A **cloze test** is an exercise, test, or assessment consisting of a portion of language with certain items, words, or signs removed, where the participant is asked to replace the missing language item.

Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference

Timo Schick
Sulzer GmbH
Munich, Germany
timo.schick@sulzer.de

Hinrich Schütze
Center for Information and Language Processing
LMU Munich, Germany
inquiries@cislmu.org

Abstract

Some NLP tasks can be solved in a fully unsupervised fashion by providing a pretrained language model with "task descriptions" in natural language (e.g., Radford et al., 2019). While this approach underperforms its supervised counterpart, we show in this work that the two ideas can be combined: We introduce Pattern-Exploiting Training (PET), a semi-supervised training procedure that reformulates input examples as cloze-style phrases to help language models understand a given task. These phrases are then used to assign soft labels to a large set of unlabeled examples. Finally, regular supervised training is performed on the resulting training set. For several tasks and languages, PET outperforms both supervised training and unsupervised approaches in low-resource settings by a large margin.[1]

? what about when we reach a large quantity of data ?

1 Introduction

Learning from examples is the predominant approach for many natural language processing tasks: A model is trained on a large number of labeled examples from which it then generalizes to unseen data. As the required labeling process is often expensive, a common scenario throughout NLP is to have only a few labeled examples. Unfortunately, learning from examples *alone* is difficult when the number of examples is small. For instance, assume we are given the following pieces of text:

- T_1 = This was the best pizza I've ever had.
- T_2 = Pretty bad. You can get better sushi down the road for half the price.
- T_3 = Pizza was average. Not worth what they were asking.

[1] Our implementation is publicly available at https://github.com/timoschick/pet.

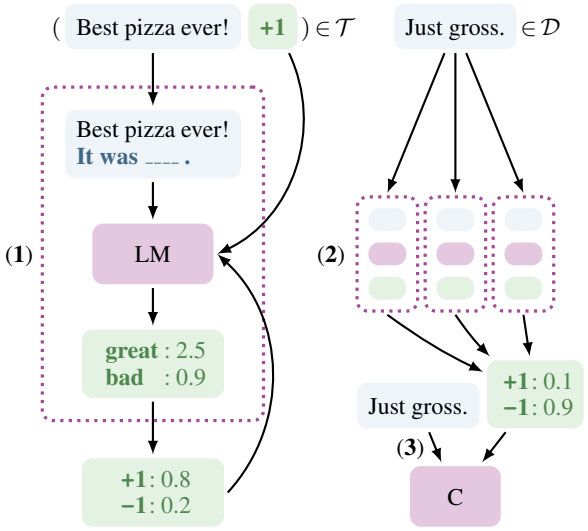

Figure 1: Exemplary application of PET for sentiment classification **(1)** Various patterns are used to convert training examples to cloze questions for finetuning language models. **(2)** The ensemble of trained models annotates unlabeled data. **(3)** A classifier is trained on the so-obtained, soft labeled dataset.

Furthermore, imagine we are told that the labels of T_1 and T_2 are l_1 and l_2, respectively, and we are asked to infer the correct label for T_3. Based only on these examples, this is impossible because plausible justifications can be found for both l_1 and l_2. However, if we know that the underlying task is to identify whether the text says anything about prices, we can easily assign l_2 to T_3. As this illustrates, solving a task from only a few examples becomes much easier when we also have a description that helps us *understand* the task.

With the rise of pretrained language models such as GPT (Radford et al., 2018), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2019), the idea of providing task descriptions has become feasible for neural architectures: We can simply append such descriptions in natural language to an input and let the language model

predict continuations that solve the task (Radford et al., 2019; Puri and Catanzaro, 2019). So far, however, this idea has only been considered in zero-shot scenarios where no training data is available at all.

In this work, we show that providing task descriptions can successfully be combined with regular supervised learning: We introduce **P**attern-**E**xploiting **T**raining (PET), a semi-supervised training procedure that uses natural language patterns to reformulate input examples into cloze-style phrases helping a language model to identify the task to be solved. As shown in Figure 1, PET works in three steps: First, for each pattern a separate language model is finetuned on a small training set $\mathcal{T}$. The ensemble of all models is then used to annotate a large unlabeled dataset $\mathcal{D}$ with soft labels. Finally, a standard classifier is trained on the soft-labeled dataset. We also devise an iterative variant of PET (iPET), in which this process is repeated with increasing training set sizes.

so the description is infered ?

what is a soft label ?

On a diverse set of NLP tasks in multiple languages, we show that using PET results in large improvements over both unsupervised approaches and regular supervised training when the number of available labeled examples is limited.

2 Related Work

Providing hints in the form of natural language patterns for zero-shot learning was first proposed by Radford et al. (2019), who used this idea for challenging tasks such as reading comprehension, machine translation and question answering. Recently, the same idea has been applied to unsupervised text classification (Puri and Catanzaro, 2019) and commonsense knowledge mining (Davison et al., 2019). Shwartz et al. (2020) use predefined patterns to generate various background questions and answers to aid unsupervised question answering. For relation extraction, Bouraoui et al. (2020) automatically identify sentences that express given relations and then use these sentences as patterns. McCann et al. (2018) present various tasks in the form of natural language sentences; however, their motivation is not to enable few-shot learning, but to provide a unified framework for several NLP tasks.

Much recent work uses cloze-style phrases to probe the knowledge that masked language models acquire during pretraining; this is typically done without any task-specific finetuning. Probing tasks include probing for factual and commonsense knowledge (Petroni et al., 2019; Wang et al., 2019), linguistic capacities (Ettinger, 2020; Kassner and Schtze, 2019), understanding of rare words (Schick and Schütze, 2020), and ability to perform symbolic reasoning (Talmor et al., 2019). Some recent work also interprets the Winograd Schema Challenge (Levesque et al., 2012) and variants thereof as cloze tasks to investigate commonsense reasoning abilities of language models (Trinh and Le, 2018; Sakaguchi et al., 2019). Similar to our work, Jiang et al. (2019) consider the problem of finding the best description for a given task. However, all of this prior work only considers the usage of patterns for probing language models and not for improving downstream task performance.

3 Pattern-Exploiting Training

Let M be a masked language model with vocabulary V and mask token $\_\_\_\_ \in V$, and let $\mathcal{L}$ be a set of labels for our target classification task A. We write an input for task A as a sequence of phrases $\mathbf{x} = (s_1, \ldots, s_k)$ with $s_i \in V^*$; for example, $k = 1$ if A is regular text classification (single input text) and $k = 2$ if A is textual inference (two input sentences). We define a *pattern* to be a function P that takes $\mathbf{x}$ as input and outputs a single phrase $P(\mathbf{x}) \in V^*$ that contains exactly one mask token, i.e., its output can be viewed as a cloze question. Furthermore, we define a *verbalizer* as an injective function $v : \mathcal{L} \rightarrow V$ that maps each label to a word from M's vocabulary. We refer to (P, v) as a *pattern-verbalizer pair* (PVP).

Using a PVP (P, v) enables us to solve task A as follows: Given an input $\mathbf{x}$, we apply P to obtain an input representation $P(\mathbf{x})$, which is then processed by M to determine the label $y \in \mathcal{L}$ for which $v(y)$ is the most likely substitute for the mask. For example, consider the task of identifying whether two sentences a and b contradict each other (label y_0) or agree with each other (y_1). For this task, we may choose the pattern

$$P(a, b) = a? \_\_\_\_, b.$$

combined with a verbalizer v that maps y_0 to "Yes" and y_1 to "No". Given an example input pair

$$\mathbf{x} = (\text{Mia likes pie, Mia hates pie}),$$

the task now changes from having to assign a label without inherent meaning to answering whether the most likely choice for the masked position in

$$P(\mathbf{x}) = \text{Mia likes pie? } \_\_\_\_, \text{ Mia hates pie.}$$

is "Yes" or "No".

3.1 PVP Training and Inference

Let $\mathbf{p} = (P, v)$ be a PVP. We assume access to a small training set $\mathcal{T}$ and a (typically much larger) set of unlabeled examples $\mathcal{D}$. For each sequence $\mathbf{z} \in V^*$ that contains exactly one mask token and $w \in V$, we denote with $M(w \mid \mathbf{z})$ the unnormalized score that the language model assigns to w at the masked position. Given some input $\mathbf{x}$, we define the score for label $l \in \mathcal{L}$ as

$$s_{\mathbf{p}}(l \mid \mathbf{x}) = M(v(l) \mid P(\mathbf{x}))$$

and obtain a probability distribution over labels using standard softmax:

$$q_{\mathbf{p}}(l \mid \mathbf{x}) = \frac{e^{s_{\mathbf{p}}(l|\mathbf{x})}}{\sum_{l' \in \mathcal{L}} e^{s_{\mathbf{p}}(l'|\mathbf{x})}}$$

We use the cross-entropy between $q_{\mathbf{p}}(l \mid \mathbf{x})$ and the true (one-hot) distribution of training example $(\mathbf{x}, l)$ – summed over all $(\mathbf{x}, l) \in \mathcal{T}$ – as loss for finetuning M for $\mathbf{p}$.

3.2 Auxiliary Language Modeling

In our application scenario, only a few training examples are available and catastrophic forgetting can occur when finetuning M. Since a model finetuned on some PVP is still a language model at its core, we address this by using language modeling as an auxiliary task. With L_{CE} denoting cross-entropy loss and L_{MLM} language modeling loss, we compute the final loss as

$$L = (1 - \alpha) \cdot L_{\text{CE}} + \alpha \cdot L_{\text{MLM}}$$

This idea was recently applied by Chronopoulou et al. (2019) in a data-rich scenario. As L_{MLM} is typically much larger than L_{CE}, in preliminary experiments, we found a small value of $\alpha = 10^{-4}$ to consistently give good results, so we use it in all our experiments. To obtain sentences for language modeling, we use the unlabeled set $\mathcal{D}$. However, we do not train directly on each $\mathbf{x} \in \mathcal{D}$, but rather on $P(\mathbf{x})$, where we never ask the language model to predict anything for the masked slot.

3.3 Combining PVPs

A key challenge for a pattern-based approach in a low-resource scenario is that in the absence of a large development set, it is hard to identify which PVPs perform well. To overcome this problem, we resort to the following strategy, which closely resembles knowledge distillation (Hinton et al., 2015). First, we define a set $\mathcal{P}$ of PVPs that intuitively make sense for a given task. We then use these PVPs to automatically create a large soft-labeled dataset $\mathcal{T}_C$ as follows:

1. We finetune a separate language model $M_{\mathbf{p}}$ for each $\mathbf{p} \in \mathcal{P}$. As $\mathcal{T}$ is small, this finetuning is cheap even for a large number of PVPs.

2. We use the ensemble $\mathcal{M} = \{M_{\mathbf{p}} \mid \mathbf{p} \in \mathcal{P}\}$ of finetuned models to annotate examples from $\mathcal{D}$. We first combine the unnormalized class scores for each example $\mathbf{x} \in \mathcal{D}$ as

$$s_{\mathcal{M}}(l \mid \mathbf{x}) = \frac{1}{Z} \sum_{\mathbf{p} \in \mathcal{P}} w(\mathbf{p}) \cdot s_{\mathbf{p}}(l \mid \mathbf{x})$$

   where $Z = \sum_{\mathbf{p} \in \mathcal{P}} w(\mathbf{p})$ and the $w(\mathbf{p})$ are weighting terms for the PVPs. We experiment with two different realizations of this weighing term: either we simply set $w(\mathbf{p}) = 1$ for all $\mathbf{p}$ or we set $w(\mathbf{p})$ to be the accuracy obtained using $\mathbf{p}$ on the training set *before* training. We refer to these two variants as *uniform* and *weighted*, respectively. An idea similar to our weighted variant was recently proposed by Jiang et al. (2019) in a zero-shot setting.

3. We transform the above scores into a probability distribution q using softmax. Following Hinton et al. (2015), we use a temperature of $T = 2$ to obtain a suitably soft distribution. The pair $(\mathbf{x}, q)$ is added to our new (soft-labeled) training set $\mathcal{T}_C$.

Finally, we finetune a pretrained language model C with a regular sequence classification head on $\mathcal{T}_C$; this model then serves as our final classifier.

3.4 Iterative PET

By directly distilling the knowledge of all pattern-based models into a single classifier C, we deprive the individual models of an opportunity to learn from each other. As we assume some patterns to perform (possibly much) worse than others, this means that the training set $\mathcal{T}_C$ for our final model may contain many falsely labeled examples.

To compensate for this shortcoming, we devise iPET, an iterative variant of PET. The core idea of iPET is to train various *generations* of models on datasets of increasing size; these datasets are

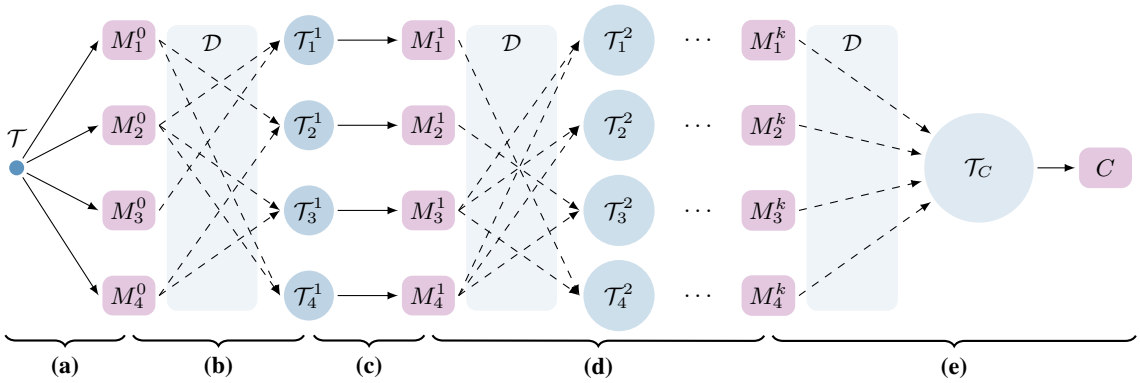

Figure 2: Schematic representation of iPET **(a)** The initial training set is used to train an ensemble of models as in regular PET. **(b)** For each model, a random subset of two other models ($\lambda = 2/3$) is used to generate a new training set by labeling examples from $\mathcal{D}$. **(c)** A new set of PET models is trained using the larger, model-specific datasets. **(d)** The previous two steps are repeated k times, each time increasing the size of the generated training sets by a factor of d. **(e)** The set of models at iteration k is used to create a soft-labeled dataset $\mathcal{T}_C$ as in regular PET; the final classifier C is trained on this dataset.

obtained by enriching the original dataset $\mathcal{T}$ with examples from $\mathcal{D}$ that are labeled using other models from the previous generation. To avoid semantic drift, we select examples $\mathbf{x} \in \mathcal{D}$ for which the previous generation of models is sufficiently confident about their prediction.

Specifically, let $\mathcal{M}^0 = \{M_1^0, \ldots, M_n^0\}$ be the initial set of PET language models finetuned on $\mathcal{T}$, where each model M_i^0 uses some PVP $\mathbf{p}_i$. Instead of directly combining their predictions to train a final classifier, we train k generations of models $\mathcal{M}^1, \ldots, \mathcal{M}^k$ where $\mathcal{M}^j = \{M_1^j, \ldots, M_n^j\}$ and each M_i^j is trained on its own training set $\mathcal{T}_i^j$ with $\mathbf{p}_i$ as PVP. In each iteration, we multiply the training set size by a fixed constant $d \in \mathbb{N}$ while maintaining the label ratio of the original dataset. That is, with $c(l)$ denoting the number of examples with label l in $\mathcal{T}$, each $\mathcal{T}_i^j$ should contain $d^j \cdot c(l)$ examples with label l. This is achieved by generating each training set $\mathcal{T}_i^j$ as follows:

1. We obtain $\mathcal{M} \subset \mathcal{M}^{j-1} \setminus \{M_i^{j-1}\}$ by randomly choosing $\lambda \cdot (n-1)$ models from the previous generation with $\lambda \in (0, 1]$ being a hyperparameter.

2. Using this subset, we create a labeled dataset

$$\mathcal{T}_{\mathcal{M}} = \{(\mathbf{x}, \arg\max_{l \in \mathcal{L}} s_{\mathcal{M}}(l \mid \mathbf{x})) \mid \mathbf{x} \in \mathcal{D}\}.$$

   For each $l \in \mathcal{L}$, we obtain $\mathcal{T}_{\mathcal{M}}(l) \subset \mathcal{T}_{\mathcal{M}}$ by randomly choosing $(d^j - 1) \cdot c(l)$ examples with label l from $\mathcal{T}_{\mathcal{M}}$. To avoid training future generations on falsely labeled data, we prefer examples for which the ensemble of models is confident in its prediction. Therefore, when drawing from $\mathcal{T}_{\mathcal{M}}$, we set the probability of each $(\mathbf{x}, y)$ proportional to $s_{\mathcal{M}}(l \mid \mathbf{x})$.

3. We define $\mathcal{T}_i^j = \mathcal{T} \cup \bigcup_{l \in \mathcal{L}} \mathcal{T}_{\mathcal{M}}(l)$. As can easily be verified, this dataset contains $d^j \cdot c(l)$ examples for each $l \in \mathcal{L}$.

We finally use $\mathcal{M}^k$ to create $\mathcal{T}_C$ and train the final classifier C as in regular PET. The core steps of iPET are also depicted in Figure 2.

With minor adjustments, iPET can even be used in a zero-shot setting. To this end, we set $\mathcal{M}^0$ to be the set of *untrained* models. As we have no initial dataset, we modify the creation of each $\mathcal{T}_i^1$ as follows:

- We define $c(l) = 10/(d \cdot |\mathcal{L}|)$ for all $l \in \mathcal{L}$;

- As $\mathcal{T}_{\mathcal{M}}$ may not contain enough examples for some label l, we create all $\mathcal{T}_{\mathcal{M}}(l)$ by sampling from the 100 examples $\mathbf{x} \in \mathcal{D}$ for which $s_{\mathcal{M}}(l \mid x)$ is the highest, even if $l \neq \arg\max_{l \in \mathcal{L}} s_{\mathcal{M}}(l \mid x)$.

For each subsequent generation, we proceed exactly as in regular iPET training.

3.5 Automatic Verbalizer Search

Given a set of patterns $P_1, \ldots, P_n$, manually finding a verbalization $v(l)$ for each $l \in \mathcal{L}$ that represents the meaning of l well and corresponds to a single token in V can be difficult. We therefore

devise *automatic verbalizer search* (AVS), a procedure that automatically finds suitable verbalizers given a training set $\mathcal{T}$ and a language model M.

Assuming we already have a PVP $\mathbf{p} = (P, v)$, we can easily check whether some token $t \in V$ is a good verbalization of $l \in \mathcal{L}$. To this end, we define $\mathbf{p}[l \leftarrow t] = (P, v')$, where v' is identical to v, except that $v'(l) = t$. Intuitively, if t represents l well, then $q_{\mathbf{p}[l \leftarrow t]}(l \mid \mathbf{x})$ (i.e., the probability M assigns to t given $P(\mathbf{x})$) should be high only for those examples $(\mathbf{x}, y) \in \mathcal{T}$ where $y = l$. We thus define the score of t for l given $\mathbf{p}$ as

$$s_l(t \mid \mathbf{p}) = \frac{1}{|\mathcal{T}_l|} \cdot \sum_{(\mathbf{x},y) \in \mathcal{T}_l} q_{\mathbf{p}[l \leftarrow t]}(l \mid \mathbf{x}) - \frac{1}{|\mathcal{T} \setminus \mathcal{T}_l|} \cdot \sum_{(\mathbf{x},y) \in \mathcal{T} \setminus \mathcal{T}_l} q_{\mathbf{p}[l \leftarrow t]}(l \mid \mathbf{x})$$

where $\mathcal{T}_l = \{(\mathbf{x}, y) \in \mathcal{T} : y = l\}$ is the set of all training examples with label l. While this allows us to easily compute the best verbalization for l as

$$\hat{t} = \arg\max_{t \in V} s_l(t \mid \mathbf{p}),$$

it requires us to already know verbalizations $v(l')$ for all other labels l'.

AVS solves this problem as follows: We first assign random verbalizations to all labels and then repeatedly recompute the best verbalization for each label. As we do not want the resulting verbalizer to depend strongly on the initial random assignment, we simply consider multiple such assignments. Specifically, we define an initial probability distribution ρ_0 where for all $t \in V, l \in \mathcal{L}$, $\rho_0(t \mid l) = 1/|V|$ is the probability of choosing t as verbalization for l. For each $l \in \mathcal{L}$, we then sample k verbalizers $v_1, \ldots, v_k$ using ρ_0 to compute

$$s_l^k(t) = \frac{1}{n \cdot k} \sum_{i=1}^{n} \sum_{j=1}^{k} s_l(t \mid (P_i, v_j))$$

for all $t \in V$.[2] These scores enable us to define a probability distribution ρ_1 that more closely reflects each word's suitability as a verbalizer for a given label:

$$\rho_1(t \mid l) = \frac{1}{Z} \max(s_l^k(t), \epsilon)$$

where $Z = \sum_{t' \in V} \max(s_l^k(t'), \epsilon)$ and $\epsilon \geq 0$ ensures that ρ_1 is a proper probability distribution.

[2]Note that the score $s_l^k(t)$ jointly considers all patterns; in preliminary experiments, we found this to result in more robust verbalizers.

We repeat this process to obtain a sequence of probability distributions $\rho_1, \ldots, \rho_{i_{\max}}$. Finally, we choose the $m \in \mathbb{N}$ most likely tokens according to $\rho_{i_{\max}}(t \mid l)$ as verbalizers for each l. During training and inference, we compute the unnormalized score $s_{\mathbf{p}}(y \mid \mathbf{x})$ for each label by averaging over its m verbalizers.

4 Experiments

4.1 Setup

We evaluate PET on four English NLP datasets: Yelp Reviews, AG's News, Yahoo Questions (Zhang et al., 2015) and MNLI (Williams et al., 2018). Additionally, we use x-stance (Vamvas and Sennrich, 2020) to investigate how well PET works for other languages. For all experiments on English datasets, we use RoBERTa large (Liu et al., 2019) as language model; for x-stance, we use XLM-R (Conneau et al., 2019). Our implementation is based on the Transformers library (Wolf et al., 2019) and PyTorch (Paszke et al., 2017).

We investigate the performance of both regular supervised training and PET for various training set sizes t. For each t, we obtain the training set $\mathcal{T}$ by choosing t examples evenly distributed across all labels. Similarly, we construct the set $\mathcal{D}$ of unlabeled examples by selecting 10,000 examples per label and removing all labels. For each task and training set size, we perform training three times using different random seeds.

As we consider a few-shot setting, we assume no access to a large development set on which hyperparameters could be optimized. Our choice of hyperparameters is thus based on choices made in previous work and practical considerations. We use a learning rate of $1 \cdot 10^{-5}$ because for regular supervised learning, we found higher learning rates to often result in unstable training with no accuracy improvements even on the training set. On all English datasets, we use a batch size of 16, a maximum sequence length of 256 and perform supervised training for 250 steps. For PET, we subdivide each batch into 4 labeled examples from $\mathcal{T}$ to compute L_{CE} and 12 unlabeled examples from $\mathcal{D}$ to compute L_{MLM}. Accordingly, we multiply the number of total training steps by 4 (i.e., 1000), so that the number of times each labeled example is seen remains constant ($16 \cdot 250 = 4 \cdot 1000$). On x-stance, we always perform training for 3 epochs to match the setup of Vamvas and Sennrich (2020) more closely.

For iPET, we set $\lambda = 0.25$ and $d = 5$; that is, we select 25% of all models to label examples for the next generation and the number of training examples is quintupled after each iteration. We train new generations until each model was trained on at least 1000 examples, i.e., we set $k = \lceil \log_d(1000/|\mathcal{T}|) \rceil$. For training the final classifier C, we use the same set of hyperparameters as for the individual PVP models, but we train for 5000 steps due to the increased training set size. As we train three models per pattern, the ensemble $\mathcal{M}$ (or $\mathcal{M}^0$ for iPET) for n PVPs contains $3 \cdot n$ models.

4.2 Patterns

We now describe the patterns and verbalizers used for all tasks. We use two vertical bars ($\|$) to mark boundaries between text segments.[3]

Yelp For the Yelp Reviews Full Star dataset (Zhang et al., 2015), the task is to estimate the rating that a customer gave to a restaurant on a 1- to 5-star scale based on their review's text. We define the following patterns for an input text a:

$P_1(a) =$ It was ____. a

$P_2(a) =$ a. All in all, it was ____.

$P_3(a) =$ Just ____! $\|$ a

$P_4(a) =$ $a \,\|$ In summary, the restaurant is ____.

We define a single verbalizer v for all patterns as

$v(1) =$ terrible $\quad v(2) =$ bad $\quad v(3) =$ okay
$v(4) =$ good $\quad v(5) =$ great

1, 2, 3, 4, 5 are the number of stars i.e. output labels

AG's News AG's News is a news classification dataset, where given a headline a and text body b, news have to be classified as belonging to one of the categories *World* (1), *Sports* (2), *Business* (3) or *Science/Tech* (4). For $\mathbf{x} = (a, b)$, we define the following patterns:

$P_1(\mathbf{x}) =$ ____: a b $\quad P_2(\mathbf{x}) =$ a (____) b

$P_3(\mathbf{x}) =$ ____ – a b $\quad P_4(\mathbf{x}) =$ a b (____)

$P_5(\mathbf{x}) =$ ____ News: a b

$P_6(\mathbf{x}) =$ [Category: ____] a b

[3]The way different segments are handled depends on the language model being used; they may e.g. be assigned different segment embeddings (Devlin et al., 2019), be separated by special tokens (Liu et al., 2019; Yang et al., 2019) or simply be ignored. For example, "$a \,\|\, b$" is given to BERT as the input "[CLS] a [SEP] b [SEP]".

Again, we use a single verbalizer v for all patterns:

$v(1) =$ World $\quad v(2) =$ Sports
$v(3) =$ Business $\quad v(4) =$ Tech

Yahoo The Yahoo Questions dataset (Zhang et al., 2015) is another text classification dataset, where given a question a and a corresponding answer b, one of ten possible categories has to be assigned. We use the same patterns as for AG's News, but we replace the word "News" in P_5 with the word "Question". Based on the dataset description, we define the verbalizer v as

$v(1) =$ Society $\quad v(2) =$ Science
$v(3) =$ Health $\quad v(4) =$ Education
$v(5) =$ Computer $\quad v(6) =$ Sports
$v(7) =$ Business $\quad v(8) =$ Entertainment
$v(9) =$ Relationship $\quad v(10) =$ Politics

MNLI The MNLI dataset (Williams et al., 2018) consists of text pairs $\mathbf{x} = (a, b)$. The task is to find out whether a implies b (0), a and b contradict each other (1) or neither (2). We define

$P_1(\mathbf{x}) =$ " a " ? $\|$ ____ , " b "

$P_2(\mathbf{x}) =$ a ? $\|$ ____ , b

and consider two different verbalizers v_1 and v_2 that are defined as follows:

$v_1(0) =$ Wrong $\quad v_1(1) =$ Right $\quad v_1(2) =$ Maybe
$v_2(0) =$ No $\quad v_2(1) =$ Yes $\quad v_2(2) =$ Maybe

Combining the two patterns with the two verbalizers results in a total of 4 PVPs.

X-Stance The x-stance dataset (Vamvas and Sennrich, 2020) is a multilingual stance detection dataset with German, French and Italian examples. Each example $\mathbf{x} = (a, b)$ consists of a question a concerning some political issue and a comment b; the task is to identify whether the writer of b supports the subject of the question (0) or not (1). We use two simple patterns

$P_1(\mathbf{x}) =$ " a " $\|$ ____ . " b "

$P_2(\mathbf{x}) =$ $a \,\|$ ____ . b

and define an English verbalizer v_{En} mapping 0 to "Yes" and 1 to "No" as well as a German (French) verbalizer v_{De} (v_{Fr}), replacing "Yes" and "No" with "Ja" and "Nein" ("Oui" and "Non"). We do not define an Italian verbalizer because x-stance does not contain any Italian training examples.

Examples	Training Mode	Yelp	AG's News	Yahoo	MNLI (m)
$\|\mathcal{T}\| = 0$	unsupervised (avg)	33.8 ± 9.6	69.5 ± 7.2	44.0 ± 9.1	39.1 ± 4.3
	unsupervised (max)	40.8 ± 0.0	79.4 ± 0.0	56.4 ± 0.0	43.8 ± 0.0
	iPET	$\mathbf{56.7 \pm 0.2}$	$\mathbf{87.5 \pm 0.1}$	$\mathbf{70.7 \pm 0.1}$	$\mathbf{53.6 \pm 0.1}$
$\|\mathcal{T}\| = 10$	supervised	21.1 ± 1.6	25.0 ± 0.1	10.1 ± 0.1	34.2 ± 2.1
	PET	52.9 ± 0.1	87.5 ± 0.0	63.8 ± 0.2	41.8 ± 0.1
	iPET	$\mathbf{57.6 \pm 0.0}$	$\mathbf{89.3 \pm 0.1}$	$\mathbf{70.7 \pm 0.1}$	$\mathbf{43.2 \pm 0.0}$
$\|\mathcal{T}\| = 50$	supervised	44.8 ± 2.7	82.1 ± 2.5	52.5 ± 3.1	45.6 ± 1.8
	PET	60.0 ± 0.1	86.3 ± 0.0	66.2 ± 0.1	63.9 ± 0.0
	iPET	$\mathbf{60.7 \pm 0.1}$	$\mathbf{88.4 \pm 0.1}$	$\mathbf{69.7 \pm 0.0}$	$\mathbf{67.4 \pm 0.3}$
$\|\mathcal{T}\| = 100$	supervised	53.0 ± 3.1	86.0 ± 0.7	62.9 ± 0.9	47.9 ± 2.8
	PET	61.9 ± 0.0	88.3 ± 0.1	69.2 ± 0.0	74.7 ± 0.3
	iPET	$\mathbf{62.9 \pm 0.0}$	$\mathbf{89.6 \pm 0.1}$	$\mathbf{71.2 \pm 0.1}$	$\mathbf{78.4 \pm 0.7}$
$\|\mathcal{T}\| = 1000$	supervised	63.0 ± 0.5	$\mathbf{86.9 \pm 0.4}$	70.5 ± 0.3	73.1 ± 0.2
	PET	$\mathbf{64.8 \pm 0.1}$	$\mathbf{86.9 \pm 0.2}$	$\mathbf{72.7 \pm 0.0}$	$\mathbf{85.3 \pm 0.2}$

Table 1: Results for RoBERTa (large) on Yelp, AG's News, Yahoo and MNLI (matched) for various training set sizes. Scores for PET were obtained using the weighted variant with manually defined verbalizers.

4.3 Results

English Datasets Table 1 shows results for all English text classification and language understanding tasks; we report mean accuracy and standard deviation across the three training runs.[4] The top rows show performance using all PVPs in a fully unsupervised setting, where both average results across all patterns (avg) and results using the best pattern (max) are reported. Importantly, finding the best pattern would require access to the test set; accordingly, this row serves only as an upper bound. The large difference between both rows highlights the importance of finding a strategy to cope with the fact that we have no means of evaluating which patterns perform well. The third row shows the performance of iPET in a zero-shot setting. As can be seen, iPET outperforms the unsupervised baselines in all cases. Furthermore, zero-shot iPET is on par with a supervised model trained on 1000 examples for Yahoo and even outperforms regular supervised training with 1000 examples on AG's News.

With just 10 training examples, regular supervised learning does not perform above chance. In contrast, PET performs much better than both the fully unsupervised baselines and regular supervised training; training multiple generations using iPET gives consistent improvements. On MNLI, iPET surprisingly performs much better in a zero-shot setting than with 10 training examples.

As we increase the training set size, the performance gains of PET and iPET become smaller, but for both 50 and 100 examples, PET continues to considerably outperform regular supervised training with iPET still giving consistent improvements. For 1000 training examples, PET has no advantage on AG's News but still improves accuracy for all other datasets; especially for MNLI, there is still a large gap between PET and supervised training.

[4]Due to the limit of 2 submissions per 14 hours for the official MNLI test set, we report results on the dev set.

Examples	Mode	De	Fr	It
$\|\mathcal{T}\| = 1000$	supervised	43.3	49.5	41.0
	PET	**66.4**	**68.7**	**64.7**
$\|\mathcal{T}\| = 2000$	supervised	57.4	62.1	52.8
	PET	**69.5**	**71.7**	**67.3**
$\|\mathcal{T}\| = 4000$	supervised	63.2	66.7	58.7
	PET	**71.7**	**74.0**	**69.5**
$\mathcal{T}_{\text{De}}$, $\mathcal{T}_{\text{Fr}}$	supervised	76.6	76.0	71.0
	PET	**77.9**	**79.0**	**73.6**
$\mathcal{T}_{\text{De}} + \mathcal{T}_{\text{Fr}}$	sup. (*)	76.8	76.7	70.2
	supervised	77.6	79.1	75.9
	PET	**78.8**	**80.6**	**77.2**

Table 2: Results on x-stance for XLM-R (base) trained on subsets of $\mathcal{T}_{\text{De}}$ and $\mathcal{T}_{\text{Fr}}$ and for jointly training on all available data ($\mathcal{T}_{\text{De}} + \mathcal{T}_{\text{Fr}}$). (*): Best results reported in Vamvas and Sennrich (2020).

	Yelp	AG's	Yahoo	MNLI
min	39.6	82.1	50.2	36.4
max	52.4	85.0	63.6	40.2
PET uniform	52.7	87.3	**63.8**	**42.0**
PET weighted	**52.9**	**87.5**	**63.8**	41.8

Table 3: Minimum (min) and maximum (max) accuracy of models based on individual patterns as well as PET after training on 10 examples

X-Stance We evaluate PET on x-stance to investigate (i) whether it can also successfully be applied to other languages than English and (ii) whether it also brings improvements for data-rich but difficult tasks. Our setup differs from that of Vamvas and Sennrich (2020) in that we do not perform any hyperparameter optimization on the dev set and use a shorter maximum sequence length (256 vs 512) to speed up training and evaluation.

To investigate whether PET brings benefits even when numerous examples are available, we consider training set sizes of 1000, 2000, and 4000; for each of these configurations, we separately finetune a German and French model to allow for a more straightforward downsampling of the training data. Additionally, we train models on the entire German ($|\mathcal{T}_{\text{De}}| = 33850$) and French train set ($|\mathcal{T}_{\text{Fr}}| = 11790$), respectively. As in this case, we do not have any additional unlabeled dataset, we simply set $\mathcal{D} = \mathcal{T}$. For the German models, we use v_{En} and v_{De} as verbalizers; for the French models, we accordingly choose v_{En} and v_{Fr}. Finally, we also investigate the performance of a model trained jointly on German and French data ($|\mathcal{T}_{\text{De}} + \mathcal{T}_{\text{Fr}}| = 45640$).

Results for all considered training set sizes are shown in Table 2; following Vamvas and Sennrich (2020), we report the macro-average of F1 scores. For Italian, we report the average zero-shot cross lingual performance of the German and French model as there are no Italian training examples. Our results show that PET brings huge improvements across all languages even when training on much more than a thousand examples; notably, PET also considerably improves zero-shot cross lingual performance.

5 Analysis

Combining PVPs We first investigate whether PET is able to cope with situations were some PVPs perform much worse than others. Table 3 compares the performance of the best and worst performing pattern to the performance of PET (for $|\mathcal{T}| = 10$). We see that even after finetuning, the gap between the best and worst pattern is large, especially for Yelp. However, PET is not only able to compensate for this, but even improves accuracies over using only the best-performing pattern across all tasks. We find no clear difference between the uniform and weighted variant of PET.

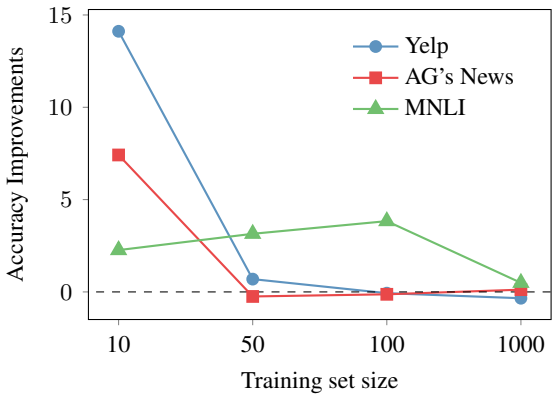

Figure 3: Accuracy improvements for regular PET due to adding L_{MLM} during training

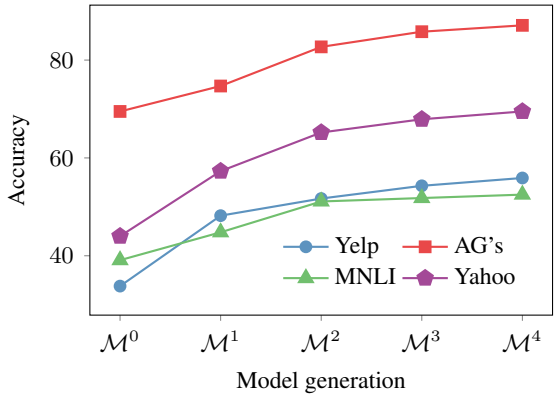

Figure 4: Average accuracy for each generation of models with iPET in a zero-shot setting

Auxiliary Language Modeling We analyze the influence of the auxiliary language modeling task on PET's performance. Figure 3 shows performance improvements from adding the language modeling task for all considered training set sizes. As can be seen, the auxiliary task is extremely valuable when training on just 10 examples. With more data, it becomes less important, sometimes even leading to worse performance. Only for MNLI, we find language modeling to consistently help for all training set sizes considered.

	Yelp	AG's	Yahoo	MNLI
supervised	44.8	82.1	52.5	45.6
PET	**60.0**	**86.3**	**66.2**	**63.9**
PET + AVS	55.2	85.0	58.2	52.6

Table 4: Results for supervised learning, PET and PET with Automatic Verbalizer Search (PET + AVS) after training on 50 examples

Iterative PET To check whether iPET is indeed able to improve models over multiple generations, Figure 4 shows the average performance of all generations of models in a zero-shot setting. As can be seen, each additional iteration does indeed further improve the ensemble's performance. We did not investigate whether continuing this process for even more iterations gives further improvements.

Automatic Verbalizer Search As an alternative to PET with manually defined verbalizers, we analyze the performance of automatic verbalizer search (AVS) as described in Section 3.5. We do so for all tasks with $|\mathcal{T}| = 50$ training examples and set $k = 250$, $\epsilon = 10^{-3}$, $i_{\text{max}} = 5$ and $m = 10$.[5] To speed up the search, we additionally restrict our search space to tokens $t \in V$ that contain at least two alphabetic characters. Of these tokens, we only keep the 10 000 most frequent ones in $\mathcal{D}$.

Results are shown in Table 4. As can be seen, carefully handcrafted verbalizers perform much better than AVS; however, PET with AVS still considerably outperforms regular supervised training while eliminating the challenge of manually finding suitable verbalizers. Table 5 shows the most probable verbalizers found using AVS for the Yelp dataset. While most of the shown verbalizers for this dataset intuitively make sense, we found AVS to struggle with finding good verbalizers for three out of ten labels in the Yahoo dataset and for all MNLI labels.

In-Domain Pretraining Unlike our supervised baseline, PET makes use of the additional unlabeled dataset $\mathcal{D}$ in order to create a soft-labeled dataset for training the final classifier. Thus, at least some of PET's performance gains over the supervised baseline may also arise from this additional in-domain data. To test this hypothesis, we simply further pretrain RoBERTa on in-domain data, a common technique for improving text classification accuracy (e.g., Howard and Ruder, 2018; Sun et al., 2019). As language model pretraining is expensive in terms of GPU usage, we do so only for the Yelp dataset. Results of supervised learning and PET both with and without this in-domain pretraining can be seen in Figure 5. While pretraining on unlabeled data does indeed improve accuracy for supervised training, the supervised model still clearly performs worse than PET, showing that the success of our method is not simply due to the usage of additional unlabeled data. Interestingly, in-domain pretraining is also helpful for PET (especially when training on just 10 examples), indicating that PET leverages unlabeled data in a very different way than regular masked language model pretraining.

[5] We tried values of k and i_{max} in $\{250, 500, 1000\}$ and $\{5, 10, 20\}$, respectively, but found the resulting verbalizers to be almost identical. We did not experiment with other values of ϵ and m.

y	Top Verbalizers
1	worthless, BAD, useless, appalling
2	worse, slow, frustrating, annoying
3	edible, mixed, cute, tasty, Okay
4	marvelous, loved, love, divine, fab
5	golden, magical, marvelous, perfection

Table 5: Most probable verbalizers according to AVS for Yelp with 50 training examples

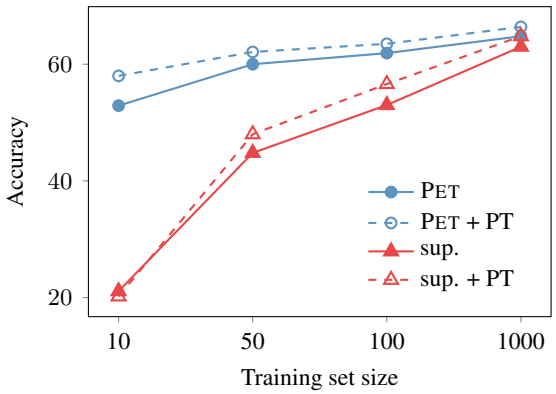

Figure 5: Accuracy of supervised learning (sup.) and PET both with and without pretraining (PT) on Yelp

6 Conclusion

We have shown that providing task descriptions to pretrained language models can be combined with regular supervised training. Our proposed method, PET, consists of defining pairs of cloze question patterns and verbalizers that help leverage the knowledge contained within pretrained lan-

guage models for downstream tasks. We finetune models for all pattern-verbalizer pairs and use them to create large annotated datasets on which regular classifiers can be trained. When the initial amount of training data is limited, PET gives large improvements over regular supervised training.

References

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from bert. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

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.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmn, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale.

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.

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.

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.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

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

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2019. How can we know what language models know?

Nora Kassner and Hinrich Schtze. 2019. Negated lama: Birds cannot fly.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

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.

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

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)*.

Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models.

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.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale.

Timo Schick and Hinrich Schütze. 2020. 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*.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics*, pages 194–206, Cham. Springer International Publishing.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2019. olmpics – on what language model pre-training captures.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning.

Jannis Vamvas and Rico Sennrich. 2020. X-stance: A multilingual multi-target dataset for stance detection. *arXiv preprint arXiv:2003.08385*.

Cunxiang Wang, Shuailong Liang, Yue Zhang, Xiaonan Li, and Tian Gao. 2019. Does it make sense? and why? a pilot study for sense making and explanation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4020–4026, Florence, Italy. Association for Computational Linguistics.

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, Pierric Cistac, Tim Rault, Rmi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Transformers: State-of-the-art natural language processing.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *Computing Research Repository*, arXiv:1906.08237.

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.