It’s Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners

Timo Schick\(^1,2\) and Hinrich Schütze\(^1\)

\(^1\) Center for Information and Language Processing, LMU Munich, Germany
\(^2\) Sulzer GmbH, Munich, Germany

timo.schick@sulzer.de

Abstract

When scaled to hundreds of billions of parameters, pretrained language models such as GPT-3 (Brown et al., 2020) achieve remarkable few-shot performance on challenging natural language understanding benchmarks. In this work, we show that performance similar to GPT-3 can be obtained with language models whose parameter count is several orders of magnitude smaller. This is achieved by converting textual inputs into cloze questions that contain some form of task description, combined with gradient-based optimization; additionally exploiting unlabeled data gives further improvements. Based on our findings, we identify several key factors required for successful natural language understanding with small language models.\(^1\)

1 Introduction

Pretraining ever-larger language models (LMs) on massive plain text corpora has led to significant improvements on a wide range of NLP tasks (Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2019, \textit{inter alia}). A standard approach is to replace the pretrained model’s output layer with a task-specific head and finetune the entire model on a set of labeled training data. However, language modeling is not only a powerful pretraining objective, but many tasks can be reformulated as cloze questions (e.g., by appending phrases such as “the correct answer is ___”), allowing pretrained LMs to solve them without any or with only very few labeled examples (Radford et al., 2019; Schick and Schütze, 2020a).

Very recently, Brown et al. (2020) introduced GPT-3, a pretrained LM with an enormous 175 billion parameters, and showed that it has amazing few-shot abilities: By reformulating tasks as language modeling problems, GPT-3 achieves near state-of-the-art results for some tasks in the SuperGLUE benchmark (Wang et al., 2019) given just 32 labeled examples. This is achieved through \textit{priming}: GPT-3 is given a few demonstrations of inputs and corresponding outputs as context for its predictions, but no gradient updates are performed. While being straightforward to use, this method has two major drawbacks:

- It requires a gigantic LM to work well, making it \textit{unusable in many real-world scenarios}.
- It \textit{does not scale to more than a few examples} as the context window of most LMs is limited to a few hundred tokens.

An alternative to priming is \textit{pattern-exploiting training} (PET) (Schick and Schütze, 2020a), which combines the idea of reformulating tasks as cloze questions with regular gradient-based finetuning. While PET additionally requires unlabeled data, unlabeled data is much easier to obtain than labeled training data.
examples for many real-world applications. Crucially, PET only works when the answers to be predicted by the LM correspond to a single token in its vocabulary; this is a severe limitation as many tasks cannot easily be worded that way.

In this work, we modify PET to also work for tasks that require predicting more than one token. We then show that in combination with ALBERT (Lan et al., 2020), PET and its iterative variant (iPET) both outperform GPT-3 on SuperGLUE with 32 training examples, while requiring only 0.1% of its parameters (Figure 1). Finally, we show that similar performance can also be achieved without unlabeled data and provide a detailed analysis of the factors contributing to PET’s strong performance: its ability to combine multiple task formulations, its resilience to wordings that are hard to understand, its usage of labeled data, and characteristics of the underlying LM.

2 Related Work

Enabling LMs to perform zero-shot learning by providing task descriptions was proposed by Radford et al. (2019) and has been applied to text classification (Puri and Catanzaro, 2019), commonsense knowledge mining (Davison et al., 2019) and argumentative relation classification (Opitz, 2019). It is also commonly used for probing the knowledge contained within LMs (Trinh and Le, 2018; Petroni et al., 2019; Talmor et al., 2019; Schick and Schütze, 2020b; Ettinger, 2020, i.a.).

As finding ways to reformulate tasks as cloze questions that are understood well by LMs is difficult (Jiang et al., 2019), Schick and Schütze (2020a) propose PET, a method that uses knowledge distillation (Hinton et al., 2015) to easily combine several reformulations. In contrast to PET, which uses gradient-based optimization, Radford et al. (2019) and Brown et al. (2020) investigate priming, where examples are given as context but no parameter updates are performed.

Our modified version of PET uses masked language models (Devlin et al., 2019) to assign probabilities to sequences of text; this is similar to using them in a generative fashion (Wang and Cho, 2019) and has previously been investigated by Salazar et al. (2020) and Ghazvininejad et al. (2019).

3 Pattern-Exploiting Training

Let $M$ be a masked language model (MLM), $T$ its vocabulary and $z \in T^s$ containing at least $k$ masks and $t \in T$, we denote with $q^k_M(t \mid z)$ the probability that $M$ assigns to $t$ at the $k$th masked position in $z$; the model’s logits before applying softmax are denoted with $s^k_M(t \mid z)$. We consider the task of mapping inputs $x \in X$ to outputs $y \in Y$, for which PET requires a set of pattern-verbalizer pairs (PVPs). Each PVP $p = (P, v)$ consists of

- a pattern $P : X \rightarrow T^s$ that maps inputs to cloze questions containing a single mask;
- a verbalizer $v : Y \rightarrow T$ that maps each output to a single token representing its task-specific meaning in the pattern.

As illustrated in Figure 2, the core idea of PET is to derive the probability of $y$ being the correct token for $x$ from the probability of $v(y)$ being the “correct” token at the masked position in $P(x)$. Based on this intuition, a conditional probability distribution $q_p$ of $Y$ given $X$ is defined as

$$q_p(y \mid x) = \frac{\exp s_p(y \mid x)}{\sum_{y' \in Y} \exp s_p(y' \mid x)} \quad (1)$$

where $s_p(y \mid x) = s^1_M(v(y) \mid P(x))$ is the raw score of $v(y)$ at the masked position in $P(x)$.

For a given task, identifying PVPs that perform well is challenging in the absence of a large development set. Therefore, PET enables a combination of multiple PVPs $P = \{p_1, \ldots, p_n\}$ as follows:

1. For each PVP $p$, a MLM is finetuned on training examples $(x, y)$ by minimizing the cross entropy between $y$ and $q_p(y \mid x)$. In practice, Schick and Schütze (2020a) train three MLMs per pattern as performance can vary substantially between runs.
As steps (2) and (3) above closely resemble knowledge distillation (Hinton et al., 2015), we also refer to them simply as distillation.

To give MLMs trained on different patterns further opportunity to learn from one another, Schick and Schütze (2020a) also propose IPET, an iterative variant of PET in which several generations of models are trained on datasets of increasing size that are labeled by previous generations. This is achieved as follows: First, an ensemble of MLMs is trained as in regular PET. For each model $M_i$, a random subset of other models is used to generate a new training set $T_i$ by assigning labels to those unlabeled examples for which the selected subset of models is most confident in its prediction. Each $M_i$ is then retrained on $T_i$; this process is repeated several times, each time increasing the number of examples in $T_i$ by a constant factor.

### 3.1 PET with Multiple Masks

An important limitation of PET is that the verbalizer $v$ must map each possible output to a single token, which is impractical or outright impossible for many tasks. We thus generalize verbalizers to functions $v : Y \rightarrow T^*$; this requires some modifications to inference and training.\(^2\) We further generalize PET in that we do not assume the output space to be identical for each input: for each $x \in X$, we denote with $Y_x \subseteq Y$ the set of possible outputs given $x$ as input. Given a PVP $p = (P, v)$, we define $l(x) = \max_{y \in Y_x} |v(y)|$ to be the maximum number of tokens required to express any output in $Y_x$ and $P^k(x)$ to be $P(x)$ with the mask token replaced by $k$ masks.

\(^2\)While our approach can easily be adapted to models capable of doing conditional generation with bidirectional context (e.g., Lewis et al., 2020; Raffel et al., 2019), we stick with MLMs as they are more lightweight and we found them to perform better on simple cloze tasks in preliminary experiments.

#### Inference

For $x \in X$, $y \in Y_x$ and $|v(y)| = k$, we redefine $q_p(y \mid x)$ in an autoregressive fashion: Starting from $P^k(x)$, we perform $k$ consecutive predictions, where we always select the next token to predict based on the MLM’s confidence. That is, we set $q_p(y \mid x) = q(v(y) \mid P^k(x))$ where

$$q(t_1 \ldots t_k|z) = \begin{cases} 1 & \text{if } k = 0 \\ \frac{q_M(t_j|z) \cdot q(t'|z')}{\sum_{y \in Y} q(t'|z')} & \text{if } k \geq 1 \end{cases}$$

with $j = \arg\max_{i=1}^k q_M(t_j \mid z)$, $z'$ is $z$ except $z_j' = t_j$ and $t' = t_1 \ldots t_{j-1}t_{j+1} \ldots t_k$. Note that unlike in original PET (Eq. 1), $q_p$ is not a probability distribution as its values do not sum to one.

#### Training

Computing $q_p(y \mid x)$ as in Eq. 3 for each training example $(x, y)$ would be prohibitively expensive. To enable computation of all required probabilities in a single forward pass, we approximate $q_p(y \mid x)$ by (i) always inserting the maximum number of mask tokens required to express any output and (ii) for each $y' \in Y_x$, predicting all tokens in $v(y') = t_1 \ldots t_k$ in parallel, where we simply ignore the model’s predictions for all $l(x) - k$ superfluous mask tokens:

$$\tilde{q}_p(y' \mid x) = \prod_{i=1}^k \tilde{q}_p(t_i \mid P^k(x))$$

As $\tilde{q}_p$ is not a probability distribution over $Y_x$, cross entropy is not an ideal training objective as it can also be minimized by reducing the probability assigned to sequences $z \notin v(Y_x)$ that are not part of the output space, despite this having no effect on the model’s prediction. We instead opt for multiclass hinge loss (Weston and Watkins, 1999; Dogan et al., 2016) and minimize:

$$\sum_{y' \in Y_x} \max \left(0; 1 - \log \tilde{q}_p(y|x) + \log \tilde{q}_p(y'|x) \right)$$

That is, we require the difference between the log probability of $y$ and the log probability of any output $y' \in Y_x \setminus \{y\}$ to be at least 1.

### 4 Experiments

We use the SuperGLUE benchmark (Wang et al., 2019) to compare PET and GPT-3. We cannot evaluate PET using the exact same training data as GPT-3 because for most tasks, GPT-3 uses a different set of training examples for each test example and for the other tasks, training sets were not available upon request; however, the exact choice
of examples has little impact on GPT-3’s performance. We thus create new training sets by randomly selecting 32 examples for each task using a fixed random seed. Following previous work (Raffel et al., 2019; Brown et al., 2020), we select only positive examples for WSC; for both MultiRC and ReCoRD, we follow Brown et al. (2020) and select a total of 32 questions.

We additionally create sets of up to 20,000 unlabeled examples for each task; this is done by removing all labels from the original training sets. As the training sets for RTE and CB are very small, we additionally select random unlabeled examples from MNLI (Williams et al., 2018) for both tasks. We refer to the resulting sets of training examples and unlabeled examples as FewGLUE.

4.1 Tasks

Below, we describe each of the SuperGLUE tasks and our corresponding PVPs. We use a vertical bar (|) to mark boundaries between text segments.

**BoolQ** (Clark et al., 2019) is a QA task where each example consists of a passage p and a yes/no question q. We use the following patterns:

- p. Question: q? Answer: __
- p. Based on the previous passage, q? __
- Based on the following passage, q? ... p

We define two verbalizers mapping questions containing a true statement to yes/true and others to no/false, respectively, for a total of 6 PVPs.

**CB** (De Marneffe et al., 2019) and **RTE** (Dagan et al., 2006) are textual entailment tasks like MNLI, so we use PVPs similar to Schick and Schütze (2020a). For a premise p and hypothesis h, we use

\[ h? \mid p \ , \ "h)? \mid p\ , \ h? \mid p \ , \ "h)? \mid p\ ]

and a verbalizer that maps entailment to yes, disagreement to no and neutral to maybe.

Given a premise p, the task in **COPA** (Roemmele et al., 2011) is to determine the cause or effect of the premise given two options c1 and c2. For determining the effect, we use the following patterns:

\[ "c1" \ or \ "c2" \ ? \ p, \ so \ ... \ , \ c1 \ or \ c2 \ ? \ p, \ so \ ... \]

For determining the cause, we use the same patterns but replace so with because. The verbalizer for c1 and c2 is the identity function.

For **WiC** (Pilehvar and Camacho-Collados, 2019), given a word w and two sentences s1 and s2 in which it occurs, the task is to decide if w is used with the same sense in both sentences. We use:

- \[ "s1" / "s2". \ Similar sense of "w"? __
- s1 \ s2 Does w have the same meaning in both sentences? ...
- w. Sense (1) (a) "s1" (... ) "s2"

For the first two patterns, we use yes as verbalization for words used in the same sense and no for other words; for the third pattern, we use b and 2.

For **WSC** (Levesque et al., 2011), each example consists of a sentence s with a marked pronoun p and noun n, and the task is to determine whether p refers to n. We follow previous work (Raffel et al., 2019; Brown et al., 2020) and treat WSC as a generative task. We highlight p in s by putting it in asterisks and use the following patterns:

- s The pronoun ‘*p*’ refers to ...
- s In the previous sentence, the pronoun ‘*p*’ refers to ...
- s In the passage above, what does the pronoun ‘*p*’ refer to? Answer: __

We use the identity function as verbalizer for n. Note that WSC is different from other tasks in that it requires free-form completion. This in turn requires some modifications during training and inference that are discussed in Appendix A.

**MultiRC** (Khashabi et al., 2018) is a QA task. Given a passage p, a question q and an answer candidate a, the task is to decide whether a is a correct answer for q. We use the same verbalizer as for BoolQ and similar patterns:

- p. Question: q? Is it a? __
- p. Question: q? Is the correct answer “a”? __
- p. Based on the previous passage, q? Is “a” a correct answer? __

For **ReCoRD** (Zhang et al., 2018), given a passage p and a cloze question q, the task is to decide which
of a given set of answer candidates is the correct replacement for the placeholder in the cloze question. As this task is already presented in the form of a cloze question, there is little room for designing PVPs, so we only use a trivial one: the concatenation of \( p \) and \( q \) as pattern and the identity function as verbalizer. With only one PVP, there is no need to perform knowledge distillation so we directly use the resulting model as our final classifier.

4.2 Setup

As underlying LM for PET we choose ALBERT-\texttt{xlarge-v2} (\cite{lan2020albert}), the best-performing MLM on SuperGLUE when training is performed on the regular, full size training sets.\footnote{Results for various models can be found at \url{https://super.gluebenchmark.com/leaderboard}.} We run PET on the FewGLUE training sets for all SuperGLUE tasks using the exact same setup as \cite{schick2020few}. For COPA, WSC and ReCoRD, we use our proposed modification of PET to support verbalizers mapping labels to multiple tokens; for all other tasks, we use regular PET. We train iPET on all tasks except COPA and WSC, as their unlabeled sets contain well below 1000 examples, as well as ReCoRD, for which iPET makes no sense as we only use a single PVP. For these three tasks, we simply reuse the results of regular PET.

4.3 Results

Our main results are shown in Table 1. As can be seen, ALBERT with PET performs similar to the largest GPT-3 model, which is larger by a factor of 785. On average, PET performs 18 points better compared to GPT-3 Med, a model of similar size. iPET brings further improvements for 3 out of 5 tasks, most notably for CB, but results in a slight performance drop for MultiRC. Similar to GPT-3, we find that PET does not perform above random chance for WiC, which is difficult to reformulate as a language modeling task. ReCoRD is the only task for which GPT-3 consistently performs better than both PET and iPET. Despite PET’s strong performance, it still clearly performs worse than a state-of-the-art model trained on the regular, full size SuperGLUE training set.

5 Analysis

We investigate the importance of several factors for good few-shot performance: the choice of patterns and verbalizers, the usage of both unlabeled and labeled data, and properties of the underlying language model. We also look into our proposed modification for PET to work with multiple masks and compare it to various baselines. Finally, we measure how choosing different sets of training examples affects performance. Our analysis focuses on PET as GPT-3 is not publicly available.\footnote{We requested access to OpenAI’s GPT-3 API, but as of this writing, it has not been granted.}

5.1 Patterns

One factor that has gained little attention in previous work is the way tasks are reformulated as cloze questions. These reformulations can be arbitrarily complex; for example, the pattern used by PET for WSC contains an introductory section...
that spans almost 30 words; it is unclear if and how
this formulation has been optimized. To inves-
tigate the importance of patterns and verbalizers, we
compare three different sets of PVPs: our initial set
of PVPs as defined in Section 4.1 (denoted pours),
the single PVP used by GPT-3 (pGPT-3), and the
combination of both (pcomb).
We train ALBERT using PET with all three sets
of patterns; results for selected SuperGLUE tasks
are shown in Table 2 (top). As can be seen, the
PVP used by GPT-3 outperforms our PVPs on RTE
whereas our initial set of patterns performs much
better on MultiRC. These large differences in per-
formance highlight the importance of finding good
ways to express tasks as cloze questions. As it
is extremely difficult to ascertain which patterns
perform well without trying them on a large set of
examples, a key challenge for few-shot approaches
is to be able to compensate for PVPs that the used
LM fails to understand well. As seen in the per-
formance of the model trained with pcomb, PET is
able to do so: not only does combining all PVPs
compensate for the worse performance of pours on
RTE and of pGPT-3 on MultiRC, it even further
improves average performance across the three tasks
considered compared to the best-performing set of
patterns. This clearly demonstrates the potential of
carefully engineering a set of suitable patterns as
opposed to just choosing a single formulation with-
out means of evaluating its actual effectiveness.

5.2 Unlabeled Data Usage
Unlike GPT-3, PET requires unlabeled data to dis-
till the knowledge of all models based on individual
PVPs into a single classifier; for iPET, unlabeled
data is additionally used to generate training sets
for future generations. While the benefit of iPET
is already analyzed in Table 1, we now investigate
the importance of unlabeled data for regular PET.
To this end, we compare the performance of the
final classifier in PET to that of directly using the
ensemble of models corresponding to individual
PVPs. While using this ensemble entirely removes
the need for unlabeled data, the ensemble for k
PVPs is larger than the distilled model by a factor
of 3 · k as we follow the default setting of PET and
train three models per PVP. However, even for a
large number of PVPs the ensemble is smaller than
GPT-3 by two orders of magnitude.

Results without distillation can be seen in Ta-
ble 2 (bottom). Averaged across the three tasks, the
ensemble performs even better than the distilled
classifier. This shows that if the goal is only to
achieve good performance, then unlabeled data is
not necessary; however, it is required to obtain a
single, lightweight model as final classifier.

Of course, there are further ways to lever-
age unlabeled data such as keeping an auxiliary
language modeling objective during finetuning
(Chronopoulou et al., 2019). While we leave in-
vestigating the impact of additionally using such
methods to future work, we note that they can easily
be applied to PET while there is no straightforward
way to combine them with priming.

5.3 Labeled Data Usage
We next investigate the effect of how labeled data
is used, which is one of the key differences be-
tween priming and PET. We first compare PET
with regular supervised training (i.e., without using
any patterns), and with a fully unsupervised model
(Table 3). Given 32 examples, PET clearly outper-
forms both baselines, which is in line with findings
by Schick and Schütze (2020a).
We next compare PET directly to priming. How-
ever, we cannot do so using ALBERT as it, like
most pretrained MLMs, is only able to process se-
quencies of up to 512 tokens, which is not enough
for a set of 32 examples. As it theoretically sup-
ports sequences of arbitrary length, we instead
use XLNet (Yang et al., 2019) for this compari-

| Model       | CB Acc. / F1 | RTE Acc. | MultiRC EM / F1a | Avg.  |
|-------------|--------------|----------|------------------|-------|
| PET (pours) | 85.1 / 59.4  | 69.8     | 37.9 / 77.3      | 66.6  |
| PET (pGPT-3)| 83.3 / 58.1  | 71.8     | 25.4 / 68.3      | 63.1  |
| PET (pcomb) | 84.5 / 59.0  | 74.7     | 39.1 / 77.7      | 68.3  |
| PET (pours) - dist | 83.9 / 76.2 | 66.4 | 38.9 / 76.2 | 68.0 |
| PET (pcomb) - dist | 83.9 / 76.2 | 72.9 | 39.6 / 76.6 | 70.4 |

Table 2: Results on selected tasks for various sets of PVPs for regular PET and for an ensemble of PET models with no knowledge distillation (“- dist”)

| Model       | CB Acc. / F1 | RTE Acc. | MultiRC EM / F1a | Avg.  |
|-------------|--------------|----------|------------------|-------|
| PET         | 85.1 / 59.4  | 69.8     | 37.9 / 77.3      | 66.6  |
| unsupervised| 33.5 / 23.1  | 55.0     | 3.9 / 60.3       | 38.5  |
| supervised  | 60.7 / 42.5  | 50.2     | 4.3 / 49.8       | 43.0  |
| PET (XLNet) | 88.7 / 83.0  | 60.4     | 21.4 / 66.6      | 63.4  |

Table 3: Results on selected tasks for various ways of using the labeled examples available in FewGLUE
Figure 3: Accuracy differences between priming with 32 examples and one-shot priming for all GPT-3 models as well as between ALBERT with PET (without distillation) and unsupervised ALBERT (bottom row).

As shown in Table 3, XLNet in general performs worse than ALBERT. More importantly, XLNet with PET performs much better than priming (and priming does not even perform above random chance for RTE). We were not able to obtain results with priming on MultiRC because the 32 examples in FewGLUE would require more than 10,000 tokens, so processing them with a standard Transformer (Vaswani et al., 2017) is infeasible due to the quadratic complexity of self-attention. This highlights another important issue with priming: It does not scale well to more than a few examples; even GPT-3 is only able to process sequences of up to 2,048 tokens. While there are some variants of the Transformer that can deal with much longer contexts (e.g., Kitaev et al., 2020; Beltagy et al., 2020), it has yet to be investigated to what extent such models are able to actually make good use of priming examples over such a long context span.

We further investigate the effectiveness of priming by looking at results obtained with GPT-3 more closely. To analyze how well GPT-3 is able to make use of examples given as context, Figure 3 shows the performance difference between priming with 32 examples and priming with just a single example for each task and model size.\(^7\) As can be seen, priming with 32 examples only slightly improves performance for most tasks and model sizes. For some tasks, adding more examples even leads to worse performance, especially for smaller models. For ReCoRD, even the largest model’s performance slightly drops when adding more examples.

The bottom row of Figure 3 shows the performance difference between ALBERT trained with PET (without distillation) and a fully unsupervised ALBERT model on all tasks. While results are not directly comparable due to different underlying models and PVPs, using PET results in much stronger performance improvements compared to priming and does not worsen results for any of the eight tasks.

Table 4: Results on selected tasks for PET without knowledge distillation combined with various LMs using \(p_{GPT-3}\) for CB/RTE and \(p_{ours}\) for MultiRC.

| Model      | Params | CB Acc. / F1 | RTE EM / F1a | MultiRC | Avg  |
|------------|--------|--------------|--------------|---------|------|
| ALBERT     | 223M   | 87.5 / 78.7  | 74.7         | 38.9 / 76.2 | 71.8 |
| RoBERTa    | 355M   | 85.7 / 77.5  | 62.8         | 23.3 / 70.0 | 63.7 |
| GPT-2      | 345M   | 73.2 / 73.7  | 47.7         | 12.4 / 57.4 | 52.0 |

\(^7\) We do not compare priming to zero-shot performance as for unknown reasons, zero-shot GPT-3 performs well below random guessing for some tasks (e.g., 0.0% accuracy for WIC). To not overestimate the benefit of priming, we therefore show gains from providing 32 examples compared to just one.

5.4 Model Type

We next look into the relevance of the underlying LM for PET’s performance. To this end, we compare ALBERT with RoBERTa large (Liu et al., 2019) and GPT-2 medium (Radford et al., 2019). As GPT-2 is a unidirectional model similar to GPT-3, it can only process patterns where the mask token is the very last token. We therefore use \(p_{GPT-3}\) for CB and RTE; for MultiRC, we stick with our original set of patterns as they already fulfill this requirement. We also do not perform distillation and instead report the ensemble’s performance as there is no established way of equipping GPT-2 with a sequence classification head.

Results for training all three LMs with PET in Table 4 show that using ALBERT as underlying LM is crucial for PET’s strong performance; exchanging ALBERT with RoBERTa results in an average performance drop of 8 points. However, RoBERTa still clearly outperforms GPT-3 13B, which is larger by two orders of magnitude. Importantly, PET with GPT-2 performs much worse than with both other models. As anticipated by previous work (Brown et al., 2020), a key reason for this drop in performance could be that like GPT-3, GPT-2 is a unidirectional LM, making tasks that require comparing two sequences of text a much greater challenge. However, it is important to note that there are also other substantial differences between

---

Figure 3: Accuracy differences between priming with 32 examples and one-shot priming for all GPT-3 models as well as between ALBERT with PET (without distillation) and unsupervised ALBERT (bottom row).

---

Table 4: Results on selected tasks for PET without knowledge distillation combined with various LMs using \(p_{GPT-3}\) for CB/RTE and \(p_{ours}\) for MultiRC.

| Model      | Params | CB Acc. / F1 | RTE EM / F1a | MultiRC | Avg  |
|------------|--------|--------------|--------------|---------|------|
| ALBERT     | 223M   | 87.5 / 78.7  | 74.7         | 38.9 / 76.2 | 71.8 |
| RoBERTa    | 355M   | 85.7 / 77.5  | 62.8         | 23.3 / 70.0 | 63.7 |
| GPT-2      | 345M   | 73.2 / 73.7  | 47.7         | 12.4 / 57.4 | 52.0 |

\(^7\) We do not compare priming to zero-shot performance as for unknown reasons, zero-shot GPT-3 performs well below random guessing for some tasks (e.g., 0.0% accuracy for WIC). To not overestimate the benefit of priming, we therefore show gains from providing 32 examples compared to just one.
We compare our decoding strategy of predicting to-

We modified P

max-first

which we refer as

which this is required: COPA, WSC and ReCoRD.

Table 5: Results on selected tasks for our proposed vari-

on different seeds. This allows us to investigate

condition of SuperGLUE,

Σ

FewGLUE. In this subsection, we create two ad-

set of SuperGLUE for random seed

Σ

i

0

be the randomly selected sub-

i

= 0

0

2

1

= 0

Σ

1

Σ

2

ET

5.5 P

PET with Multiple Masks

We modified PET to work for outputs that require

more than a single token. To investigate the impact

of this modification, we look at the three tasks for

which this is required: COPA, WSC and ReCoRD.

We compare our decoding strategy of predicting to-

kens in order of the probability assigned to them, to

which we refer as max-first, with two alternatives:

decoding left-to-right (ltr) as is common for many

autoregressive language models, and decoding all

tokens simultaneously (parallel) as is done during

training. Additionally, we compare PET with un-

trained ALBERT to measure the effectiveness of

our proposed training loss.

Results are shown in Table 5. PET clearly out-

performs untrained ALBERT for the three tasks.

Not performing distillation hurts performance for

COPA, but leads to slight improvements on WSC.

Our decoding strategy is clearly superior to par-

allel decoding except for WSC, for which most

predictions consist only of one or two tokens, and

performs slightly better than left-to-right decoding.

5.6 Training Examples

Recall that we conduct our experiments with training

examples from FewGLUE, a randomly selected

subset of the original SuperGLUE training ex-

amples. We used a fixed random seed s₀ to generate

FewGLUE. Let Σ₁ be the randomly selected sub-

set of SuperGLUE for random seed s₁, so Σ₂ =

FewGLUE. In this subsection, we create two ad-

tional subsets of SuperGLUE, Σ₁ and Σ₂, based

on different seeds. This allows us to investigate

how different sets of training examples affect per-

formance. To this end, we run PET for CB, RTE

and MultiRC using all three Σᵢ. To measure only

the effect of varying the training set while ignoring

unlabeled examples, we do not use distillation.

Results in Table 6 show that for all tasks, chang-

ing the set of training examples can result in sig-

nificant performance differences for PET. This

highlights the importance of using the same set

of examples when comparing different few-shot

approaches, which is why we make the particular

set of examples in FewGLUE publicly available.

However, we note that the average performance of

PET is similar to that of GPT-3 for all seeds.

While our results may seem contrary to the in-

sight that for GPT-3, the exact choice of examples
does not play a major role, we suspect this to be
due to the fact that priming benefits much less from
training examples than PET (cf. Section 5.3); ac-
cordingly, the influence of the exact set of training
examples on the model’s performance is smaller.

6 Conclusion

We have shown that it is possible to achieve few-

shot performance similar to GPT-3 on SuperGLUE

with LMs that have three orders of magnitude fewer

parameters. This is achieved using PET, a method

that reformulates tasks as cloze questions and trains

an ensemble of models for different reformulations.

We have proposed a simple yet effective modifi-
cation to PET that enables us to use it for tasks

that require predicting multiple tokens. In exten-
sive experiments, we have identified several factors
responsible for the strong performance of PET com-

bined with pretrained ALBERT: the possibility to

concurrently use multiple patterns for transforming
examples into cloze questions, the ability to com-

pensate for patterns that are difficult to understand,

the usage of labeled data to perform parameter up-
dates, and the underlying LM itself. To enable

comparisons with our work, we make our dataset

of few-shot training examples publicly available.
References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. Computing Research Repository, arXiv:2004.05150.

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

Alexandra Chronopoulou, Christos Bazioti, and Alexandros Potamianos. 2019. An embarrassingly simple approach for transfer learning from pre-trained language models. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 2089–2095, Minneapolis, Minnesota. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Proceedings of NAACL-HLT 2019.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In Machine learning challenges, evaluating predictive uncertainty, visual object classification, and recognising textual entailment, pages 177–190. Springer.

Joe Davison, Joshua Feldman, and Alexander Rush. 2019. Commonsense knowledge mining from pre-trained 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), pages 1173–1178, Hong Kong, China. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The CommitmentBank: Investigating projection in naturally occurring discourse. In Proceedings of Sinn und Bedeutung 23.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Urün Dogan, Tobias Glasmachers, and Christian Igel. 2016. A unified view on multi-class support vector classification. J. Mach. Learn. Res., 17(45):1–32.

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. Transactions of the Association for Computational Linguistics, 8:3448.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language 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), pages 6112–6121, Hong Kong, China. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. Computing Research Repository, arXiv:1503.02531.

Zhenzhao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2019. How can we know what language models know? Computing Research Repository, arXiv:1911.12543.

Daniel Khoshabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 252–262.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In International Conference on Learning Representations.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In International Conference on Learning Representations.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge. In AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning, volume 46, page 47.

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

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 pre-training approach. Computing Research Repository, arXiv:1907.11692.

Juri Opitz. 2019. Argumentative relation classification as plausibility ranking. In Preliminary proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers, pages 193–202, Erlangen, Germany. German Society for Computational Linguistics & Language Technology.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In NIPS Autodiff Workshop.

Fabio Petroni, Tim Rocktschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: The word-in-context dataset for evaluating context-sensitive meaning representations. In Proceedings of NAACL-HLT.

Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. Computing Research Repository, arXiv:1912.10165.

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

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. Computing Research Repository, arXiv:1910.10683.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In 2011 AAAI Spring Symposium Series.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrina Kirchoff. 2020. Masked language model scoring. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2699–2712, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2019. Learning semantic representations for novel words: Leveraging both form and context. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence.

Timo Schick and Hinrich Schütze. 2020a. Exploiting cloze questions for few shot text classification and natural language inference. Computing Research Repository, arXiv:2001.07676.

Timo Schick and Hinrich Schütze. 2020b. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2019. oLMpics – on what language model pre-training captures. Computing Research Repository, arXiv:1912.13283.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc.

Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a Markov random field language model. In Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation, pages 30–36, Minneapolis, Minnesota. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. arXiv preprint 1905.00537.

Jason Weston and Chris Watkins. 1999. Support vector machines for multi-class pattern recognition. In ESANN, volume 99, pages 219–224.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick Cistac, Tim Rault, Rni Louf, Morgan Funtowicz, and Jamie Brew. 2019. Transformers: State-of-the-art natural language processing. Computing Research Repository, arXiv:1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc,
Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. ReCoRD: Bridging the gap between human and machine commonsense reading comprehension. Computing Research Repository, arXiv:1810.12885.

A Training Details

Our implementation extends the original implementation of PET by Schick and Schütze (2020a) which, in turn, is based on the Transformers library (Wolf et al., 2019) and PyTorch (Paszke et al., 2017). Unless explicitly stated differently, we use the exact same set of hyperparameters as Schick and Schütze (2020a) with the only difference that for iPET, we only train 3 generations of models to speed up training. Below, we list task-specific implementation details for all tasks in SuperGLUE.

COPA For COPA, we randomly switch the two options \(c_1\) and \(c_2\) during training with a probability of 50% to make the input more diverse; for inference, we always keep the original order. For distilling the final PET model, we obtain logits for unlabeled examples \(x\) from individual PVPs \(p\) as 
\[ s_p(y \mid x) = \log q_p(y \mid x); \]
we use the input format proposed by Liu et al. (2019).

WiC Similar to COPA, we randomly switch the input sentences \(s_1\) and \(s_2\) during training. Given a word \(w\) and two sentences \(s_1\) and \(s_2\), we use the sequence \(w: s_1 \mid s_2\) as input for the final sequence classification model, where \(\mid\) marks the boundary between two text segments.

WSC Unlike other SuperGLUE tasks, the WSC formulation of Raffel et al. (2019) and Brown et al. (2020) requires free-form completion, meaning that for each sentence \(s\) and pronoun \(p\), we only have a single correct choice \(n\) that the model needs to predict, but we do not provide any alternatives. During training, we thus use regular cross entropy loss between \(n\) and \(\tilde{q}_p(n \mid s, p)\) as defined in Eq. 4. However, in many cases this would allow the LM to easily identify the correct target based on the number of masks provided, so we modify each target by randomly adding up to three additional mask tokens, for which we require the model to predict a special \(<\text{pad}>\) token. For inference, we always just add a single mask token to ensure consistent results across multiple evaluations and perform greedy decoding as described in Section 3.

We then follow Raffel et al. (2019) to map the output produced by the LM to a label \(y \in \{\text{true}, \text{false}\}\). For distillation, given an unlabeled example \(x\) we set \(s_p(y \mid x) = 1\) if the model’s output for \(x\) was mapped to \(y\) and \(s_p(y \mid x) = 0\) otherwise. We provide inputs to the final PET model in the format \(s \mid n\) where \(\mid\) is the boundary between two text segments and mark \(p\) in \(s\) with asterisks.

MultiRC Deviating from the hyperparameters used by Schick and Schütze (2019), we use a maximum sequence length of 512 tokens for MultiRC both during training and inference because we found many passages to be much longer than 256 tokens. Input for the final sequence classification model is of the form \(p \mid q \mid a\) where \(p\) is the passage, \(q\) is the question, \(a\) is the answer candidate and we use \(\mid\) to mark boundaries between text segments.

ReCoRD For ReCoRD, we again use a maximum sequence length of 512 because many passages require more than 256 tokens. For some questions \(q\), the ReCoRD training set contains a huge number of answer candidates. To facilitate training, we split each example into multiple examples as follows: let \(C\) be the set of answer candidates with \(C^+ \subset C\) being the set of correct answers. We create a training example for each \(c \in C^+\) by randomly selecting up to 9 negative examples from \(C \setminus C^+\) for a total of 10 answer candidates.