small subset of pretraining data that directly supports the model's predictions in a given task; we call this subset *supporting data evidence*. We show that supporting data evidence offers new insights about the prompted language models. For example, in the tasks of sentiment analysis and textual entailment, BERT shows a substantial reliance on BookCorpus—the smaller corpus of BERT's two pretraining corpora—as well as on pretraining examples that mask out synonyms to the task labels used in prompts.[1]

1 INTRODUCTION

Large language models (LLMs) are trained on massive text corpora from the web, referred to as the pretraining data (e.g., Devlin et al., 2019; Raffel et al., 2020). Due to their volume, pretraining data typically cannot be inspected manually and are prone to spelling/logic errors, domain mismatch w.r.t. target tasks, social biases, and other unexpected artifacts (Bender et al., 2021). Yet, LLMs pretrained with such noisy data attain surprisingly good performance on numerous downstream tasks, with little or no task-specific tuning (Petroni et al., 2019; Brown et al., 2020).

There are several hypotheses explaining the power of pretrained LLMs. One hypothesis is that the pretraining data is huge and the model *might* be shallowly memorizing patterns in data (Bender et al., 2021; Carlini et al., 2021; Razeghi et al., 2022). An alternative hypothesis is that LLMs *might* be learning to reason through observed patterns in the pretraining data in novel ways (McCoy et al., 2021). However, the *evidence* of these conjectures, especially in arbitrary downstream tasks, remains underexplored.

Such evidence is useful as it can help explain model decisions, surface problematic patterns in data or model behavior, and shed new light on how to improve the model and data (Zhong et al., 2019; Han & Tsvetkov, 2020; 2021; Pruthi et al., 2022). Moreover, it can facilitate the trustworthiness of the models (Lipton, 2018; Jacovi et al., 2021).

In this work, we develop a methodology to provide such evidence. Our hypothesis is that among the enormous pretraining corpora, there is a subset of pretraining data that contributes to the model's behavior on a downstream task more than the rest of the pretraining data. Therefore, our task is to locate a task-specific evidence set—a very small amount of pretraining data that particularly helps the model's performance on the task. We call it *supporting data evidence* (SDE). Such SDE can help interpret the model if we analyze its task-relevant patterns compared to the rest of the corpora.

A related line of interpretability research focuses on instance attribution (Koh & Liang, 2017; Yeh et al., 2018; Pruthi et al., 2020; Han et al., 2020), where the goal is to find which training examples

[1]Code and data will be released at `ANONYMIZED` upon publication.

are most influential to the model's decision, focusing on individual test examples. However, in this work we are interested in locating sets of pretraining data influencing the *whole task* (i.e., a full test set, rather than individual test instances). We seek such "global" evidence for the task because given the scale of the pretraining and task data, it could be inefficient or even infeasible to find and inspect the evidence for each of the task examples.[2]

We first formulate the problem of finding SDE in pretraining data by upweighting the SDE set and measuring its impact on model performance (§2.1). In §2.2, we propose a novel method ORCA[3] that effectively identifies the SDE by iteratively using task-specific gradient information. On two classification tasks—sentiment analysis and textual entailment—in a prompt-based setup (§3), we show the effectiveness of the SDE discovered by ORCA compared to random data subsets and nearest-neighbor data in an embedding space (§4). Our analyses of the discovered SDE (§5) show that our base language model BERT (Devlin et al., 2019) has an interestingly high reliance on the smaller corpus of its two pretraining corpora (BookCorpus, Zhu et al., 2015). Also the pretraining examples in SDE typically mask out synonyms to the task verbalizers (i.e., words mapped to the task labels in the prompts, Schick & Schütze, 2021).

2 ORCA

We develop a method to explain the competence of large pretrained language models used in zero- or few-shot prompt-based classification (Petroni et al., 2019; Brown et al., 2020).[4] Without conventional finetuning, model decisions rely on knowledge learned from the pretraining data, and our goal is to identify what supporting data evidence (SDE) in pretraining data facilitates model's competence in a specific downstream task.

2.1 Problem formulation

Assume θ^{PT}, a LLM pretrained with a dataset $D^{\text{PT}} \ni (x^{\text{PT}}_{\text{context}}, y^{\text{PT}}_{\text{masked}})$. For example, for a masked language model $x^{\text{PT}}_{\text{context}}$ is a block of text with certain tokens masked, and $y^{\text{PT}}_{\text{masked}}$ are the masked tokens in their original forms, to be reconstructed. θ^{PT} is trained to minimize a loss $\mathcal{L}$ over the pretraining examples, $\theta^{\text{PT}} = \arg\min_\theta \mathcal{L}(D^{\text{PT}}; \theta)$.

The LLM can be applied to many downstream tasks without finetuning, via prompting (Schick & Schütze, 2021; Liu et al., 2021). Given a dataset in a downstream classification task $D^{\text{task}} \ni (x^{\text{task}}, y^{\text{task}})$, the LLM is applied by measuring $p_\theta(\text{verbalizer}(y^{\text{task}}) \mid \text{template}(x^{\text{task}}))$. The template supplies a prompt tailored to the task for the model, and the verbalizer maps the output of the language model to the task's label space (more details in §3.2).

We interpret the model decisions by finding the SDE $S \subset D^{\text{PT}}$ w.r.t. the task data D^{task}. The size of S should be very small (e.g., a few hundred) compared to the whole pretraining data, $|S| \ll |D^{\text{PT}}|$, to facilitate further manual or semi-automatic analyses. More importantly, S should "contribute" significantly to the performance of the model on the downstream task.

However, we first observe that defining this contribution is a non-trivial problem. Prior work in instance attribution like influence functions (Koh & Liang, 2017) adopts a "leave-one-out" perspective (Cook, 1977). In our case, this would mean removing S from D^{PT}, retraining a new LLM from scratch, and testing it on D^{task}. This is prohibitively expensive.[5]

We adopt an "upweighting" perspective. Instead of leave-one-out, we upweight certain pretraining examples (e.g., S) by training the model on these examples for an additional epoch. The resulting change to the model should be small to prevent overfitting. Specifically, we randomly batch the SDE S to mini-batches, thereby updating the model via a very small number of optimizer updates:

$$\theta^{\text{PT}}_{\text{new}} \leftarrow \theta^{\text{PT}} + \textit{updates}_{\theta,\mathcal{L}}(S) \tag{1}$$

[2]Directly applying instance attribution methods to the task level has also been shown to yield negative results (Kocijan & Bowman, 2020).

[3]Named after the marine mammal for n**O** pa**R**ti**C**ular re**A**son.

[4]While in this work we focus on text classification, the framework is also adaptable to generation problems.

[5]Moreover, the definition of influence functions and even leave-one-out can sometimes be arguable, especially in non-convex models (Basu et al., 2021; K & Søgaard, 2021).

For the simplicity of notation, we fold a sequence of optimizer updates into $\textit{updates}_{\theta,\mathcal{L}}$ that depends on a sequence of batched data S, the model parameters θ, the loss function $\mathcal{L}$, and an optimization algorithm used during pretraining.

The quality of the data evidence is reflected in the *performance difference* in the downstream task between the new model $\theta^{\text{PT}}_{\text{new}}$ and the original pretrained model θ^{PT}, measured by the metric associated with the task.[6]

2.2 IDENTIFYING SUPPORTING DATA EVIDENCE

We now propose a novel method ORCA identifying the supporting data evidence $S \subset D^{\text{PT}}$. The goal is to find a subset of the pretraining data $|S| \ll |D^{\text{PT}}|$ that is directly helpful to the downstream task when we continue pretraining the language model over it (as described in §2.1). The intuitions behind our method are simple: (1) We aim to find contributive examples in D^{PT} that exert a similar change to the model parameters as D^{task} would. (2) There could be multiple subsets of pretraining data that, in conjunction, are useful to the task. We thus select S in several iterations rather than at once.

We build our SDE S in m iterations, $S_1, S_2, \ldots, S_m$; the size of the subset at each iteration is $\frac{|S|}{m}$. To find the first evidence subset S_1, we rely on the intuition that continuing training on the task data D^{task} directly is likely to improve the original model θ^{PT} on the task. We thus batch the task data and calculate a batch gradient $\nabla_\theta \mathcal{L}_{\text{task}}(D^{\text{task}}; \theta^{\text{PT}})$.[7] Descending along the gradient direction should improve θ^{PT}, and we find a subset S_1 of the pretraining data that exerts a similar gradient of the model as $\nabla_\theta \mathcal{L}_{\text{task}}(D^{\text{task}}; \theta^{\text{PT}})$:

$$S_1 = \{d \in D^{\text{PT}} \mid \cos(\nabla_\theta \mathcal{L}_{\text{LM}}(d, \theta^{\text{PT}}), \nabla_\theta \mathcal{L}_{\text{task}}(D^{\text{task}}; \theta^{\text{PT}})) > \delta_1\} \tag{2}$$

We measure a cosine similarity between the gradient for each example in D^{PT} and the batch gradient for the task.[8] We then select for S_1 the top-k examples in D^{PT} with highest cosine; δ_1 is simply the cosine score of the k-th ranked example. §3.4 specifies the size of selection along with other hyperparameters.

Now with the first data evidence subset S_1, we continue pretraining an intermediate model θ^{PT}_1:

$$\theta^{\text{PT}}_1 \leftarrow \theta^{\text{PT}} + \textit{updates}_{\theta,\mathcal{L}}(S_1) \tag{3}$$

The procedure to find the rest of the data evidence subset S_i with $i = 2, 3, \ldots, m$ is similar to the above but with one difference: these subsets should ideally be beneficial to the model in a way that is not already fully captured by S_1.

We hypothesize that the intermediate model θ^{PT}_1 captures information about S_1. Therefore, for later iterations we want to calculate a task batch gradient based on the previous intermediate model, $\nabla_\theta \mathcal{L}_{\text{task}}(D^{\text{task}}; \theta^{\text{PT}}_{i-1})$. The data evidence subset at each iteration should again exert a similar gradient:

$$S_i = \{d \in D^{\text{PT}} \mid \cos(\nabla_\theta \mathcal{L}_{\text{LM}}(d, \theta^{\text{PT}}_{\lfloor i-1 \rfloor}), \nabla_\theta \mathcal{L}_{\text{task}}(D^{\text{task}}; \theta^{\text{PT}}_{i-1})) > \delta_i\} \tag{4}$$

with δ_i as a threshold for selecting $|S_i|$ elements like δ_1. $\lfloor i-1 \rfloor$ is a design choice that can allow for a "lagged" model (i.e., computing the gradient of the LM loss w.r.t. the model several iterations before vs. the immediate previous intermediate model). This lagging aims to improve the stability of the method. For the experiments in this work, $\theta^{\text{PT}}_{\lfloor i-1 \rfloor}$ is by default θ^{PT}, for a maximum lagging; $\theta^{\text{PT}}_{\lfloor i-1 \rfloor}$ is θ^{PT}_{i-1} in cases denoted by *NL* (no lagging).[9]

At each iteration, having a total of i data evidence subsets, we continue pretraining an intermediate model θ^{PT}_i:

$$\theta^{\text{PT}}_i \leftarrow \theta^{\text{PT}} + \textit{updates}_{\theta,\mathcal{L}}(\cup^i_{j=1} S_j) \tag{5}$$

[6] A discussion over the limitations of our problem formulation can be found in §A.

[7] The task loss over a single task example is $\mathcal{L}_{\text{task}}(x^{\text{task}}, y^{\text{task}}) = -\log p_\theta(\text{verbalizer}(y^{\text{task}}) \mid \text{template}(x^{\text{task}}))$.

[8] The LM loss over a single pretraining example is $\mathcal{L}_{\text{LM}}(x^{\text{PT}}_{\text{context}}, y^{\text{PT}}_{\text{masked}}) = -\log p_\theta(y^{\text{PT}}_{\text{masked}} \mid x^{\text{PT}}_{\text{context}})$.

[9] Compared to θ^{PT}, the no-lagging version uses θ^{PT}_{i-1} which comes from continuing pretraining over a small amount of examples and can have a higher variance. Adding this lagging also remotely shares intuition with some RL methods addressing training stability (Mnih et al., 2016).

It is worth noting that for every iteration, we continue pretraining over the *original* language model, and the data evidence subsets are unordered.

After the m-th iteration, we complete building our full supporting data evidence $S = \cup_{j=1}^{m} S_j$. The resulting model θ_m^{PT} is essentially the final upweighted model, i.e., $\theta_{\text{new}}^{\text{PT}}$ introduced in §2.1.

3 Experimental Setup

3.1 Baseline methods

ORCA is a greedy algorithm and thus not guaranteed to find the global optimal SDE out of the $\binom{|D^{\text{PT}}|}{|S|}$ candidates. To evaluate the efficacy of the identified evidence, we compare ORCA with the following baseline methods:

Random sampling We simply sample at random $|S|$ examples from D^{PT} as the SDE.

Embedding nearest neighbors Enhancing language models using examples with nearest neighboring embeddings is a common approach in domain adaptation of LMs and kNN-LMs (Gururangan et al., 2020; Khandelwal et al., 2020). Here we find nearest neighboring pretraining examples to the task examples. We define a similarity score as below:

$$\cos(h_{\text{masked}}(\hat{x}^{\text{PT}}_{\text{context}}), h_{\text{verbalizer}}(\text{template}(\hat{x}^{\text{task}}))) \tag{6}$$

- h_{masked} is the last hidden representation at the position of the masked pretraining token.
- $h_{\text{verbalizer}}$ is the last hidden representation at the position of the task verbalizer token.
- $\hat{x}^{\text{PT}}_{\text{context}}$ is the pretraining input to the model but containing the ground truth masked token.
- $\text{template}(\hat{x}^{\text{task}})$ is the templated task input but supplying the ground truth verbalized label.

We use the ground truth information here for a fair comparison with our method ORCA, where the calculation of gradients involves the ground truth information as well.

Practically, since $|D^{\text{task}}|$ can be large and well over $|S|$, we first sample t examples from D^{task}. Then, for each of the t sampled task examples, we find the top-k nearest neighboring pretraining examples in D^{PT}. Finally, from the pool of the $t \cdot k$ pretraining examples, we sample $|S|$ of them as the SDE. We additionally have a hyperparameter max-r controlling the maximum allowed data repetitions in the selected data evidence.

Iterative selection with embeddings We use the gradient information $\cos(\nabla_\theta \mathcal{L}_{\text{LM}}(.), \nabla_\theta \mathcal{L}_{\text{task}}(.))$ when collecting the SDE subsets in ORCA iteratively. Here we test whether we can substitute the gradients with embeddings while keeping the selection iterative. Reusing the notations in the embedding nearest neighbors baseline, we use $\cos(h_{\text{masked}}(.), \frac{1}{|D^{\text{task}}|}\sum_{x^{\text{task}}} h_{\text{verbalizer}}(.))$ for all the gradient cosine operations in ORCA. Note that we use the average embeddings of all task examples to replace the batch gradient over all task examples. Other design decisions of ORCA remain unchanged.

3.2 Language model and downstream tasks

BERT We use the BERT-large language model as θ^{PT} (Devlin et al., 2019).[10]

IMDB We primarily experiment with two text classification tasks, sentiment analysis and textual entailment.[11] For sentiment analysis, we use the IMDB movie review dataset (Maas et al., 2011). The task data D^{task} here is the IMDB test split containing 25,000 examples.[12] The template for the IMDB examples is "*It was* `[MASK]`. *<REVIEW>*". The verbalizer maps the reconstruction of the `[MASK]` token to the label space, {"good" $\rightarrow$ $positive$, "bad" $\rightarrow$ $negative$}.

[10] We choose it primarily due to the limited computing resources we have—BERT is small both in terms of the number of model parameters and the size of the original pretraining data. ORCA is extendable to other language models as well.

[11] Future work can explore more tasks, but we select two typical ones here mainly due to our computational resources (more details in §B).

[12] The use of test set is our deliberate choice in this work. In §D, we further discuss the purpose of it and show a sanity check on an alternative setup using the training set.

MNLI For the textual entailment task, we use the MNLI dataset (Williams et al., 2018). The task data D^{task} here is the MNLI matched validation split containing 9,815 examples (the test split is private). The template for the MNLI examples is "*<PREMISE>* `[MASK]`, *<HYPOTHESIS>*". The verbalizer maps the reconstruction of the `[MASK]` token to the label space, {"yes" $\rightarrow$ $entailment$, "no" $\rightarrow$ $contradiction$, "maybe" $\rightarrow$ $neutral$}. We use the OpenPrompt library (Ding et al., 2022) to prompt the BERT model with the templates and verbalizers inherited from Gao et al. (2021b).

Zero-shot transfer and prompt tuning When we formulate our problem, we are interested in the evidence in *pretraining* that directly impact the pretrained model's performance on the downstream task—a zero-shot transfer scenario. There is no notion of *finetuning* with the in-task training data. However, research in prompt tuning (e.g., Lester et al., 2021) folds the usage of in-task training data into the template for the task. A sequence of soft embeddings is added to the beginning of the template and trained with the in-task training data. The language model parameters remain unchanged. Apart from our main experiments with the zero-shot transfer model, we also consider such prompt tuning scenarios, finding pretraining data evidence useful for the task when the template is enhanced with some in-task training data.[13]

3.3 PRETRAINING DATA

Source BERT uses the English *Wikipedia* and *BookCorpus* (Zhu et al., 2015) as its pretraining data. During pretraining, 15% of the tokens are randomly masked out to be reconstructed. Though BERT's pretraining data is already small compared to those of many other language models (e.g., Raffel et al., 2020; Gao et al., 2021a), we unfortunately still do not have the resource to process the full dataset. In fact, in this work we only randomly sample 0.5% of the full pretraining data.

Format During pretraining, BERT would reconstruct the masked 15% tokens in a sequence in parallel (i.e., the reconstruction loss for each token is independent). From the training perspective, this is efficient. However, this work aims to find the SDE. We particularly want to know learning the reconstruction of *which* token could most impact the downstream task performance. Therefore, we expand each pretraining data and treat each masked token as a standalone example.[14] More specifically in our setup, $D^{\text{PT}} \ni (x^{\text{PT}}_{\text{context}}, y^{\text{PT}}_{\text{masked}})$. $x^{\text{PT}}_{\text{context}}$ is a sequence of 512 tokens, and $y^{\text{PT}}_{\text{masked}}$ is a single masked token in the sequence. Together this makes $|D^{\text{PT}}|$ = 3,924,635 (with 52,640 unique $x^{\text{PT}}_{\text{context}}$ sequences). We choose at most 2,000 instances from D^{PT} as the SDE S.

3.4 HYPERPARAMETERS

ORCA finds S in iterations. In this work we use m=20 iterations, with each iteration finding 100 examples from D^{PT} (with a total $|S|$=2000).

For the embedding kNN baseline, we sample t=1000 task examples and choose k={10, 20, 50, 100} most similar pretraining data. Within the $t \cdot k$ candidate pool, we sample $|S|$=2000 examples, with a max number of repetitions r={1, 20, 2000}.[15]

During the continued pretraining for all methods, we use a batch size of 16, resulting in *at most 125 optimizer updates* from the original language model. The learning rate is set at one of BERT's default values 2e-5.

4 EVALUATION

[13] We use different amounts of in-task training data for prompt tuning depending on the task performance. For IMDB, we use 100 examples per class, whereas for MNLI, we use 10,000 examples per class.

[14] Other masked tokens are still masked in the input context, to be faithful to the original LM objective.

[15] In our method ORCA, though the examples *within* S_i are strictly non-overlapping, we don't enforce distinctiveness *across* S_i since we only work with 0.5% of the pretraining data. This means an example could at *maximum* appear 20 times in our method. Therefore, we include the $r > 1$ options for the embedding kNN baseline as well for a fairer comparison. In §C we further discuss the choice of t and k.

[16] Additional details and discussion of the results can be found in §E.

Table 1: Main results (accuracy) of ORCA and baselines on the zero-shot model. NL means the no-lagging variant. Numbers in regular fonts are averaged from 5 random seeds, while numbers in small fonts show a trajectory of performance with one seed.[16]

On zero-shot model	IMDB	MNLI
Null	*73.50*	*43.70*
Random	71.25 ± 2.56	42.56 ± 0.43
Embedding kNN	76.55 ± 2.16	45.15 ± 0.48
Iterative embeddings	75.11 ± 4.59	43.74 ± 0.68
ORCA (NL)	**84.51** ± 0.77	45.46 ± 0.73
$0 < \|S\| \leq 500$	79.81	44.85
$500 < \|S\| \leq 1000$	83.87	45.64
$1000 < \|S\| \leq 1500$	84.40	46.10
$1500 < \|S\| \leq 2000$	85.17	46.49
ORCA	84.33 ± 1.51	**46.06** ± 0.35
$0 < \|S\| \leq 500$	81.60	45.99
$500 < \|S\| \leq 1000$	83.23	45.75
$1000 < \|S\| \leq 1500$	84.42	46.40
$1500 < \|S\| \leq 2000$	85.15	46.26

We evaluate the supporting data evidence S, identified using ORCA and the baselines, by quantifying the supportiveness of S (as defined in §2.1). Note that this section does not focus on whether or not the discovered data evidence is plausible to humans. We will explore what humans can interpret from the actual data evidence in §5.

Table 1 shows our main results: the performance of our zero-shot language model pretrained additionally on S, as identified by different methods. We first notice a performance gap between our original model on IMDB and MNLI, indicating the entailment task is intrinsically harder for models that have only been trained on pretraining data. We observe a moderate performance improvement using the embedding nearest neighbors method ($|S| = 2000$). The best performance is achieved by our proposed method ORCA, especially in the task of IMDB by a large margin (even with $|S| \leq 500$), showing the effectiveness of our method.

Table 2 shows some additional results on the effect of the SDE S on a prompt-tuned model. These results show that, compared to the zero-shot model, a prompt-tuned model is more difficult to improve since the prompt may already be highly specialized towards the task, using the in-task training data. The *additional* signals in the pretraining data that are useful to the task can be scarce. That said, the pretraining data S identified by ORCA still improves the model on IMDB.

5 Analysis

While useful in showing the effectiveness of the SDE S, evaluations in §4 do not provide us with tangible insights about the model itself. In this section, we analyze some properties of S, and see whether they reflect humans' expectations for the model. We first show a few qualitative examples of the evidence discovered by ORCA in Table 3.

Which source corpus does the supporting data evidence come from? The pretraining data of BERT consist of the English Wikipedia and BookCorpus. We show the source corpus of examples in S in Figure 1.

Table 2: Additional results (accuracy) of ORCA and the baselines on the prompt-tuned model. All the numbers are averaged from 5 random seeds.

On prompt-tuned model	IMDB	MNLI
Null	*87.83*	***70.19***
Random	86.06 ± 0.82	69.07 ± 0.50
Embedding kNN	86.53 ± 0.77	68.93 ± 0.61
Iterative embeddings	87.80 ± 0.03	68.45 ± 0.36
ORCA (NL)	87.65 ± 1.10	68.79 ± 0.28
ORCA	**88.10** ± 0.65	68.61 ± 0.44

We find that though the pretraining set consists of considerably more data from Wikipedia than from BookCorpus (76.5% vs. 23.5%), the SDE identified by ORCA has a drastically different source corpus distribution. In IMDB, 64.1% and 92.6% of the examples in S come from BookCorpus, using the default ORCA and its no-lagging variant respectively. The demotion of Wikipedia examples in the sentiment analysis task is somewhat reasonable, since Wikipedia is meant to have a neutral point of view (NPOV).[17] On the other hand, BookCorpus consists of novels that could involve strong emotions and sentiments.

[17] https://en.wikipedia.org/wiki/Wikipedia:Neutral_point_of_view

Table 3: Examples of the supporting data evidence (S) in pretraining data discovered by ORCA for IMDB and MNLI. The masked token ($y^{\text{PT}}_{\text{masked}}$) in each example is underlined. The example evidence for IMDB expresses sentiments, while it is less clear whether the example evidence for MNLI is related to entailment.

IMDB	... we have to think that were awfully lucky as human beings to have the nice precise system. the sloppy system is probably **good** enough for bacteria. it turns out – much to geneticists surprise – that lowly bacteria store genes as whole units (weasel) ...
	it was the only place she could afford. her meager earnings didnt provide much in the ways of clean, modern style along with the privacy she required. she felt better if she thought about how **bad** it could be. a year ago, shed lived with her mother. anywhere was better than living with her ...
MNLI	... he then cut the cord that bound her hands and legs. are you ok to walk? he asked hoping the answer was **yes**. i think so. but im quite stiff, she said. he helped her up. stretch your legs a little. theyll feel better ...
	... there was no way to hide the shock on her face, and she knew he saw it by his sigh. "do you think yourself less than me?" "**no**!" she absolutely didn't but ... he nodded his head. "i see. you thought i would think you were less than me." she was ashamed. "i'm sorry." ...
	... he shook his head, incredulous. in fact, he looked like he was considering throttling me. "you're just not getting it. maybe that's my fault. **maybe** it's because i don't tell you i love you often enough. baby, you're the only 'good' thing that i've ever had ...

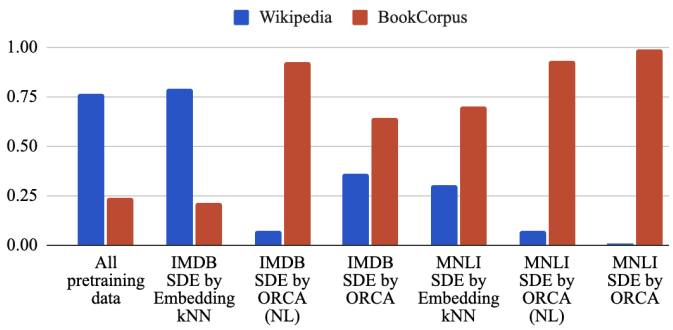

Figure 1: Source corpus distribution of the supporting data evidence (SDE) in IMDB and MNLI.

A similar trend can be found in MNLI as well, with 99.0% and 92.9% of the examples in S coming from BookCorpus, using the default and NL variant of ORCA. We conjecture that the over-reliance on BookCorpus in MNLI could be due to the selection of the colloquial verbalizer words (e.g., "yes", "maybe"), which can be scarce in Wikipedia. Also, the BookCorpus data could contain more everyday topics that match MNLI's genres (e.g., fiction, letters, telephone speech). However, whether it is reasonable for the model to rely on BookCorpus for textual entailment is arguable: Wikipedia should be a more reliable source if we want the model to build more upon factual information.

What are the masked tokens in the supporting data evidence? Prompted language models use a verbalizer to adapt to the downstream task. For example, outputting "good" for a templated IMDB input indicates a positive sentiment, "yes" for MNLI indicates entailment, etc. For a pretraining example that supports the task, are there any relations between its masked, to-be-reconstructed pretraining token ($y^{\text{PT}}_{\text{masked}}$) and the verbalizer words for the task (verbalizer(y^{task}))? In Table 4, we show the 10 most frequent masked words (types) in S, for each method in IMDB and MNLI.

We observe that the verbalizer words, in their original forms, are always the most common masked token in S. For all of the methods in both tasks, over 50% of the masked tokens are exactly the verbalizer words. Though there is some noise in $y^{\text{PT}}_{\text{masked}}$ (e.g., symbols that carry no task-relevant meaning), most of the other masked tokens are synonyms to the verbalizer words in IMDB by observation. In MNLI, the other masked tokens may capture relations between clauses similar to the verbalizer words (e.g., then, to, probably). Overall, we find that $y^{\text{PT}}_{\text{masked}}$ in the discovered S is reasonable for the sentiment analysis and textual entailment task.

Table 4: Masked tokens ($y^{\text{PT}}_{\text{masked}}$) in the supporting data evidence of IMDB and MNLI.

Task	Method	Most frequent $y^{\text{PT}}_{\text{masked}}$ in S
IMDB	Embedding kNN	**bad**, **good**, terrible, great, badly, excellent, worst, negative, better, disappointment, ... [*11 distinct tokens in total, 94.8% **verbalizer words***]
	ORCA (NL)	**bad**, **good**, worst, n, worse, ', wrong, -, horrible, poisonous, ... [*91 distinct tokens in total, 90.0% **verbalizer words***]
	ORCA	**bad**, **good**, `, horrible, not, worse, ugly, hated, poor, terrible, ... [*285 distinct tokens in total, 55.9% **verbalizer words***]
MNLI	Embedding kNN	**no**, **yes**, **maybe**, `, yeah, However, perhaps, n,), No, ... [*258 distinct tokens in total, 58.3% **verbalizer words***]
	ORCA (NL)	**maybe**, **yes**, **no**, `, n, that, -, then, perhaps, the, ... [*176 distinct tokens in total, 69.1% **verbalizer words***]
	ORCA	**maybe**, **yes**, **no**, `, perhaps, to, probably, has, in, big, ... [*125 distinct tokens in total, 59.4% **verbalizer words***]

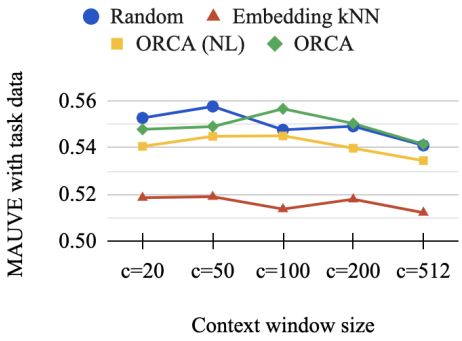

Figure 2: MAUVE similarity on **IMDB**, between the sets of $x^{\text{PT}}_{\text{context}}$ in S and x^{task}.

Is the context of the supporting data evidence similar to the task input data? We are interested in the relationship between the context of the selected pretraining data ($x^{\text{PT}}_{\text{context}}$) and the input of the downstream task (x^{task}). Are they exceptionally similar, indicating that the model may be memorizing shallow patterns? Alternatively, are they considerably different, indicating that the model needs to transfer some learnt knowledge from pretraining to the task (either in a reasonable or spurious way)? Our exploratory step uses an automatic metric between two distributions of texts, MAUVE (Pillutla et al., 2021), to measure the similarity between our sets of $x^{\text{PT}}_{\text{context}}$ and x^{task}. As a method based on quantized language model embeddings, MAUVE similarity may capture text attributes such as topics and style.[18]

Apart from using all 512 tokens in the context of the data evidence ($x^{\text{PT}}_{\text{context}}$), we also truncate the context, keeping the surrounding c tokens of the masked token ($y^{\text{PT}}_{\text{masked}}$). We control for the scope of the context by varying c. For x^{task}, we randomly sample 2000 examples to match the size of S.[19] Figure 2 shows the results on IMDB. See §F for MNLI results.

We observe that the MAUVE scores between $x^{\text{PT}}_{\text{context}}$ and x^{task} are all between 0.512 and 0.577. In contrast, the MAUVE score between the training set of the task and the test set (x^{task}) is 0.998 and 0.997 for IMDB and MNLI respectively. This substantial difference in MAUVE scores may indicate a disparity in topics and style between the context of the pretraining evidence and the task data. Additionally, the MAUVE score of our selected SDE is not higher than a random sample in most cases. This further shows that the signal in the evidence context useful for the task is *subtle*, in a way that MAUVE cannot capture. This is in contrast to the conjecture that the model must have seen the exact inputs in the pretraining data and is only performing a shallow memorization.

While not within the scope of this paper, future investigation can also extend the analysis of the supporting evidence with feature attribution methods (Pezeshkpour et al., 2022) or a human evaluation with domain experts of the task (e.g., what exact spans in the SDE contribute to their supportiveness). We further discuss the limitations of our method, computational resources, and future directions in §A and §B.

[18] Grammaticality can be another attribute as Pillutla et al. (2021) work with machine-generated texts. This is less relevant in our case as our sets of texts are naturally occurring.

[19] Here x^{task} is without template, and $x^{\text{PT}}_{\text{context}}$ has the masked token recovered. This is to give MAUVE most natural texts for evaluation.

6 Related Work

LLMs have been showing competence in various downstream tasks in NLP with little to no task-specific tuning, using prompts (Petroni et al., 2019; Brown et al., 2020; Schick & Schütze, 2021; Gao et al., 2021b; Lester et al., 2021). We are especially interested in interpreting LLMs under a zero-shot setup, where the knowledge relevant to the downstream task must come from the noisy pretraining data.[20]

One common interpretability method for NLP models is feature attribution, where important tokens or spans in the inference-time input are highlighted, indicating their contributions to the model's decision (Simonyan et al., 2014; Li et al., 2016; Ribeiro et al., 2016; Lundberg & Lee, 2017). However, information relevant to explaining the model's decisions (especially if abstract) is often not in the inference-time input (Han et al., 2020; Wiegreffe & Marasović, 2021; Pezeshkpour et al., 2022). On language models, feature attribution has been used to interpret and verify grammatical phenomena (Yin & Neubig, 2022).

Another type of interpretation that aligns more with our focus is instance attribution, where important training examples are highlighted for their influence on the model (Koh & Liang, 2017; Yeh et al., 2018; Pruthi et al., 2020; Han et al., 2020; Guo et al., 2021). In this work, we are instead interested in the influence of pretraining data and in finding SDE for the entire task rather than individual test examples.[21] There has also been prior work analyzing what amount of data is needed during pretraining to achieve models with certain capabilities (Zhang et al., 2021), but these works do not attribute model performance to specific pretraining data.

Our formulation may seem similar to prior work in task-enhancing pretraining (Han & Eisenstein, 2019; Gururangan et al., 2020; Yao et al., 2021). However, such methods typically use a large amount of loosely relevant pretraining data along with the in-task training data, to improve performance. We instead aim to find an orders-of-magnitude-smaller set of pretraining data, providing a clearer signal of their impact on the task for interpretability purposes.

Our proposed method to find the data evidence, ORCA, shares a similar intuition with prior work that reweighs training data (Wang et al., 2020), as both methods use the gradient information of the test data. However, their target model depends on an *ordered sequence* of data weights and model checkpoints. In contrast, we apply an *unordered* data evidence set to the *original* model, mimicking an upweighting in pretraining. In addition to the difference in methods, the purpose is different as well: theirs is performance-oriented while ours is interpretability-oriented.

Other remotely related line of work in machine learning includes coreset construction (Coleman et al., 2020; Mirzasoleiman et al., 2020; Huang et al., 2021) and dataset distillation (Wang et al., 2018; Zhao et al., 2021). Their focus is typically an empirical risk minimization problem on the training data, without a notion of downstream tasks or task transfer. They aim to create a substitution set for the full training data for an efficiency purpose.

7 Conclusion

The source of competence of zero- and few-shot prompted language models on downstream tasks is mysterious. The models should be gaining task-specific knowledge from the pretraining data, but what pretraining data leads to the capability of the models is an underexplored area of research. In this work, we formulate the problem of finding supporting data evidence in the pretraining data of LLMs for downstream tasks. We propose ORCA to effectively identify such evidence with an iterative guide from task-specific gradient information. Deeper analyses into the evidence show that a prompted BERT on sentiment analysis and textual entailment relies heavily on the BookCorpus data, as well as on pretraining examples that mask out task verbalizers and their synonyms.

[20] Interpreting the role of pretraining data in an unprompted, finetuning setup is intrinsically harder, but prior work like Chen et al. (2020) have made attempts.

[21] A concurrent work by Akyürek et al. (2022) builds a candidate set for fact-tracing in question answering; the difference is the use of task-related training examples instead of pretraining data, and an information retrieval evaluation.

REFERENCES

Ekin Akyürek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. Tracing knowledge in language models back to the training data. *ArXiv*, abs/2205.11482, 2022.

Sanjeev Arora. Advanced algorithm design lecture 9 : High dimensional geometry , curse of dimensionality , dimension reduction. 2013.

Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *ICLR*, 2021.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? *Proc. FAccT*, 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Xiaodong Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *USENIX Security Symposium*, 2021.

Hongge Chen, Si Si, Yang Li, Ciprian Chelba, Sanjiv Kumar, Duane Boning, and Cho-Jui Hsieh. Multi-stage influence function. In *Proc. NeurIPS*, 2020.

Cody A. Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei A. Zaharia. Selection via proxy: Efficient data selection for deep learning. In *Proc. ICLR*, 2020.

R Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, 19(1): 15–18, 1977.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*, 2019.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Haitao Zheng, and Maosong Sun. Openprompt: An open-source framework for prompt-learning. In *Proc. ACL (System Demonstrations)*, 2022.

Leo Gao, Stella Rose Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *ArXiv*, abs/2101.00027, 2021a.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proc. ACL*, 2021b.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *Proc. ICML*, 2019.

Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. Fastif: Scalable influence functions for efficient model interpretation and debugging. In *Proc. EMNLP*, 2021.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proc. ACL*, 2020.

Xiaochuang Han and Jacob Eisenstein. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *Proc. EMNLP*, 2019.

Xiaochuang Han and Yulia Tsvetkov. Fortifying toxic speech detectors against veiled toxicity. In *Proc. EMNLP*, 2020.

Xiaochuang Han and Yulia Tsvetkov. Influence tuning: Demoting spurious correlations via instance attribution and instance-driven updates. In *Proc. Findings of EMNLP*, 2021.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proc. ACL*, 2020.

Horace He and Richard Zou. functorch: Jax-like composable function transforms for pytorch. `https://github.com/pytorch/functorch`, 2021.

Ari Holtzman, Peter West, Vered Schwartz, Yejin Choi, and Luke Zettlemoyer. Surface form competition: Why the highest probability answer isn't always right. In *Proc. EMNLP*, 2021.

Jiawei Huang, Ru Huang, Wenjie Liu, Nikolaos M. Freris, and Huihua Ding. A novel sequential coreset method for gradient descent algorithms. In *Proc. ICML*, 2021.

Alon Jacovi, Ana Marasović, Tim Miller, and Yoav Goldberg. Formalizing trust in artificial intelligence: Prerequisites, causes and goals of human trust in ai. *Proc. FAccT*, 2021.

Karthikeyan K and Anders Søgaard. Revisiting methods for finding influential examples. *ArXiv*, abs/2111.04683, 2021.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *Proc. ICLR*, 2020.

Vid Kocijan and Samuel R. Bowman. Influence functions do not seem to predict usefulness in nlp transfer learning. `https://wp.nyu.edu/cilvr/2020/08/27/influence-functions-do-not-seem-to-predict-usefulness-in-nlp-transfer-learning/`, 2020.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proc. ICML*, 2017.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proc. EMNLP*, 2021.

Jiwei Li, Xinlei Chen, Eduard H. Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. In *Proc. NAACL-HLT*, 2016.

Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

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*, abs/2107.13586, 2021.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Proc. NeurIPS*, 2017.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proc. ACL*, 2011.

R. Thomas McCoy, Paul Smolensky, Tal Linzen, Jianfeng Gao, and Asli Celikyilmaz. How much do language models copy from their training data? evaluating linguistic novelty in text generation using raven. *ArXiv*, abs/2111.09509, 2021.

Baharan Mirzasoleiman, Jeff A. Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *Proc. ICML*, 2020.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. Language models as knowledge bases? In *Proc. EMNLP*, 2019.

Pouya Pezeshkpour, Sarthak Jain, Sameer Singh, and Byron C. Wallace. Combining feature and instance attribution to detect artifacts. In *Proc. Findings of ACL*, 2022.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaïd Harchaoui. Mauve: Measuring the gap between neural text and human text using divergence frontiers. In *Proc. NeurIPS*, 2021.

Danish Pruthi, Bhuwan Dhingra, Livio Baldini Soares, Michael Collins, Zachary Chase Lipton, Graham Neubig, and William W. Cohen. Evaluating explanations: How much do explanations from the teacher aid students? *TACL*, 10:359–375, 2022.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracking gradient descent. In *Proc. NeurIPS*, 2020.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. Impact of pretraining term frequencies on few-shot reasoning. *ArXiv*, abs/2202.07206, 2022.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Proc. KDD*, 2016.

Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proc. EACL*, 2021.

Lloyd S Shapley. Notes on the n-person game—ii: The value of an n-person game. 1951.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. ICLR*, 2014.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proc. ACL*, 2019.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation. *ArXiv*, abs/1811.10959, 2018.

Xinyi Wang, Hieu Pham, Paul Michel, Antonios Anastasopoulos, Jaime Carbonell, and Graham Neubig. Optimizing data usage via differentiable rewards. In *Proc. ICML*, 2020.

Sarah Wiegreffe and Ana Marasović. Teach me to explain: A review of datasets for explainable natural language processing. In *NeurIPS Datasets and Benchmarks*, 2021.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. NAACL*, 2018.

Xingcheng Yao, Yanan Zheng, Xiaocong Yang, and Zhilin Yang. Nlp from scratch without large-scale pretraining: A simple and efficient framework. *ArXiv*, abs/2111.04130, 2021.

Chih-Kuan Yeh, Joon Sik Kim, Ian En-Hsu Yen, and Pradeep Ravikumar. Representer point selection for explaining deep neural networks. In *Proc. NeurIPS*, 2018.

Kayo Yin and Graham Neubig. Interpreting language models with contrastive explanations. *ArXiv*, abs/2202.10419, 2022.

Yian Zhang, Alex Warstadt, Haau-Sing Li, and Samuel R. Bowman. When do you need billions of words of pretraining data? In *Proc. ACL*, 2021.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *ICLR*, 2021.

Ruiqi Zhong, Steven Shao, and Kathleen McKeown. Fine-grained sentiment analysis with faithful attention. *arXiv preprint arXiv:1908.06870*, 2019.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proc. ICCV*, 2015.

A Limitations of the problem formulation

Implication of supporting data evidence One limitation of our setup is that the data outside our defined SDE S may have additional value on the model which the current formulation does not capture. For instance, there could be certain examples that exceptionally help the model capture the grammar of the language, making an indirect contribution to the task. However, they are rather unlikely to be picked as SDE: in the scope of this paper, we focus on finding the evidence that is *directly* related to the downstream task. The indirect evidence is also interesting and may be valuable for future work to investigate further (e.g., via finding which examples contribute most to the *contribution of SDE*).

Alternatives to our problem formulation Another limitation of our problem formulation is that the *history* of the model pretraining is ignored. Is it sufficient to check whether the evidence data help the *fully* pretrained model? What if a data is truly related to the task but got overfitted during earlier stages of pretraining, so continuing pretraining on it would not change the model? We think this is possible, but only to some extent and with a limited risk, since the huge volume of pretraining data can make it difficult to overfit to all examples related to an arbitrary task in a particular way. Nevertheless, to improve that case, a potentially better solution would be continuing pretraining *each checkpoint* of the model across the whole pretraining procedure. This would share an intuition with some instance attribution methods based on model checkpoints (Pruthi et al., 2020). However, we lack the resources to perform such experiments in this work, so we defer that to future research.[22]

B Ethics Statement

One ethical consideration and a limitation of our approach is the large computational cost, and consequently the environmental impact caused by our computationally-expensive method (Strubell et al., 2019). On our machine with 8 Nvidia A40 GPUs, the full 20 epochs of each ORCA experiment would take about 7 days in total. The long computing time is partly because we need to calculate per-sample gradients for 4M data points in each epoch, and also because there is no efficient way to calculate the per-sample gradients in PyTorch at the time of our implementation.[23] However, our goal is to develop a research prototype (which can be optimized in the future) that will enable opening up the black box of large language models. Insights into their pretraining data will potentially lead to positive impacts—removing problematic data sources, demoting spurious correlations, and alleviating other ethical issues caused by our current inability to interpret decisions of large language models.

C Continued discussion on experimental setup

The sample size t in the embedding kNN baseline For the embedding kNN baseline, we sample $t = 1000$ task examples from D^{task} and find $k = \{10, 20, 50, 100\}$ most similar pretraining data to each of the task example. This t should already be large enough since even with the smallest k, $t \cdot k$ is well over $|S|$ (meaning that we need to downsample anyway). We did not use the entire D^{task} because IMDB has 25K examples, and the calculation has a heavy requirement on the storage and memory. Nevertheless, to check whether the current t is adequate, we experiment with $t = 8000$ and $t = 9800$ for IMDB and MNLI respectively. The performance is 74.82 and 45.04, no better than the $t = 1000$ performance in the main evaluation.

[22] With unconstrained resources, one can extend the checkpoint proposal and even measure Shapley values (Shapley, 1951) of the data, an equitable valuation method (Ghorbani & Zou, 2019).

[23] This may be possible with the latest release of functorch (He & Zou, 2021). We plan to work on it and expect a significant speedup.

Exact optimizer steps based on S For all of our experiments, we have $|S| \leq 2000$ and a batch size of 16. We additionally held out 5% of S during optimization as a sanity check for the LM loss. Therefore, *at most 119* optimizer steps were applied in all of the experiments.

D USING THE TEST SET AS D^{TASK}

The use of test set over training set is our deliberate choice in this work.[24] There are two main reasons.

- We assume that the task's test data is the only instantiation of the task, since we are interpreting a model deployed in a zero-shot setup. We cannot assume the availability of a "training set" in such scenario.
- More importantly, this work is about interpretability, not absolute model performance. We are interpreting why the model can achieve a good *test* performance, not a good *training* performance. Similar to other interpretability research, no matter feature attribution or instance attribution (Ribeiro et al., 2016; Koh & Liang, 2017), we must use the test data, since we are interpreting the model's behavior on test examples exactly. This may also help us reveal the data artifacts inside the test set (Han et al., 2020).

That being said, we do have sanity checks at the place we use test data in ORCA, that the effect of using the training set gradient would be very close. For all 20 intermediate stages, the cosine similarity between the test set and training set gradients is $0.977_{\pm 0.023}$ for IMDB and $0.947_{\pm 0.042}$ for MNLI, in a range of [-1,1].[25]

E CONTINUED DISCUSSION ON MAIN EVALUATION

Importance of selecting S in several iterations While not shown in the main evaluation table, the *first* epoch of ORCA without later iterations ($|S| = 100$) would actually *hurt* the performance (57.39 and 37.43 for IMDB and MNLI respectively). This is due to the imbalance of the selected pretraining data, favoring only one label in the task. ORCA's iterative selection strategy in this case is essential, a difference from the instance attribution methods in previous interpretability research.

Calibration While not a focus of this work, prompt-based language models can be improved with calibration techniques such as PMI_{DC} (Holtzman et al., 2021). We did not use such calibration in our work because our pilot study shows a rather even prior distribution among the labels—applying PMI_{DC} on our model (LM, template, verbalizers) in IMDB yields a less than 1% performance improvement.

Hyperparameter search Due to the high computing cost mentioned in §B, we did not perform hyperparameter search in our ORCA experiments (whereas we did a search for the embedding kNN baseline). It is possible that some other sets of ORCA hyperparameters can lead to better performance numbers than those in the main evaluation table. We will release all of the code, experiment scripts, and data at `ANONYMIZED` upon publication.

F ADDITIONAL ANALYSIS RESULTS

In Figure 3, we show the MAUVE similarity analysis on MNLI, accompanying the MAUVE analysis on IMDB in the main paper.

[24] Also note that all of the investigated algorithms, ORCA and the baselines, have the same D^{task} setup.

[25] These numbers mean the effect of using training and test gradients is extremely similar, noting that the dimension of the gradients is very large (340M). Given two random vectors a and b at this dimension, $Pr[|\cos(a, b)| > 0.9] < 1/e^{2.75 \times 10^8}$ (Arora, 2013).

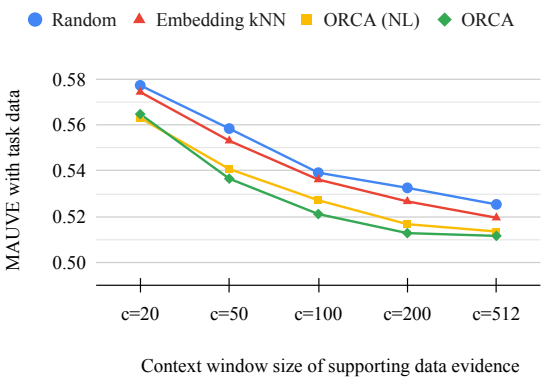

Figure 3: MAUVE similarity on **MNLI**, between the sets of $x^{\text{PT}}_{\text{context}}$ in S and x^{task}.