True Few-Shot Learning with Prompts—A Real-World Perspective

Timo Schick and Hinrich Schütze
Center for Information and Language Processing (CIS), LMU Munich, Germany
schickt@cis.lmu.de, inquiries@cislmu.org

Abstract

Prompt-based approaches excel at few-shot learning. However, Perez et al. (2021) recently cast doubt on their performance as they had difficulty getting good results in a ''true'' few-shot setting in which prompts and hyperparameters cannot be tuned on a dev set. In view of this, we conduct an extensive study of PET, a method that combines textual instructions with example-based finetuning. We show that, if correctly configured, PET performs strongly in true few-shot settings without a dev set. Crucial for this strong performance is a number of design choices, including PET's ability to intelligently handle multiple prompts. We put our findings to a real-world test by running PET on RAFT, a benchmark of tasks taken from realistic NLP applications for which no labeled dev or test sets are available. PET achieves a new state of the art on RAFT and performs close to non-expert humans for 7 out of 11 tasks. These results demonstrate that prompt-based learners can successfully be applied in true few-shot settings and underpin our belief that learning from instructions will play an important role on the path towards human-like few-shot learning capabilities.

1 Introduction

With pretrained language models (LMs) getting ever larger (Radford et al., 2019; Raffel et al., 2020; Brown et al., 2020; Fedus et al., 2021), *instruction-based learning* is a powerful method for few-shot text classification (e.g., Schick and Schütze, 2020; Jiang et al., 2020; Schick and Schütze, 2021; Brown et al., 2020; Wei et al., 2022; Sanh et al., 2022). The key idea is to give an LM access to descriptive names for all possible outputs and to short prompts explaining the task to be solved. In settings where at most a few dozen examples are available, this simple idea leads to substantial improvements over other approaches (Schick and Schütze, 2020, 2021; Gao et al., 2021; Tam et al., 2021).

However, recent work has questioned the strong few-shot performance of instruction-based approaches, arguing that they are evaluated in scenarios that are not *true* few-shot settings (Perez et al., 2021; Logan IV et al., 2021), mainly for two reasons. First, some approaches (e.g., Xie et al., 2019; Zhang et al., 2020; Chen et al., 2020; Tam et al., 2021) make use of large development sets to optimize hyperparameters. Second, it is argued that manually designed instructions require manual tuning on development sets to achieve strong performance (Perez et al., 2021; Logan IV et al., 2021). Indeed, performance can vary greatly—and in mostly unpredictable ways—across different instructions (Jiang et al., 2020; Schick and Schütze, 2020); this issue even persists after finetuning on hundreds of instructions (Sanh et al., 2022). More generally, the need for human involvement is seen as a serious drawback of manually designed instructions (Shin et al., 2020; Lester et al., 2021). Thus, several recent studies have abandoned them in favor of automatically generated prompts (Shin et al., 2020; Gao et al., 2021; Hambardzumyan et al., 2021; Li and Liang, 2021; Lester et al., 2021).

Contrary to this trend, we argue that when correctly configured, prompt-based approaches achieve strong performance even in true few-shot settings and that there is no problem with using manually designed instructions. Quite the opposite: Such instructions are often easy to specify if one is familiar with the task to be solved, they provide an intuitive interface to convey task-specific knowledge, and, if properly used, they can considerably improve model performance in few-shot settings.

To provide empirical support for these claims, we revisit PET (Schick and Schütze, 2020), a method for combining instructions with example-based finetuning, and thoroughly examine its performance with human-made instructions in true few-shot settings. We simulate a real-world scenario by proceeding in two steps: First, we conduct

Transactions of the Association for Computational Linguistics, vol. 10, pp. 716–731, 2022. https://doi.org/10.1162/tacl_a_00485
Action Editor: Alexander Rush. Submission batch: 12/2021; Revision batch: 3/2022; Published 6/2022.

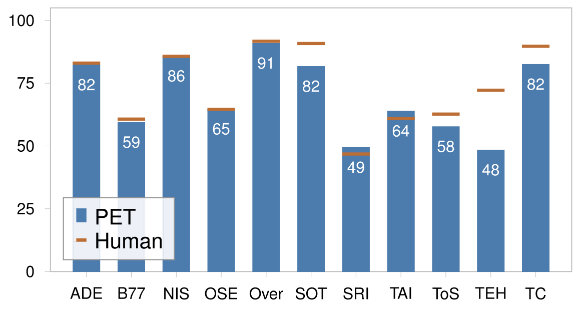

Figure 1: PET achieves near-human performance for 7 out of 11 tasks of the RAFT benchmark (Alex et al., 2021), for which labeled dev and test sets are not available. This demonstrates the potential of prompt-based learners for few-shot learning in ‘‘true’’ real-world settings, i.e., without any tuning of instructions or hyperparameters on a dev set.

an extensive study of PET using three academic datasets to analyze its ability to perform true few-shot learning in a controlled environment and derive best practices for the choice of instructions and hyperparameters. We then put our findings to the test and evaluate PET on a large variety of real-world tasks from the RAFT benchmark (Alex et al., 2021), for which no labeled dev or test sets are available, enforcing a *true* few-shot setting (Perez et al., 2021). On average, PET clearly outperforms all baselines on this dataset and comes surprisingly close to non-expert human performance (see Figure 1). This demonstrates that instruction-based learning can successfully be applied to real-world tasks in true few-shot settings.

Our main contributions are as follows:

- We investigate the performance of PET for various models, tasks, and training set sizes, its ability to cope with different instructions, and its robustness to hyperparameter choices in true few-shot settings.
- We show how PET can be used when no unlabeled data is available and propose a method for efficient classification in scenarios with many different classes, addressing two frequent real-world scenarios.
- We apply PET to RAFT (Alex et al., 2021), a benchmark of real-world tasks. PET obtains a new state of the art and achieves near-human performance for 7 out of 11 tasks in a true few-shot setting.

2 Related Work

As a precursor to instruction-based learning, some studies have investigated ways of informing classifiers about the meaning of different output classes both for text (Chang et al., 2008; Veeranna et al., 2016; Zhou et al., 2018) and image classification (Norouzi et al., 2014; Romera-Paredes and Torr, 2015); providing instructions in the form of short prompts was first proposed by Radford et al. (2019). This idea has since been applied to solve a wide range of NLP tasks without any task-specific training data (Puri and Catanzaro, 2019; Opitz, 2019; Davison et al., 2019; Schick et al., 2021; Schick and Schütze, 2021; Wei et al., 2022; Sanh et al., 2022). While most approaches rephrase tasks as a language modeling problem, some use prompts to reformulate them as different tasks for which large amounts of training data are available (Levy et al., 2017; McCann et al., 2018; Yin et al., 2019; Sun et al., 2021; Sainz et al., 2021). Instruction-based learning has also been used in few-shot settings; popular variants include *in-context learning*, where the model’s parameters are fixed and examples are provided as additional context (Brown et al., 2020; Lu et al., 2021; Kumar and Talukdar, 2021; Zhao et al., 2021; Min et al., 2021), *finetuning* the entire model (Schick and Schütze, 2020, 2021; Gao et al., 2021; Tam et al., 2021), and *prompt tuning*, where only the instruction itself is optimized (Shin et al., 2020; Hambardzumyan et al., 2021; Li and Liang, 2021; Lester et al., 2021).

Several works investigating the limitations of instruction-based few-shot approaches find that current LMs are mostly unable to understand complex instructions that go beyond short prompts or simple questions (Efrat and Levy, 2020; Weller et al., 2020; Webson and Pavlick, 2021) and that they are highly sensitive to the exact wording of the instructions provided (Jiang et al., 2020; Schick and Schütze, 2020; Chu et al., 2021; Elazar et al., 2021). In a similar vein, Perez et al. (2021) and Logan IV et al. (2021) argue that prior work overestimates few-shot performance as manual prompt tuning is required to achieve good performance. Accordingly, some studies attempt to obtain either prompts (Shin et al., 2020; Gao et al., 2021; Li and Liang, 2021; Lester et al., 2021) or meaningful names for output classes (Schick et al., 2020; Gao et al., 2021) without human involvement.

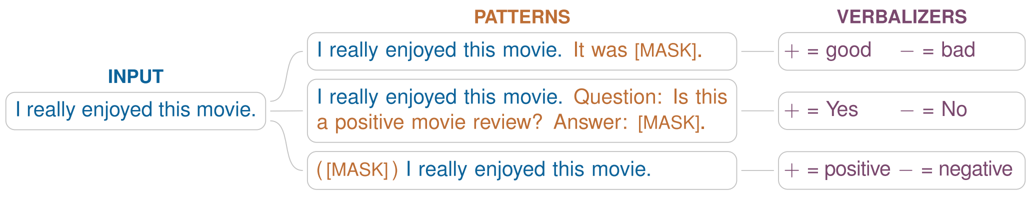
PATTERNS
VERBALIZERS
INPUT
I really enjoyed this movie.
I really enjoyed this movie. It was [MASK].
I really enjoyed this movie. Question: Is this a positive movie review? Answer: [MASK].
([MASK]) I really enjoyed this movie.
+ = good − = bad
+ = Yes − = No
+ = positive − = negative

Figure 2: Different choices of patterns and corresponding verbalizers for classifying movie reviews as *positive* (+) or *negative* (−). The input is first converted into a cloze question using the pattern; classification is done by computing the output whose verbalization is, according to the MLM, the most likely substitute for the mask.

Finally, many benchmarks have been proposed for comparing few-shot approaches in a standardized way (e.g., Mishra et al., 2021; Bragg et al., 2021; Xu et al., 2021; Ye et al., 2021; Alex et al., 2021). As our focus is on the real-world applicability of few-shot methods, we evaluate PET on the RAFT benchmark (Alex et al., 2021), which measures performance in applied settings.

3 Pattern-Exploiting Training

We briefly review *pattern-exploiting training* (PET) (Schick and Schütze, 2020, 2021), the method we use for instruction-based text classification. At its core, PET combines textual instructions with regular finetuning using labeled examples. To that end, users must specify one or more *patterns* that convert an input example x into a cloze question so that it can readily be processed by a masked language model (MLM) (Devlin et al., 2019).[1] These patterns can take on very different forms; some examples are shown in Figure 2. In addition, users must inform the model about the meaning of all output classes; this is done with a *verbalizer* that assigns a natural language expression to each output y (see Figure 2, right). We refer to the combination of a pattern and verbalizer as a *pattern-verbalizer pair* (PVP).

Given a single PVP, let $p(y \mid x)$ be the probability that an MLM assigns to y's verbalization in the cloze question obtained by applying the pattern to x, normalized over all y. The MLM is finetuned on labeled examples (x, y) by minimizing the cross-entropy loss between $p(y \mid x)$ and a distribution that assigns a probability of 1.0 to y.

If a user specifies *multiple* PVPs, individual models are trained for each pair. Similar to knowledge distillation (Hinton et al., 2015), they are then used to annotate unlabeled examples for training a final classifier with a regular sequence classification head (Devlin et al., 2019). We use the *weighted* variant of PET without auxiliary language modeling; see Schick and Schütze (2020) for details.

[1]We use the term *prompt* to refer to a short sequence of tokens that typically contains some form of instruction; *pattern* is used to denote the function that adds a prompt to an input.

4 True Few-Shot Learning with PET

After describing our experimental setup, we conduct experiments on academic datasets to answer 6 important research questions (Q1–Q6) on the extent to which true few-shot learning is possible with PET. The purpose of our experiments is also to establish best practices for real-world scenarios and experiments on RAFT (Alex et al., 2021).

Tasks and Datasets While they were heavily used in prior work (e.g., Brown et al., 2020; Schick and Schütze, 2021; Logan IV et al., 2021; Webson and Pavlick, 2021), we decide against tasks and datasets from GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) as they are different from what we expect to see in real-world applications. Instead, we experiment with AG's News, Yelp Reviews Full Star, and Yahoo Questions (Zhang et al., 2015) as these datasets represent classification tasks in three different domains that resemble real-world settings. We create a broad variety of instructions for each task to be able to experiment with a large number of different patterns.

We consider settings with $n = 10$ and $n = 100$ training examples. For each n, we generate five different training sets per task by randomly sampling examples from the original training set while ensuring that the number of examples is about the same for each possible output class. In addition, for both $n = 10$ and $n = 100$, we sample 1,000 unlabeled examples from the original training set. We repeat all of our experiments for all five training sets and, by default, report average performance.

PVPs We manually write a total of 23 patterns per task, all of which can be categorized into one of the following groups:[2]

- NULL: Following Logan IV et al. (2021), these patterns simply insert a mask token.
- PUNC: Similar to some patterns of Schick and Schütze (2020), these patterns only add punctuation characters and a mask token.
- PROMPTS: Patterns in this group add short prompts—typically consisting of no more than three words—to the input, similar to Radford et al. (2019) and Schick and Schütze (2020).
- Q&A: These patterns rephrase the task as a question q and append

  Question: q Answer: [MASK].

  to the input, similar to Brown et al. (2020) and Schick et al. (2021).

For all patterns, we use a single verbalizer, adopted from Schick and Schütze (2020). There is often a single natural choice for the verbalizer (e.g., the category names for AG's News / Yahoo Questions), so finding many good verbalizers is challenging.

Hyperparameters We consider a setting similar to that of Schick and Schütze (2020) and Schick and Schütze (2021) and, unless otherwise specified, use the default settings of the PET library.[3] As our experiments require training hundreds of models, we make a few changes to reduce environmental impact (Strubell et al., 2019) and computational cost: We use the base variant of RoBERTa (Liu et al., 2019) as underlying LM both for individual models and the final classifier, we train only one model per PVP, and we reduce the training steps for all individual models and the final classifier to 100 and 1,000, respectively.

Monitoring Finetuning LMs on small datasets is unstable (Devlin et al., 2019; Dodge et al., 2020) and sometimes results in poor performance. We aim to detect failed finetuning without a labeled test set using the following two checks:

- TRAIN SET UNDERFITTING: We check for training runs that result in less than 50% accuracy on the training set. As finetuning on up to 100 examples typically leads to perfect predictions on the training set, this is a clear indicator of failed finetuning.
- CONSTANT PREDICTIONS: We check for training runs that result in the same class being predicted for all inputs, both on the training set and the unlabeled set. Again, this is a clear indicator of failed finetuning.

Whenever one of these two events occurs, we restart training using a different seed.

Q1: How can we find *the* best pattern—or do we even need to?

Slightly different patterns can have very different performance (Jiang et al., 2020; Schick and Schütze, 2020; Schick et al., 2021; Webson and Pavlick, 2021; Sanh et al., 2022, inter alia) and popular model selection criteria cannot reliably identify the best-performing patterns in few-shot settings (Perez et al., 2021). We thus investigate to what extent PET can eliminate the need to find the best instruction even in extreme settings where there are dozens of candidates to choose from.

Setup Using our default setup, we train individual models for each PVP and a final PET model; we also train models with iPET, an iterative variant of PET introduced by Schick and Schütze (2020), using 3 iterations.

Results Performance of individual models for each pattern and of the distilled models obtained using PET and iPET is shown in Figure 3. Interestingly, sorting all pattern groups by their average performance gives the exact same order for each task and training set size: NULL patterns clearly perform worst, followed by PUNC and PROMPT; Q&A gives the best average results. Contrary to findings of Logan IV et al. (2021), this shows that LMs can benefit considerably from manually written instructions even if combined with finetuning.

Crucially, PET's performance is much higher than average performance of individual patterns; further, it consistently outperforms even the best

[2]The full set of PVPs can be found at `https://github.com/timoschick/pet/tree/master/true-fsl`.

[3]See `https://github.com/timoschick/pet`.

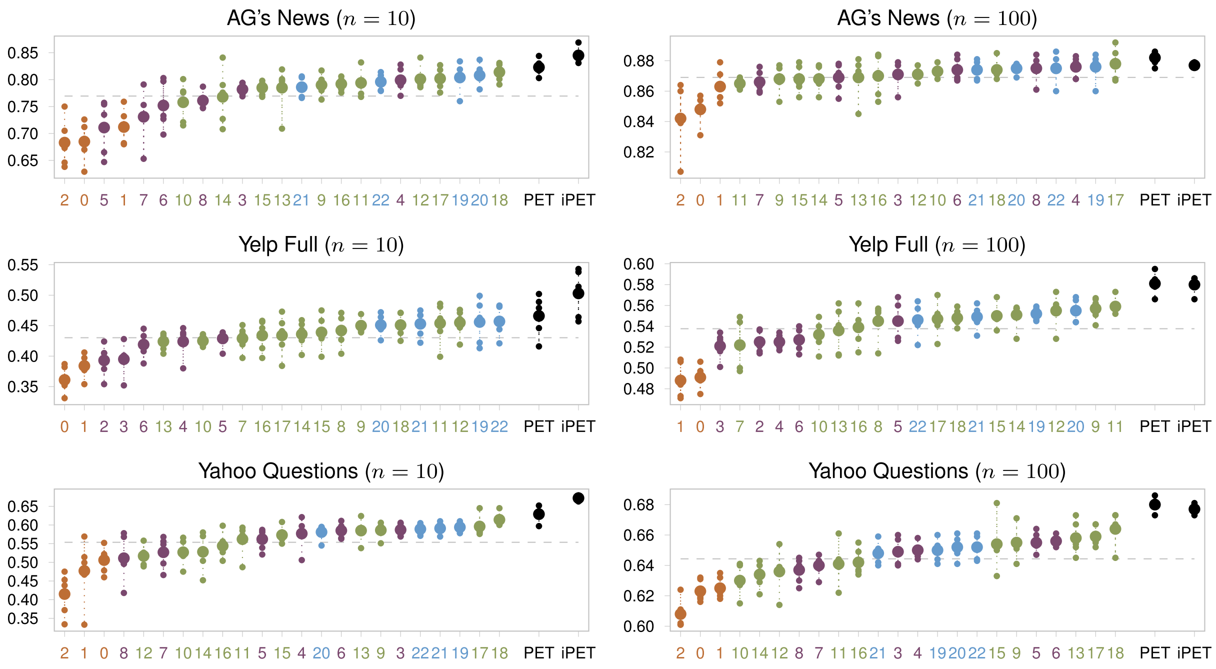

Figure 3: Performance of individual patterns, PET and iPET on all tasks considered. Accuracy is shown on the y-axis; the x-axis shows individual pattern ids where color is used to distinguish the different pattern categories (NULL, PUNC, PROMPT, Q&A). Small bullets (•) correspond to individual training sets, large bullets (●) correspond to average performance. Average performance across all patterns is shown as a dashed gray line. Q&A and PROMPT perform better than NULL and PUNC; PET consistently outperforms even the best individual pattern.

pattern, verifying that PET indeed removes the need to find *the* best pattern. While iPET gives clear improvements for $n = 10$, it performs worse than PET for $n = 100$. The reason for this may be that we use a much smaller set of unlabeled examples than prior work (Schick and Schütze, 2020, 2021).

Q2: Does performance of different patterns transfer across models?

While our results for Q1 show a consistent order of pattern groups for different training set sizes and tasks, an important question for real-world applications is whether the same finding also holds for different model sizes and entirely different models.

Setup We consider BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and ALBERT (Lan et al., 2020) as underlying LMs;[4] we experiment with the base and large variants. For each model and size, we repeat the same experiment as for Q1.

[4]For Yahoo, we do not consider BERT as it uses a vocabulary that does not assign a single token to each verbalization.

Results Figure 4 shows the performance of each pattern group (i.e., average performance of all individual patterns in this group) and PET; scores are normalized so that the best-performing approach for each task, training set size, and model gets a score of 1.0. With few exceptions, our findings from Q1 regarding the relative performance of pattern groups and PET (NULL < PUNC < PROMPT < Q&A < PET) also hold for different models and sizes. The performance of individual patterns strongly correlates between different models and sizes (Spearman's $\rho \geq 0.7$ except in one case).

Q3: Does PET still work if some PVPs are not well understood?

Q1 and Q2 show that PET performs even better than the best PVP for a large set of high-quality PVPs. But perhaps the performance is much worse if the LM fails to understand many patterns and verbalizers, for example, because they are in a style different from the model's pretraining data? For real-world scenarios, we want to know how such ‘‘bad’’ PVPs affect the performance of PET.

Setup It is difficult to obtain large quantities of bad instructions as they might occur in real-world

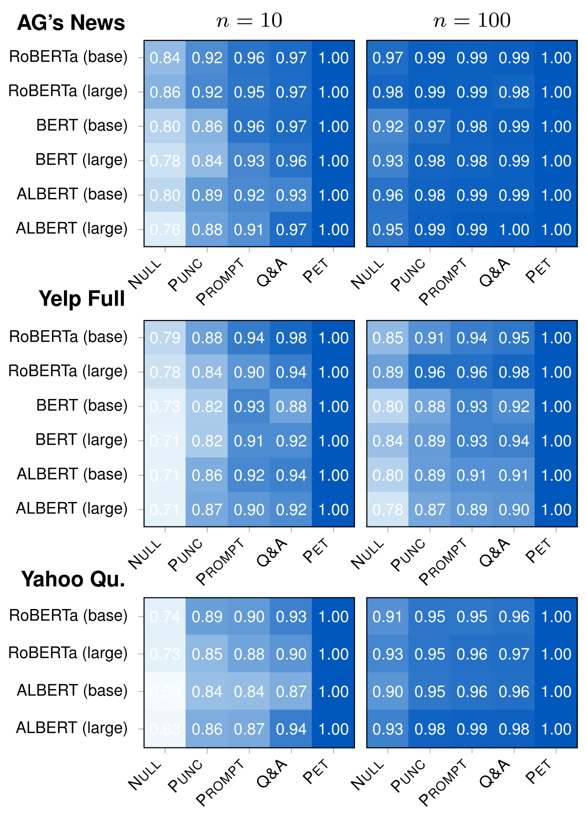

Figure 4: Relative performance of individual pattern groups and PET for different models and sizes. Scores are normalized so that the best performance for each task, number of examples, model, and size is 1.0. Our findings from Q1 (NULL < PUNC < PROMPT < Q&A < PET) also hold for other models and sizes.

scenarios. As a proxy, we resort to *noise patterns* that add random tokens to the input, serving as a lower bound for truly bad patterns. In concrete terms, we add up to three randomly sampled tokens before and after the input.[5] We also create *noise verbalizers* by assigning uniformly selected tokens to each output class. Using this process, we obtain 20 different intentionally bad PVPs per task. For each task, we start with 3 randomly selected, high-quality patterns from our original set of manually designed instructions, add noise PVPs one by one, and investigate the effect on performance.

Results The effect of adding noise PVPs is shown in Figure 5. Performance hardly changes even if more than half of the PVPs are noise PVPs, demonstrating that PET is robust to "bad" instructions. Figure 5 also shows that performance is substantially worse when using *only* noise PVPs.

[5]If there are multiple input texts, we shuffle their order and additionally add 0–3 tokens in between them.

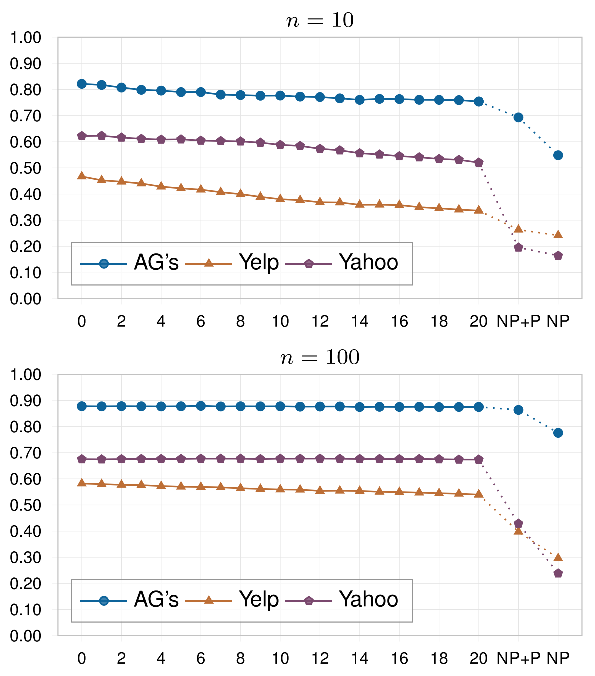

Figure 5: Performance of PET with three randomly selected patterns when adding noise PVPs; the x-axis shows the number of noise PVPs added. Performance hardly changes when adding noise PVPs, showing that PET is very robust to "bad" instructions. We also show performance of using *only* noise PVPs with PET (NP+P) and their average performance without PET (NP).

Q4: How many patterns are required for good performance?

Orthogonal to Q3, what is the minimum number of *high-quality* prompts required for good performance? This is important because we want to minimize the amount of time spent creating PVPs in a practical setting.

Setup We generate, per task, 10 random permutations of the 23 patterns. For each permutation and training set, we use the same setup as in Q1 to compute the average performance obtained with PET when using only the first i, $1 \leq i \leq 5$, patterns.

Results Average performance of PET trained with the first i patterns is shown in Figure 6, relative to the performance of PET trained with all 23 patterns. For all tasks and training set sizes, as little as four patterns are already sufficient to achieve performance close to that of PET trained with all 23 patterns. Surprisingly, PET's performance is much higher than the average performance of a model trained on individual patterns even with $i = 1$.

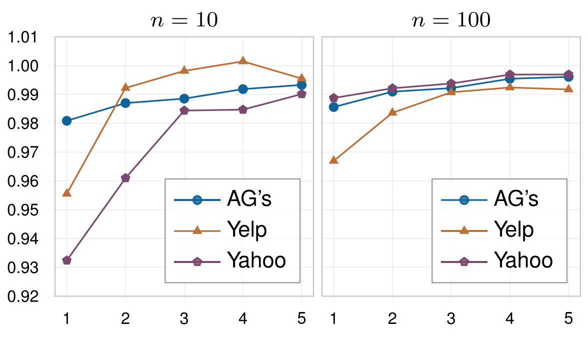

Figure 6: Relative performance of PET with only a subset of patterns compared to that achieved using all 23 manually designed patterns. The x-axis shows the number of patterns used. As little as 4 patterns are sufficient to almost match the performance of a model trained on all patterns.

This indicates that the process of knowledge distillation using unlabeled data is also beneficial when using only a single instruction.

Q5: Are other hyperparameters important?

For true few-shot settings, we want the same set of hyperparameter values to perform well across different tasks; this enables us to adopt these values for new tasks without tuning on task-specific validation sets. We investigate how the hyperparameters, learning rate, training steps, and batch size affect performance.

Setup Based on previous work, we consider learning rates from 10^{-4} to 10^{-6}, training steps from 10 to 1,000, and batch sizes from 1 to 32. Learning rate and batch size are changed for the individual models and the final classifier simultaneously; the number of training steps is varied only for individual models. We modify each hyperparameter independently, keeping all other parameters at their default value (i.e., a learning rate of 10^{-5}, 100 steps and a batch size of 4).

Results Results are shown in Figure 7. For training steps and batch size, performance is relatively stable across a wide range of different values, with more steps and larger batch sizes typically leading to slightly better performance (especially for $n = 100$). Learning rate clearly has the strongest impact on performance, but values of 10^{-5} and $5 \cdot 10^{-5}$ consistently give the best results across tasks; these are also the values typically used for finetuning in prior work (Devlin et al., 2019; Liu et al., 2019).

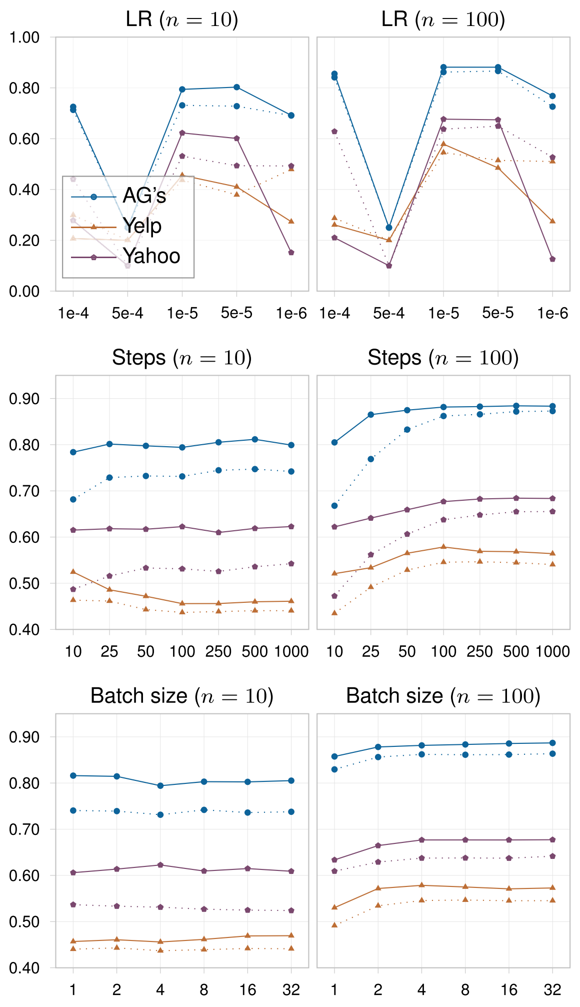

Figure 7: Performance of PET (solid) and avg. performance of individual models (dotted) for different learning rates (LR), training steps (Steps), and batch sizes. Except for LR, performance is stable across a wide range of values. For readability, the legend is only shown in the top left.

Q6: Do we really need unlabeled data?

In contrast to individual PVPs, PET needs unlabeled data, which is not available in some real-world settings. Building on earlier work (Anaby-Tavor et al., 2020; Papanikolaou and Pierleoni, 2020; Yang et al., 2020; Mohapatra et al., 2020; Kumar et al., 2020; Schick and Schütze, 2021), we investigate whether synthetic examples can replace unlabeled data.

Setup We use GPT2-XL (Radford et al., 2019) to generate synthetic unlabeled data: We provide one or two random training examples without labels as *in-context examples* (Brown et al., 2020) and let the model generate an additional example.

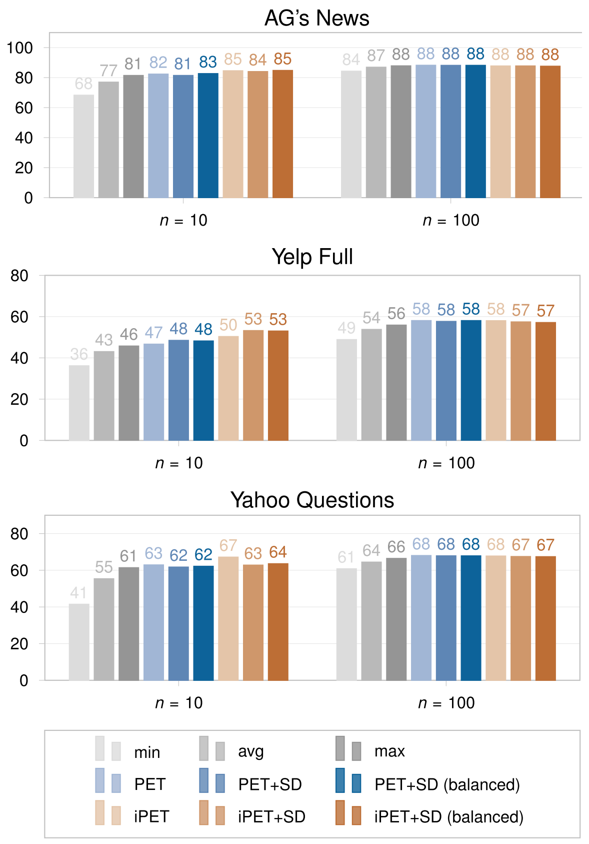

Figure 8: Minimum, average, and maximum accuracy (in percent) of individual patterns compared to regular PET and iPET as well as PET and iPET with synthetic data (+SD). Accuracy with synthetic data is very similar to that obtained with real data.

For two inputs x_1 and x_2, the input given to the model is

Example 1:x_1 ↩ Example 2: x_2 ↩ Example 3:

where ↩, denotes two consecutive line breaks. If an input consists of two texts, we simply concatenate them using the sequence +++ as a separator.

We generate 10,000 examples for $n = 10$ and 30,000 examples for $n = 100$ using top-p sampling (Holtzman et al., 2020) with $p = 0.9$. For each input, we stop the generation process as soon as the model generates two consecutive line breaks. We discard all examples for which the model does not generate two consecutive line breaks within 128 tokens; for datasets with text pairs, we also discard examples where the model fails to generate the sequence separator (+++).

As the datasets obtained with this method may be highly imbalanced regarding the distribution of (unknown) labels, we also experiment with a *balanced* variant: We use the ensemble of models trained on individual PVPs to assign labels to each example and only keep so many examples per label that the resulting dataset—which is used for training the final classifier—is balanced.

Results Figure 8 shows the performance of individual patterns as well as PET and iPET with real and synthetic unlabeled data. Except for iPET on Yahoo Questions with $n = 10$, the accuracy of synthetic data is within one point of real data, with our balanced version performing slightly better. For $n = 10$, using synthetic data even improves accuracy in some cases. This shows that in the absence of unlabeled examples, synthetic data obtained from generative language models can serve as a drop-in replacement without substantially degrading performance.

5 PET for Real-World Tasks

We use our insights from §4 to apply PET to the RAFT benchmark, a collection of 11 diverse real-world tasks whose automated solution has inherent value to someone (Alex et al., 2021). These tasks are challenging for few-shot approaches: they require domain expertise, understanding of detailed instructions, processing of long inputs, and handling a large number of output classes.

Tasks and Datasets The RAFT benchmark includes 11 tasks from different domains: ADE, B77, NIS, OSE, Over, SOT, SRT, TAI, ToS, TEH, and TC; for a detailed overview see Alex et al. (2021). Each task comes with 50 labeled training examples; in accordance with the RAFT rules, we additionally make use of the unlabeled data (ranging from 150 to 5,000 examples) for PET's distillation step. In the case of RAFT, the unlabeled set is the same as the test set. So unlike in §4, our final classifier is directly trained on (unlabeled) test examples.

PVPs Based on Q1 and Q2, we only employ Q&A prompts. To obtain the question q, we make minimal changes to the original instructions of Alex et al. (2021); we rephrase all binary classification tasks as yes/no questions. For example, we rephrase the instruction ‘‘Label the sentence based on whether it is related to an adverse drug effect (ADE).’’ as ‘‘Is this sentence related to an adverse drug effect (ADE)?’’ Following our results from Q4, we specify 4 PVPs per task. For binary classification, we use two different patterns that either include or omit the full task specification of Alex et al. (2021) and combine them with both a yes/no verbalizer and a true/false verbalizer.[6]

Hyperparameters Based on the results of Q5, we mostly keep hyperparameter defaults from §4. However, we make the following changes:

- We replace RoBERTa (base) with ALBERT (xxlarge, v2). While being much slower to train, ALBERT was shown to outperform RoBERTa both in regular and few-shot settings (Lan et al., 2020; Schick and Schütze, 2021; Logan IV et al., 2021).
- Since 1,000 steps cover only 4,000 examples at batch size 4, we finetune the distilled model for 2,000 steps for tasks with more than 4,000 unlabeled examples. This makes sure all examples are seen at least once.
- Following Schick and Schütze (2020) and Schick and Schütze (2021) we train three individual models per PVP. This improves robustness as performance can vary greatly between individual finetuning runs.

Handling Many Labels The B77 dataset consists of banking customer service queries, each annotated with one of 77 possible intents. That large a number of outputs leads to several issues for PET: First, it is impossible to specify a meaningful verbalizer that maps each intent to a single token. We initially experimented with PET's multi-mask version (Schick and Schütze, 2021), but it was too inefficient for experimentation. We instead proceed as follows. We rephrase the task as binary classification, where for each pair of query x and intent y, the task is to decide whether y is the correct intent for x. For each original training example (x, y), we generate one example (x, y, True) and four examples (x, y', False) with randomly sampled, incorrect intents y'. As this increases the amount of data fivefold, we finetune each individual model for 500 instead of 100 steps.

This approach still is not particularly efficient: Reframing the task as binary classification means that for each input, 77 forward passes are required to find the correct intent. We thus train the final model as a regular classifier with 77 different output classes; for training this classifier on input x, we set the target probability of output y proportional to the probability of True being the correct output for (x, y) according to our ensemble of binary classifiers.

Finally, another issue is that with 50 labeled examples, at least 27 labels are not covered in the training set; this may bias a model to never predict these labels. To alleviate this issue, we train two generations of models using iPET. For training the second generation, we obtain training data covering all possible labels similar to Schick and Schütze (2020): For each label, we pick the two examples from the unlabeled data for which this label is most likely according to the first generation.

The nature of RAFT makes it hard to measure the impact of any of these choices. While we could conduct experiments similar to those in §4, none of the datasets considered there has a structure similar to B77; as our modifications affect only one of 11 tasks, we leave further analysis for future work.

Monitoring We checked for TRAIN SET UNDERFITTING and CONSTANT PREDICTIONS (§4) to detect finetuning issues. Unlike in §4, on RAFT we encountered some issues that could *not* be resolved simply by retraining with a different seed:

- We observed TRAIN SET UNDERFITTING for the final classifier on B77. This may be due to the classification head for 77 classes introducing many new parameters; we train the final model for 5,000 instead of 2,000 steps, which fixed this issue.
- We observed CONSTANT PREDICTIONS for the ToS training set. Doubling the number of training steps resolved this problem.
- Finally, we also observed CONSTANT PREDICTIONS on the unlabeled data of SRI. Upon manually inspecting the training set, we observed that all but one out of 50 examples have the same label. As all models already classified the training set perfectly, we left the setup for our SRI submission unchanged.

Results For all 11 tasks, Table 1 shows results of PET and baselines.[7] As can be seen, PET performs

[6]For a full list of all task specifications, see `https://github.com/oughtinc/raft-baselines`. The full set of PVPs can be found at `https://github.com/timoschick/pet/tree/master/true-fsl`.

[7]All results are taken directly from the leaderboard at `https://huggingface.co/spaces/ought/raft-leaderboard`.

Method	ADE	B77	NIS	OSE	Over	SOT	SRI	TAI	ToS	TEH	TC	Avg
GPT-2	60.0	12.1	56.1	24.5	49.8	38.0	49.2	61.2	49.8	31.1	72.3	45.8
GPT-Neo	45.2	14.9	40.8	34.3	68.1	40.6	49.3	60.5	56.5	55.4	63.6	48.1
AdaBoost	54.3	02.3	62.6	47.5	83.8	45.5	50.6	55.6	56.0	44.3	62.5	51.4
snlt	60.3	24.8	58.5	30.2	83.1	33.6	49.2	62.6	54.0	44.9	79.1	52.8
GPT-3	68.6	29.9	67.9	43.1	**93.7**	76.9	**51.6**	**65.6**	57.4	52.6	82.1	62.7
SetFit	72.6	53.8	**87.2**	52.1	90.7	68.2	49.3	62.8	**62.0**	**53.2**	**83.7**	66.9
Pet	**82.2**	**59.3**	85.7	**64.6**	90.8	**81.6**	49.3	63.8	57.6	48.3	82.4	**69.6**
Human	83.0	60.7	85.7	64.6	91.7	90.8	46.8	60.9	62.7	72.2	89.7	73.5

Table 1: Performance (macro F1 multiplied by 100) of baselines and Pet on the RAFT benchmark (Alex et al., 2021). Best model performance is shown in **bold**, best overall performance (including human annotators) is underlined. The final column shows average performance across all 11 tasks.

better than all other approaches on average, achieving near-human performance for 7 out of 11 tasks. Note, however, that non-expert humans perform worse than the majority baseline on SRI, so results on this task should be taken with a grain of salt. Pet also clearly outperforms a GPT-3 model (Brown et al., 2020) by almost 7 points, despite the latter being larger by several orders of magnitude. While Pet is particularly successful on ADE, B77, and OSE (where it outperforms GPT-3 by 13.6, 21.5, and 29.4 points, respectively), it performs comparably poorly on datasets in the law (Over, ToS) and social media (TEH, TC) domains. Our approach for handling many labels performs surprisingly well on B77 without any tuning of its parameters. Due to the nature of RAFT, we cannot perform further analysis or ablation studies.

6 Discussion

Our experimental results in §4 and §5 show that strong performance in few-shot settings is clearly possible without manual prompt tuning or hyperparameter optimization on large dev sets; in other words, Pet *can successfully be applied in true few-shot settings.* While we believe that it should be an important goal of future work to make LMs more robust to different instructions, even with current models it is relatively easy to successfully apply Pet when following a few simple principles—such as rephrasing the task in a Q&A format, using simple vocabulary and single-token verbalizers where possible, and specifying at least a handful of different patterns. In light of these findings, we also hope that future work will not view human involvement in prompt design as a drawback of instruction-based approaches, but rather as an exciting possibility to communicate with models in ways other than exclusively through examples.

Our study has limitations. First, a major obstacle to using Pet in real-world applications is that we do not know a priori how well it performs for a given task; we therefore believe an important next step is to investigate methods for estimating performance without access to large test sets—for example, through model calibration (Desai and Durrett, 2020; Jiang et al., 2021; Zhao et al., 2021)—in real-world settings. In addition, we did not fully explore the capabilities of Pet; for example, we did not investigate domain-adaptive pretraining (Gururangan et al., 2020) and auxiliary language modeling (Chronopoulou et al., 2019), both of which were shown to be helpful by Schick and Schütze, (2020). We also did not quantify the impact of our decisions regarding B77 and the effectiveness of monitoring (§4) and only considered English models and datasets. Finally, we did not examine Pet's performance beyond aggregate scores. While this is not feasible on RAFT due to its nature, performing such analysis either with other datasets or with methods such as the ones proposed by Ribeiro et al. (2020) would be relevant future work to understand real-world capabilities of instruction-based approaches more comprehensively.

7 Conclusion

In light of recent work casting doubt on the performance of prompt-based approaches in true few-shot settings (Perez et al., 2021), we have conducted an extensive study of Pet. In a controlled environment, we found that manually designed instructions outperform null prompts, with Q&A-style prompts performing best (Q1, Q2). Across different tasks, models and training set sizes, Pet consistently outperforms even the best individual prompt (Q1, Q2). We have also shown that Pet is robust to uninformative prompts and to different choices of hyperparameters (Q3, Q5), that as little as four prompts are sufficient to reach good performance (Q4), and that synthetic examples can be used to replace unlabeled data (Q6). Based on these insights, we applied Pet to a benchmark of real-world tasks, where it achieves near-human performance for 7 out of 11 tasks without any tuning on a dev set, demonstrating the potential of instruction-based approaches in true few-shot settings.

Acknowledgments

This work was funded by the European Research Council (ERC #740516). We thank the anonymous reviewers and the action editor for their helpful comments.

References

Neel Alex, Eli Lifland, Lewis Tunstall, Abhishek Thakur, Pegah Maham, C. Jess Riedel, Emmie Hine, Carolyn Ashurst, Paul Sedille, Alexis Carlier, Michael Noetel, and Andreas Stuhlmüller. 2021. RAFT: A real-world few-shot text classification benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? Deep learning to the rescue! *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7383–7390. https://doi.org/10.1609/aaai.v34i05.6233

Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. FLEX: Unifying evaluation for few-shot NLP. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

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

Ming-Wei Chang, Lev Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, pages 830–835. AAAI Press.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.194

Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. 2019. An embarrassingly simple approach for transfer learning from pretrained 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. https://doi.org/10.18653/v1/N19-1213

Zewei Chu, Karl Stratos, and Kevin Gimpel. 2021. Unsupervised label refinement improves dataless text classification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4165–4178, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.365

Joe Davison, Joshua Feldman, and Alexander Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings* *of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, Hong Kong, China. Association for Computational Linguistics. `https://doi.org/10.18653/v1/D19-1109`

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2020.emnlp-main.21`

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.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *Computing Research Repository*, arXiv:2002.06305.

Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? *Computing Research Repository*, arXiv: 2010.11982.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031. `https://doi.org/10.1162/tacl_a_00410`

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Computing Research Repository*, arXiv:2101.03961.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2020.acl-main.740`

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: Word-level Adversarial ReProgramming. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2021.acl-long.381`

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

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? On the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977. `https://doi.org/10.1162/tacl_a_00407`

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438. `https://doi.org/10.1162/tacl_a_00324`

Sawan Kumar and Partha Talukdar. 2021. Reordering examples helps during priming-based few-shot learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4507–4518, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.395

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

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*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.243

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/K17-1034

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

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

Robert L. Logan IV, Ivana Balaževic, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models. *Computing Research Repository*, arXiv:2106.13353.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *Computing Research Repository*, arXiv:2104.08786.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *Computing Research Repository*, arXiv:1806.08730.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2021. Noisy channel language model prompting for few-shot text classification. *Computing Research Repository*, arXiv:2108.04106.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *Computing Research Repository*, arXiv:2104.08773.

Biswesh Mohapatra, Gaurav Pandey, Danish Contractor, and Sachindra Joshi. 2020. Simulated chats for task-oriented dialog: Learning to generate conversations from instructions. *Computing Research Repository*, arXiv:2010.10216. https://doi.org/10.18653/v1/2021.findings-emnlp.103

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S. Corrado, and Jeffrey Dean. 2014. Zero-shot learning by convex combination of semantic embeddings.

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.

Yannis Papanikolaou and Andrea Pierleoni. 2020. DARE: Data augmented relation extraction with GPT-2. *Computing Research Repository*, arXiv:2004.13845.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

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

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2020.acl-main.442`

Bernardino Romera-Paredes and Philip Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161.

Oscar Sainz, Oier Lopez de Lacalle, Gorka Labaka, Ander Barrena, and Eneko Agirre. 2021. Label verbalization and entailment for effective zero and few-shot relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1199–1212, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2021.emnlp-main.92`

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M. Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5569–5578, Barcelona, Spain (Online). International Committee on Computational Linguistics. `https://doi.org/10.18653/v1/2020.coling-main.488`

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2021.naacl-main.185`

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. *Computing Research Repository*, arXiv:2001.07676. `https://doi.org/10.18653/v1/2021.eacl-main.20`

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2021.emnlp-main.555`

Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in NLP. *Transactions of the Association for Computational Linguistics*. `https://doi.org/10.1162/tacl_a_00434`

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020.

AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.346

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1355

Yi Sun, Yu Zheng, Chao Hao, and Hangping Qiu. 2021. NSP-BERT: A prompt-based zero-shot learner through an original pre-training task–next sentence prediction. *Computing Research Repository*, arXiv:2109.03564.

Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4980–4991, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.407

Sappadla Prateek Veeranna, Jinseok Nam, Eneldo Loza Mencıa, and Johannes Fürnkranz. 2016. Using semantic similarity for multi-label zero-shot classification of text documents. In *Proceeding of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges, Belgium: Elsevier*, pages 423–428.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Albert Webson and Ellie Pavlick. 2021. Do prompt-based models really understand the meaning of their prompts? *Computing Research Repository*, arXiv:2109.01247. https://doi.org/10.18653/v1/W18-5446

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E. Peters. 2020. Learning from task descriptions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1361–1375, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.105

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation for consistency training. *Computing Research Repository*, arXiv:1904.12848.

Liang Xu, Xiaojing Lu, Chenyang Yuan, Xuanwei Zhang, Huilin Xu, Hu Yuan, Guoao Wei, Xiang Pan, Xin Tian, Libo Qin, and Hu Hai. 2021. FewCLUE: A Chinese few-shot learning evaluation benchmark. *Computing Research Repository*, arXiv:2107.07498.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.90

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. *Computing Research Repository*, arXiv:2104.08835.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. 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 3914–3923, Hong Kong, China. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339, Virtual. PMLR.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Ben Zhou, Daniel Khashabi, Chen-Tse Tsai, and Dan Roth. 2018. Zero-shot open entity typing as type-compatible grounding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2065–2076, Brussels, Belgium. Association for Computational Linguistics. `https://doi.org/10.18653/v1/D18-1231`