Automatically Identifying Words That Can Serve as Labels for Few-Shot Text Classification

Timo Schick   Helmut Schmid   Hinrich Schütze
Center for Information and Language Processing, LMU Munich, Germany
schickt@cis.lmu.de

Abstract

A recent approach for few-shot text classification is to convert textual inputs to cloze questions that contain some form of task description, process them with a pretrained language model and map the predicted words to labels. Manually defining this mapping between words and labels requires both domain expertise and an understanding of the language model's abilities. To mitigate this issue, we devise an approach that automatically finds such a mapping given small amounts of training data. For a number of tasks, the mapping found by our approach performs almost as well as hand-crafted label-to-word mappings.[1]

1 Introduction

Pretraining language models on large corpora has led to improvements on a wide range of NLP tasks (Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019, *inter alia*), but learning to solve tasks from only a few examples remains a challenging problem. As small datasets are common for many real-world applications of NLP, solving this challenge is crucial to enable broad applicability. A promising direction for many tasks is to reformulate them (e.g., by appending an instruction such as "translate into French") so that they can directly be solved by a pretrained language model (Radford et al., 2019; Schick and Schütze, 2020a; Brown et al., 2020). The key idea of PET (Schick and Schütze, 2020a), one such approach aimed at text classification, is to rephrase each input as a cloze question for which the language model's prediction can somehow be mapped to a label; an example is illustrated in Figure 1. While PET achieves remarkable results with little or no labeled training data, manually defining the required mapping between a language model's predictions and labels is difficult as it requires both task-specific knowledge and an understanding of the language model's inner workings to identify words that it understands sufficiently well.

In this work, we show how this mapping can be obtained automatically, removing the need for expert knowledge: We introduce PET *with Automatic Labels* (PETAL), a simple approach for identifying words that can serve as proxies for labels given small amounts of training data. At its core, our approach breaks the intractable problem of finding the mapping that maximizes the likelihood of the training data into several manageable subproblems. Integrating our approach into PET significantly outperforms regular supervised training and almost matches the performance of PET with a manually defined mapping.

2 Related Work

Reformulating problems as language modeling tasks has been explored in fully unsupervised settings (Radford et al., 2019; Puri and Catanzaro, 2019; Davison et al., 2019), in few-shot scenarios with limited amounts of training data (Opitz, 2019; Shwartz et al., 2020; Brown et al., 2020), and even in high-resource settings (Raffel et al., 2019). The same idea is also commonly used for probing the knowledge contained within pretrained language models (Petroni et al., 2019; Talmor et al., 2019; Schick and Schütze, 2020b; Ettinger, 2020, *inter alia*).

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

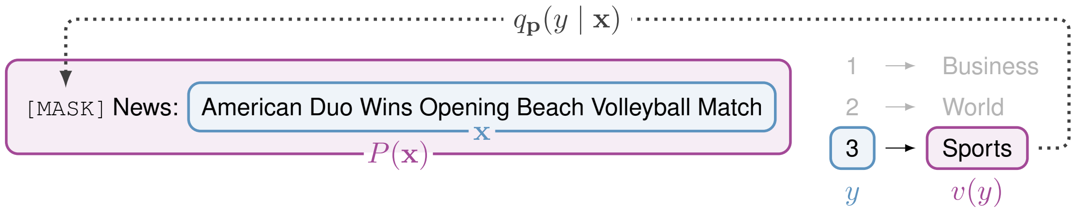
$q_{\mathbf{p}}(y \mid \mathbf{x})$
[MASK] News:
American Duo Wins Opening Beach Volleyball Match
$\mathbf{x}$
$P(\mathbf{x})$
1 → Business
2 → World
3 → Sports
y
$v(y)$

Figure 1: Exemplary application of a pattern-verbalizer pair $\mathbf{p} = (P, v)$: An input $\mathbf{x}$ is converted into a cloze question by applying P. The probability $q_{\mathbf{p}}(y \mid x)$ of each label y is derived from the probability of its verbalization $v(y)$ being a plausible choice for the masked position.

Our method is a direct extension of PET (Schick and Schütze, 2020a) and is similar in spirit to *automatic verbalizer search* (AVS) introduced therein. AVS is another method for automatically finding a mapping from labels to words that works as follows: First, the mapping is initialized by assigning a random word to each label and then, the mapping is improved over multiple iterations by successively replacing words with better alternatives given the current mapping in a greedy fashion. In contrast, our approach offers a closed-form solution that is conceptually simpler and faster, requires fewer hyperparameters – which can be crucial in a data-scarce scenario – and performs much better, especially for difficult tasks.

For PET, expert knowledge is mostly encoded in the mapping from a language model's prediction to labels, which is why we focus on automating this part. The complementary problem of automatically transforming inputs *before* processing them with a language model has been studied by Jiang et al. (2019). This is also closely related to approaches for extracting patterns in relation extraction (Brin, 1999; Agichtein and Gravano, 2000; Batista et al., 2015; Bouraoui et al., 2020).

3 Pattern-Exploiting Training

We review Pattern-Exploiting Training (PET) as proposed by Schick and Schütze (2020a). Let M be a pretrained masked language model (MLM), T its vocabulary and `[MASK]` $\in T$ the mask token. We consider the task of mapping textual inputs $\mathbf{x} \in X$ to some label $y \in Y$ where we assume w.l.o.g. that $Y = \{1, \ldots, k\}$ for some $k \in \mathbb{N}$. In addition to training data $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, PET requires a set of *pattern-verbalizer pairs* (PVPs). As exemplified in Figure 1, each PVP $\mathbf{p} = (P, v)$ consists of

- a *pattern* P that is used to convert inputs to cloze questions. Formally, $P : X \to T^*$ is defined as a function that maps each input to a sequence of tokens containing exactly one `[MASK]` token;
- a *verbalizer* $v : Y \to T$ that maps each label to a single token representing its meaning. For PET to work, the verbalizer must be chosen so that for each input $\mathbf{x} \in X$, $v(y)$ is a suitable replacement for the mask token in $P(\mathbf{x})$ if and only if y is the correct label for $\mathbf{x}$. We call $v(y)$ the *verbalization* of y and abbreviate it as v_y.

Based on this intuition, Schick and Schütze (2020a) define the conditional probability distribution $q_{\mathbf{p}}$ of Y given X as

$$q_{\mathbf{p}}(y \mid \mathbf{x}) = \frac{\exp M(v_y \mid P(\mathbf{x}))}{\sum_{i=1}^{k} \exp M(v_i \mid P(\mathbf{x}))} \tag{1}$$

where $M(t \mid P(\mathbf{x}))$ denotes the raw score that M assigns to t at the masked position in $P(\mathbf{x})$; that is, the probability of y being the correct label for $\mathbf{x}$ is derived from the probability of its verbalization v_y being the "correct" token at the masked position in $P(\mathbf{x})$.

PET basically works in three steps:

1. For each PVP $\mathbf{p}$, a separate MLM is finetuned on $\mathcal{T}$, using the cross entropy between the true labels y_i and $q_{\mathbf{p}}(y_i \mid \mathbf{x}_i)$ as loss function.
2. The resulting ensemble of finetuned MLMs is used to annotate a large set of unlabeled examples with soft labels.

3. Another pretrained language model with a sequence classification head is finetuned on the resulting soft-labeled dataset; this model serves as the final classifier for the task considered.

There are several additional details to PET (e.g., an additional language modeling objective to prevent catastrophic forgetting); we skip these details as they are not relevant to our approach. For a more thorough explanation, we refer to Schick and Schütze (2020a).

4 Likelihood Ratio Verbalizer Search

Manually defining the verbalizer $v : Y \to T$ required for PET can be challenging: It requires knowledge not only of a task's labels and how they can best be expressed in natural language using a single word, but also of the used MLM's capabilities as it is crucial to choose only such words as verbalizations that are understood sufficiently well by the language model and correspond to a single token in its vocabulary. We thus aim to automatically find a good verbalizer v for some pattern P without requiring task- or model-specific knowledge.

Our method requires sets $\mathcal{V}_y \subseteq T$ of *verbalization candidates* for each label $y \in Y$; for now, we simply assume $\mathcal{V}_y = T$ for all y. Let $\mathcal{V}$ be the set of all verbalizers consistent with these candidate sets, i.e., $v \in \mathcal{V}$ if and only if $v_y \in \mathcal{V}_y$ for all $y \in Y$. A natural criterion for measuring the suitability of a verbalizer v is to compute the likelihood of the training data given v, leading to the maximum likelihood estimate

$$\hat{v} = \operatorname*{arg\,max}_{v \in \mathcal{V}} \prod_{(\mathbf{x},y) \in \mathcal{T}} q_{(P,v)}(y \mid \mathbf{x}) \tag{2}$$

Unfortunately, iterating over $\mathcal{V}$ to find the best verbalizer is intractable: the number of possible verbalizers $|\mathcal{V}| = |T|^k$ grows exponentially in the number of labels and for a typical MLM, T contains tens of thousands of tokens.

To circumvent this problem, we reframe the k-class classification task as k one-vs-rest classifications: For each $y \in Y$, we search for a verbalization v_y that enables M to distinguish examples with label y from examples with *any* other label. To this end, we introduce binarized training sets $\mathcal{T}_y = \{(\mathbf{x}_1, \tilde{y}_1), \ldots, (\mathbf{x}_n, \tilde{y}_n)\}$ where $\tilde{y}_i = 1$ if $y_i = y$ and 0 otherwise. For $t \in T$, we define

$$q_{(P,t)}(1 \mid \mathbf{x}) = \frac{\exp M(t \mid P(\mathbf{x}))}{\sum_{t' \in T} \exp M(t' \mid P(\mathbf{x}))} \tag{3}$$

analogous to Eq. 1 except that we consider *all* tokens $t' \in T$ for normalization, and $q_{(P,t)}(0 \mid \mathbf{x}) = 1 - q_{(P,t)}(1 \mid \mathbf{x})$. This enables us to formulate (and compute) the maximum likelihood estimate for each verbalization v_y independently as

$$\hat{v}_y = \operatorname*{arg\,max}_{v_y \in \mathcal{V}_y} \prod_{(\mathbf{x},\tilde{y}) \in \mathcal{T}_y} q_{(P,v_y)}(\tilde{y} \mid \mathbf{x}) \tag{4}$$

However, this reframing creates a label imbalance: If $\mathcal{T}$ is balanced, each $\mathcal{T}_y$ contains $k-1$ times as many negative examples as positive ones. To compensate for this, we raise each $q_{(P,v_y)}(\tilde{y} \mid \mathbf{x})$ to the power of

$$s(\tilde{y}) = \begin{cases} 1 & \text{if } \tilde{y} = 1 \\ n_y / (|\mathcal{T}| - n_y) & \text{otherwise} \end{cases} \tag{5}$$

where n_y is the number of examples in $\mathcal{T}$ with label y. A similar fix for this imbalance problem was suggested by Lee et al. (2001) for multi-class classification with support vector machines.

We next reformulate maximizing the likelihood as minimizing the cross entropy between $\tilde{y}$ and $q_{(P,v_y)}(\tilde{y} \mid \mathbf{x})$, that is, $\hat{v}_y = \arg\min_{v_y \in \mathcal{V}_y} L_{\text{CE}}(\mathcal{T}; v_y)$ where

$$L_{\text{CE}}(\mathcal{T}; v_y) = - \sum_{(\mathbf{x},\tilde{y}) \in \mathcal{T}_y} s(\tilde{y}) \cdot \log q_{(P,v_y)}(\tilde{y} \mid \mathbf{x}) \tag{6}$$

This can easily be derived from Eq. 4 after compensating for the label imbalance as described above. Unfortunately, there is the following problem with Eq. 6: As the vocabulary T is quite large for most pretrained MLMs, $q_{(P,v_y)}(0 \mid \mathbf{x})$ will almost always be close to 1 and thus, $\log q_{(P,v_y)}(0 \mid \mathbf{x}) \approx \log 1 = 0$. This means that negative examples contribute almost nothing to this cross entropy loss, so optimizing for L_{CE} results in verbalizations $\hat{v}_y$ that are *overall highly likely*, but do not necessarily reflect the meaning of y. We fix this problem by considering not the absolute values of $q_{(P,v_i)}(\tilde{y} \mid \mathbf{x})$, but the likelihood *ratio* (LR):

$$L_{\text{LR}}(\mathcal{T}; v_y) = -\sum_{(\mathbf{x},\tilde{y}) \in \mathcal{T}_y} s(\tilde{y}) \cdot \log \frac{q_{(P,v_y)}(\tilde{y} \mid \mathbf{x})}{q_{(P,v_y)}(1 - \tilde{y} \mid \mathbf{x})} \tag{7}$$

Independently, this LR criterion was recently shown to compare favorably to cross entropy in gradient-based neural network training for image classification (Yao et al., 2020).

To arrive at L_{LR}, we have made quite a number of modifications to our starting point, the intractable maximum likelihood estimate. However, the two objectives are in fact quite similar. The key difference is that Eq. 2 enforces a large distance between $M(v_y \mid P(\mathbf{x}))$ and the *maximum* score assigned to the verbalizations of other labels, whereas Eq. 7 enforces a large distance between $M(v_y \mid P(\mathbf{x}))$ and the *average* score assigned to the verbalizations of other labels; this is shown in Appendix A.

4.1 Verbalization Candidates

Our above formulation requires sets of verbalization candidates $\mathcal{V}_y$ for each $y \in Y$. These candidate sets can trivially be obtained by setting $\mathcal{V}_y = T$, but to facilitate verbalizer search, we create candidate sets $\mathcal{V}_y \subset T$ containing only a small subset of the vocabulary. First, we follow Schick and Schütze (2020a) and reduce T by removing all tokens that do not correspond to real words or do not contain at least 2 alphabetic characters. From the remaining list, we collect the 10,000 tokens that occur most frequently in the task's unlabeled data and denote this filtered vocabulary by T_f.

As our loss formulation in Eq. 7 considers the likelihood *ratio*, it is indifferent to the overall likelihood of a token. To make sure that candidates are both syntactically and semantically plausible for a given pattern, we further restrict the set of candidates by keeping only tokens that maximize the likelihood of all *positive examples*: For each label $y \in Y$, we define a candidate set $T_{f,y}$ that contains the 1000 tokens $t \in T_f$ that maximize $L_{\text{CE}}(\mathcal{T}_y^+; t)$ where $\mathcal{T}_y^+ = \{(\mathbf{x}, \tilde{y}) \in \mathcal{T}_y \mid \tilde{y} = 1\}$. Naturally, this induces a bias towards frequent words. As recently shown by Schick and Schütze (2020b), pretrained language models tend to understand frequent words much better than rare words, so all other things being equal, a frequent word should be preferred over a rare word as verbalization; that is, this bias towards frequent words is indeed desirable.

4.2 Multi-Verbalizers

For some tasks, it makes sense to assign multiple verbalizations to some label.[2] This applies all the more if the verbalizations are found automatically, as it may easily occur that the most likely verbalizations for a given label cover different aspects thereof. We thus introduce the concept of *multi-verbalizers*, a generalization of verbalizers to functions $v : Y \to \mathcal{P}(T)$ where $\mathcal{P}(T)$ denotes the power set of T. To integrate multi-verbalizers into PET, we replace the conditional probability distribution in Eq. 1 with

$$q_{\mathbf{p}}(y \mid \mathbf{x}) = \frac{\exp\left(\frac{1}{|v_y|} \sum_{t \in v_y} M(t \mid P(\mathbf{x}))\right)}{\sum_{i=1}^{k} \exp\left(\frac{1}{|v_i|} \sum_{t \in v_i} M(t \mid P(\mathbf{x}))\right)} \tag{8}$$

That is, we substitute the raw score that M assigns to a label's verbalization in standard PET with the average score across all its verbalizations.

[2]For example, one of the categories in the AG's News classification dataset (Zhang et al., 2015) is "Science/Tech" which can best be modeled by using two verbalizations "Science" and "Tech".

Label	**CE**	**LR** ($\mathcal{V}_y = T$)	**LR** ($\mathcal{V}_y = T_{f,y}$)
Society	the, The, reader	Medieval, tradition, Biblical	Dictionary, historical, Bible
Science	Your, the, The	PLoS, biomedical, phylogen	scientists, Physics, scientist
Health	Your, the, reader	Patients, health, Health	health, Health, clinical
Education	reader, Your, FAQ	Libraries, library, bookstore	library, teacher, Teachers
Computer	reader, the, FAQ	toolbar, linux, gcc	Linux, hardware, software
Sports	reader, Your, the	Racing, Motorsport, Sporting	sports, Sports, NASCAR
Business	reader, Your, the	leases, leasing, mortgages	estate, property, finance
Entertainment	reader, Your, the	Movie, fandom, Film	Movie, casting, DVD
Relationship	the, reader, The	couples, Marriage, girlfriends	couples, Marriage, psychologist
Politics	the, The, Your	DOJ, Constitutional, ACLU	Constitutional, ACLU, Federal

Table 1: Most likely verbalizations for the Yahoo Questions dataset obtained using **CE** and **LR** with different candidate sets

Label	**AVS** ($\mathcal{V}_y = T_f$)	**LR** ($\mathcal{V}_y = T_{f,y}$)
Contradiction	insists, Kings, insist, contrary, disagree, Nor, Boris, maintains, Oliver, asserts	but, yet, whereas, Yet, except, unless, But, reason, unfortunately, However
Neutral	sales, Detroit, revenue, earliest, roads, artwork, designs, revenues, walls, Square	she, he, both, god, meaning, ok, Abdul, Georgia, ad, significant
Entailment	prompted, contacted, randomly, monitor, database, Register, requested, investigating, investigate, printer	Register, Computer, Yes, Yeah, Alan, Sure, Clear, Any, Through, Howard

Table 2: Most likely verbalizations for the MNLI dataset obtained using **AVS** and **LR**. Suitable verbalizations are underlined.

5 Experiments

For our experiments with PETAL, we use the PET implementation of Schick and Schütze (2020a) and follow their experimental setup. In particular, we use RoBERTa-large (Liu et al., 2019) as underlying MLM, we use the same set of hyperparameters for PET, the same evaluation tasks with the same patterns, and the same strategy for downsampling training sets. We deviate from Schick and Schütze (2020a) in that we convert all inputs to single sequences (i.e., we remove all `[SEP]` tokens) as we found this to slightly improve the verbalizers found by our approach in preliminary experiments. To ensure that our results are comparable with previous work and improvements in PET's performance are not simply due to this modification of patterns, we do so only for finding verbalizers and not for actual PET training and inference.

We first analyze the verbalizers found by our method qualitatively. To this end, we consider Yahoo Questions (Zhang et al., 2015), a dataset consisting of questions and answers that have to be categorized into one of ten possible categories such as "Health", "Sports" and "Politics". We use the simple pattern

$$P(\mathbf{x}) = \texttt{[MASK]} \text{ Question: } \mathbf{x}$$

and 50 training examples, meaning that we provide just five examples per label. Table 1 shows the most likely verbalizations obtained for all labels using L_{CE} and L_{LR}; for the latter, we consider both an unrestricted set of verbalization candidates and the candidate sets defined in Section 4. As can be seen, L_{CE} does not lead to useful verbalizers for the reason outlined in Section 4: it only identifies words that are overall highly likely substitutes for the `[MASK]` in $P(\mathbf{x})$. While L_{LR} with $\mathcal{V}_y = T$ finds reasonable verbalizers, some verbalizations are rather uncommon tokens ("PLoS", "phylogen",

Method	Yelp	AG's	Yahoo	MNLI	Avg.
supervised	44.8	82.1	52.5	45.6	56.3
PET + random	49.3	83.4	47.0	49.2	57.2
PET + AVS	55.2	**85.0**	58.2	52.6	62.8
PETAL (joint)	**56.5**	84.9	61.1	60.9	65.9
PETAL (sep)	55.9	84.2	**62.9**	**62.4**	**66.4**
PET + manual	<u>60.0</u>	<u>86.3</u>	<u>66.2</u>	<u>63.9</u>	<u>69.1</u>

Table 3: Accuracy of six methods for $|\mathcal{T}| = 50$ training examples. Avg: Average across all tasks. Underlined: best overall result, bold: best result obtained without using additional task-specific knowledge

"gcc"); using more restrained candidate sets ($\mathcal{V}_y = T_{f,y}$) mitigates this issue and finds words that, in most instances, correspond well to the task's actual labels. The shown verbalizations also illustrate the benefit of using multi-verbalizers. For example, the verbalizations for "Computer" include "hardware" and "software"; in isolation, none of these terms fully covers this category, but their combination does cover most of its aspects.

Next, we consider the more challenging MNLI dataset (Williams et al., 2018), a natural language inference dataset where given two sentences $\mathbf{x}_1$ and $\mathbf{x}_2$, the task is to decide whether both sentences contradict each other, one sentence entails the other, or neither. On this dataset, Table 2 compares PETAL to AVS, the approach of Schick and Schütze (2020a) for automatically finding verbalizers, using the pattern

$$P(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1? \texttt{ [MASK]}, \mathbf{x}_2$$

and 50 labeled training examples. While both approaches clearly fail to find good verbalizations for the label "Neutral", using PETAL results in much better verbalizations for the other two labels, with most of the words identified by AVS being entirely unrelated to the considered labels.

To evaluate our approach quantitatively, we use the Yelp Review Full Star (Yelp) and AG's News (AG's) datasets (Zhang et al., 2015) in addition to Yahoo Questions and MNLI. The task for Yelp is to guess the number of stars (ranging from 1 to 5) that a customer gave to a restaurant based on their textual review; for AG's, one of the four categories "World", "Business", "Sports" and "Science/Tech" has to be assigned to a news article.

Following Schick and Schütze (2020a), we again consider a scenario where we have $|\mathcal{T}| = 50$ labeled training examples and a set of $10\,000 \cdot k$ unlabeled examples for each task; the unlabeled examples are only required for PET and not used for finding a verbalizer. For our approach, we consider both a variant where verbalizers are computed for each pattern separately (sep), and a variant were a single verbalizer is computed for *all* patterns as in AVS (joint); for the latter, the likelihood ratio losses for all patterns are simply added up and minimized jointly. We use a multi-verbalizer $\hat{v}$ where $\hat{v}(y)$ are the $n_v = 10$ most likely verbalizations per label and compare PETAL to the following baselines:

- **supervised**: Regular supervised learning without PET, i.e., we add a regular sequence classification head on top of the pretrained language model and perform finetuning as in Devlin et al. (2019).

- **PET + random**: We generate a multi-verbalizer by randomly choosing 10 words per label uniformly from T_f. We include this baseline to verify that any improvements over supervised learning are not simply due to PET using additional unlabeled examples and auxiliary objectives, but that the actual source of improvement is the improved verbalizer.

- **PET + AVS**: We generate a multi-verbalizer with 10 labels per word using automatic verbalizer search with its default parameters.

- **PET + manual**: We consider the manually defined verbalizers of Schick and Schütze (2020a). This serves as an upper bound of what is achievable by incorporating task- and model-specific knowledge.

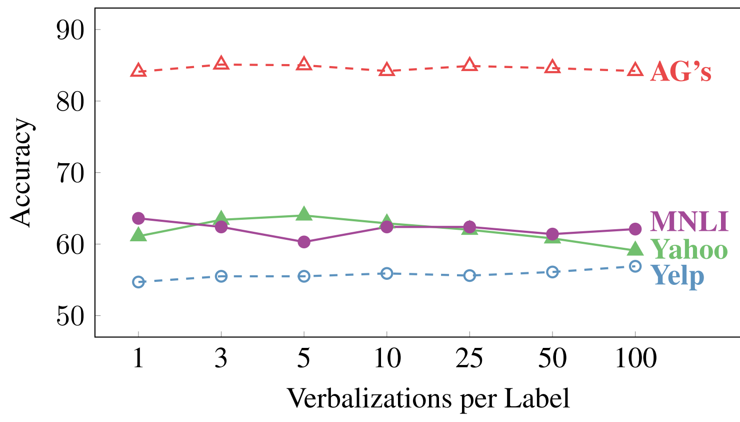

Figure 2: Performance of PETAL (sep) on all four tasks as a function of the number of verbalizations per label (n_v)

Results can be seen in Table 3. On average, PET with random verbalizers performs slightly better than regular supervised learning; we surmise that this is due to PET leveraging additional unlabeled data. Random verbalizers perform much worse than AVS which, in turn, is cleary outperformed by our method for 3 out of 4 tasks, with an especially large margin on MNLI. This holds true for both the joint and sep variant of PETAL, with the latter performing slightly better on average. Furthermore, especially for MNLI, our approach almost matches the performance of PET with manually defined mappings while requiring no task-specific knowledge for finding verbalizers. The large gap between supervised learning and PETAL is especially surprising given that the patterns – the only other source of task-specific knowledge in PET – are very generic in nature.

We finally note that our method adds a single hyperparameter to PET: the number of verbalizations per label n_v, which may be difficult to optimize for small training sets. However, as shown in Figure 2, results on all tasks are relatively stable for a wide range of values ranging from 1 to 100; the best result across all tasks is obtained for $n_v = 3$.

6 Conclusion

We have devised PETAL, a simple approach that enriches PET with the ability to automatically map labels to words. Qualitative and quantitative analysis shows that our approach is able to identify words that are suitable to represent labels with as little as 50 examples and almost matches the performance of hand-crafted mappings for some tasks. For future work, it would be interesting to see whether the patterns required by PET can similarly be obtained in an automated fashion.

Acknowledgements

This work was supported by the European Research Council (grant #740516).

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, page 85–94, New York, NY, USA. Association for Computing Machinery.

David S. Batista, Bruno Martins, and Mário J. Silva. 2015. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 499–504, Lisbon, Portugal, September. Association for Computational Linguistics.

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*.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In Paolo Atzeni, Alberto Mendelzon, and Giansalvatore Mecca, editors, *The World Wide Web and Databases*, pages 172–183, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

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, November. 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, June. 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*, 8:34–48, Jan.

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

Yoonkyung Lee, Yi Lin, and Grace Wahba. 2001. Multicategory support vector machines. Technical report, Department of Statistics, University of Madison, Wisconsin.

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.

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.

Fabio Petroni, Tim Rocktäschel, 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. *Computing Research Repository*, arXiv:1912.10165.

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

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

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

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

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

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. *Computing Research Repository*, arXiv:2004.05483.

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

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.

Hengshuai Yao, Dong-lai Zhu, Bei Jiang, and Peng Yu. 2020. Negative log likelihood ratio loss for deep neural network classification. In Kohei Arai, Rahul Bhatia, and Supriya Kapoor, editors, *Proceedings of the Future Technologies Conference (FTC) 2019*, pages 276–282, Cham. Springer International Publishing.

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.

A Relation of Maximum Likelihood Estimate and One-Vs-Rest Likelihood Ratio

We analyze the impact of all modifications introduced in Section 4: reframing k-class classification as k one-vs-rest classifications, downsampling negative examples and replacing L_{CE} with L_{LR}. For the sake of conciseness, we drop the condition on $\mathbf{x}$ and $P(\mathbf{x})$ in $q_{\mathbf{P}}(y \mid \mathbf{x})$ and $M(y \mid P(\mathbf{x}))$, respectively. We start by reformulating the maximum likelihood estimate in Eq. 2 as

$$\hat{v} = \arg\min_{v \in \mathcal{V}} -\sum_{(\mathbf{x},y)\in\mathcal{T}} \log q_{(P,v)}(y) \tag{9}$$

through logarithmization and multiplication by -1. By applying the definition of $q_{\mathbf{P}}$, we obtain

$$\begin{aligned}
\hat{v} &= \arg\min_{v \in \mathcal{V}} -\sum_{(\mathbf{x},y)\in\mathcal{T}} \log\left(\frac{e^{M(v_y)}}{\sum_{i=1}^{k} e^{M(v_i)}}\right) && (10)\\
&= \arg\min_{v \in \mathcal{V}} -\sum_{(\mathbf{x},y)\in\mathcal{T}} \left(\log(e^{M(v_y)}) - \log(\sum_{y'\in Y} e^{M(v_{y'})})\right) && (11)\\
&= \arg\min_{v \in \mathcal{V}} -\sum_{(\mathbf{x},y)\in\mathcal{T}} \left(M(v_y) - \log(\sum_{y'\in Y} e^{M(v_{y'})})\right) && (12)
\end{aligned}$$

Finally, we can derive from the tangent line approximation $\log(a+b) \approx \log a + b/a$ that the left part of each addend is a soft approximation of $\max_{y'\in Y} M(v_{y'})$ (also commonly referred to as *LogSumExp*), so we can approximate $\hat{v}$ as

$$\hat{v} \approx \arg\min_{v \in \mathcal{V}} -\sum_{(\mathbf{x},y)\in\mathcal{T}} \left(M(v_y) - \max_{y'\in Y} M(v_{y'})\right) \tag{13}$$

We now consider the verbalizer obtained using L_{LR} as in Eq. 7, for which we assume that $\mathcal{T}$ is a balanced dataset. That is, for each label $y \in Y$, there are $|\mathcal{T}|/k$ examples with label y in $\mathcal{T}$. We abbreviate the set $Y \setminus \{y\}$ of all labels except y as $Y_{\setminus y}$.

As L_{LR} for each verbalization v_y is independent of all verbalizations for other labels, we can simply write the optimization criterion for $\hat{v}$ as the sum of likelihood ratio losses for all verbalizations:

$$\hat{v} = \arg\min_{v \in \mathcal{V}} -\sum_{y\in Y} \sum_{(\mathbf{x},\tilde{y})\in\mathcal{T}_y} s(\tilde{y}) \cdot \log \frac{q_{(P,v_y)}(\tilde{y})}{q_{(P,v_y)}(1-\tilde{y})} \tag{14}$$

As can be seen in the definition of $\mathcal{T}_y$, each $(\mathbf{x}, y) \in \mathcal{T}$ contributes to the above sum k times: $k-1$ times as negative example $(\mathbf{x}, 0) \in \mathcal{T}_{y'}$ for each $y' \neq y$, and once as a positive example $(\mathbf{x}, 1) \in \mathcal{T}_y$. We can thus rewrite the above as

$$\hat{v} = \arg\min_{v \in \mathcal{V}} -\sum_{(\mathbf{x},y)\in\mathcal{T}} \left(s(1) \cdot \log \frac{q_{(P,v_y)}(1)}{q_{(P,v_y)}(0)} + \sum_{y'\in Y_{\setminus y}} s(0) \cdot \log \frac{q_{(P,v_{y'})}(0)}{q_{(P,v_{y'})}(1)}\right) \tag{15}$$

and again use the fact that $q_{(P,t)}(0) \approx 1$ for all $t \in T$ as well as the definition of $q_{(P,t)}$ and s to obtain:

$$\hat{v} \approx \arg\min_{v \in \mathcal{V}} - \sum_{(\mathbf{x},y) \in \mathcal{T}} \left(\log q_{(P,v_y)}(1) - \sum_{y' \in Y_{\setminus y}} s(0) \cdot \log q_{(P,v_{y'})}(1) \right) \tag{16}$$

$$= \arg\min_{v \in \mathcal{V}} - \sum_{(\mathbf{x},y) \in \mathcal{T}} \left(\log \frac{e^{M(v_y)}}{\sum_{t \in T} e^{M(t)}} - \frac{1}{k-1} \sum_{y' \in Y_{\setminus y}} \log \frac{e^{M(v_{y'})}}{\sum_{t \in T} e^{M(t)}} \right) \tag{17}$$

Using $\log(a/b) = \log a - \log b$ and the fact that $\sum_{t \in T} e^{M(t)}$ is independent of v, we can further simplify:

$$\hat{v} \approx \arg\min_{v \in \mathcal{V}} - \sum_{(\mathbf{x},y) \in \mathcal{T}} \left(\log e^{M(v_y)} - \frac{1}{k-1} \sum_{y' \in Y_{\setminus y}} \log e^{M(v_{y'})} \right) \tag{18}$$

$$= \arg\min_{v \in \mathcal{V}} - \sum_{(\mathbf{x},y) \in \mathcal{T}} \left(M(v_y) - \frac{1}{k-1} \sum_{y' \in Y_{\setminus y}} M(v_{y'}) \right) \tag{19}$$

$$= \arg\min_{v \in \mathcal{V}} - \sum_{(\mathbf{x},y) \in \mathcal{T}} \left(M(v_y) - \operatorname*{avg}_{y' \in Y_{\setminus y}} M(v_{y'}) \right) \tag{20}$$

This concludes our verification of the statement made in Section 4: Eq. 2 enforces a large distance between $M(v_y)$ and the *maximum* score of other verbalizations, whereas Eq. 7 penalizes their *average* score.