Semantic-Oriented Unlabeled Priming for Large-Scale Language Models

Anonymous ACL submission

Abstract

Due to the high costs associated with finetuning large language models, various recent works propose to adapt them to specific tasks without any parameter updates through in-context learning. Unfortunately, for in-context learning there is currently no way to leverage unlabeled data, which is often much easier to obtain in large quantities than labeled examples. In this work, we therefore investigate ways to make use of unlabeled examples to improve the zero-shot performance of pretrained language models without any finetuning: We introduce Semantic-Oriented Unlabeled Priming (SOUP), a method that classifies examples by retrieving semantically similar unlabeled examples, assigning labels to them in a zero-shot fashion, and then using them for in-context learning. We also propose *bag-of-contexts* priming, a new priming strategy that is more suitable for our setting and enables the usage of more examples than fit into the context window.

1 Introduction

In recent years, there has been a trend in NLP towards larger and larger language models (LMs) (Radford et al., 2018, 2019; Raffel et al., 2020; Brown et al., 2020; Fedus et al., 2021). Different from prior pretrained LMs that are typically finetuned for specific downstream tasks using labeled training datasets (Devlin et al., 2019; Liu et al., 2019), recent work proposes to use such large models in zero- or few-shot settings without any finetuning (Brown et al., 2020; Sanh et al., 2021) due to the often prohibitive costs associated with training, storing and deploying large models (Strubell et al., 2019). In particular, Brown et al. (2020) propose *priming* where training examples are simply provided as additional context together with test examples; this *in-context learning* does not require updating the parameters of the model.

In prior work on in-context learning, only labeled examples are used for priming (Brown et al., 2020; Lu et al., 2021; Kumar and Talukdar, 2021; Min et al., 2021; Jiang et al., 2021). But in many settings, these are extremely scarce or even entirely unavailable, while unlabeled examples can easily be accessed. Unfortunately, there is currently no way to leverage unlabeled examples for priming. Other approaches for leveraging unlabeled data such as domain-adaptive pretraining (Gururangan et al., 2020) would again require finetuning.

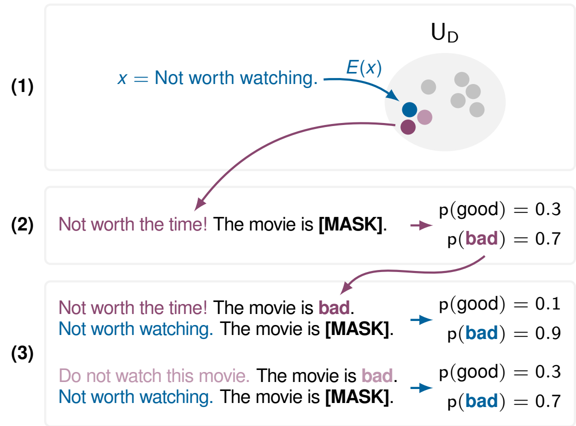

Figure 1: Schematic representation of the steps involved in SOUP for binary sentiment classification of movie reviews. (1) **Semantic Search**: For a given input x, we retrieve semantically similar, unlabeled examples from a set U_D using a sentence encoder E. (2) **Self-Prediction**: We obtain zero-shot predictions for all similar examples using natural language prompts. (3) **Bag-of-Contexts Priming**: We use the retrieved examples along with their most probable labels one at a time as in-context examples to obtain predictions for x; the resulting distributions over possible labels are finally averaged.

Therefore, we investigate how we can make use of unlabeled examples to improve the performance of large-scale language models without requiring changes to their parameters: We propose a self-supervised method called Semantic-Oriented Unlabeled Priming (SOUP), which uses unlabeled examples for in-context learning. Following the observation that semantically similar examples are better candidates as in-context examples than dissimilar ones (Gao et al., 2021a; Liu et al., 2021), we first retrieve the semantically most similar unlabeled examples as contexts for a given input; then, we query the language model to obtain predictions for these unlabeled examples, and finally provide them along with their most likely labels as additional context. Intuitively, this approach is particularly helpful whenever the retrieved examples are easier to classify then the actual input of interest.

Whereas in prior work, the in-context examples and test example are usually concatenated to form a single input that is provided to the LM, we propose to use one in-context example at a time and compute a weighted average of the so-obtained label distributions to obtain a final prediction. Besides resulting in much better performance, one benefit of this methods is that we are no longer constrained by the maximum sequence length of the used LM and thus, more neighbors can be used for priming than with the usual, concatenation-based approach. We also investigate an iterative variant of our approach where predictions for unlabeled examples are iteratively improved with SOUP. On four English text classification datasets, we show that SOUP improves performance of pretrained LMs.

2 Related Work

First proposed by Brown et al. (2020), in-context learning has been studied by many recent works (Lu et al., 2021; Kumar and Talukdar, 2021; Min et al., 2021; Jiang et al., 2021). Concurrent with our work, Min et al. (2021) also propose to perform priming with individual examples and combine the resulting predictions; however, they use a different combination technique and, similar to all prior work on in-context learning, only investigate settings with labeled examples. Our approach is also related to various approaches that leverage unlabeled data in few- or zero-shot settings (Xie et al., 2019; Gururangan et al., 2020; Schick and Schütze, 2021a), but all of them require finetuning the underlying language model.

We make use of different Transformer-based sentence encoders (Reimers and Gurevych, 2019; Gao et al., 2021b) and of textual instructions to improve model performance, an approach that was first proposed by Radford et al. (2019) and has since been investigated extensively (Schick and Schütze, 2021a,b,c; Gao et al., 2021a, i.a.).

3 Semantic-Oriented Unlabeled Priming

We introduce Semantic-Oriented Unlabeled Priming (SOUP), our approach for in-context learning with unlabeled examples. To this end, let M be a masked language model (Devlin et al., 2019) where for some sequence of tokens $t_1, \ldots, t_k$ that contains exactly one mask token, $M(t \mid t_1, \ldots, t_k)$ denotes the probability that M assigns to t at the masked position.[1] Further, let E be a sentence encoder where $E(x)$ denotes the representation assigned to x by E, and D_U be a set of unlabeled examples. We consider a text classification setup where for a given input x, a label y from a set Y has to be predicted.

Obtaining predictions for x with SOUP consists of the following steps:

1. **Semantic Search**: We search for unlabeled examples that are semantically most similar to x using the sentence encoder E.

2. **Self-Prediction**: We use M to obtain predictions for these neighboring examples.

3. **Bag-of-Contexts Priming**: We use the neighbors and their estimated labels as additional context for priming M and compute an average of the resulting label distributions to obtain a final prediction for x.

3.1 Semantic Search

Similar to prior work (Gao et al., 2021a; Liu et al., 2021), the unlabeled examples $x_u \in D_U$ are encoded to obtain vector representations $E(x_u)$; this can be done in advance for the entire set D_U. We also compute the representation $e(x)$ of our test example and use semantic search to find the k nearest neighbors of x according to a specific similarity measure (e.g., cosine similarity). We denote the set of neighbors as $N_x = \{x_1, ..., x_k\} \subseteq D_U$.

3.2 Self-Prediction for Unlabeled Examples

We use M to predict the label distribution for each $x_i \in N_x$, which is done similar to prior work by providing a short prompt and assigning meaningful names to all labels (e.g., Radford et al., 2019; Schick and Schütze, 2021a,c). We use the same notation as Schick and Schütze (2021a,c) in that we make use of a *pattern* P that converts inputs x into cloze questions $P(x)$ containing a single mask,

[1] We focus on masked language models, but our approach can easily be transferred to autoregressive language models.

and a *verbalizer* v that maps each label $y \in Y$ to a single token $v(y)$ representing its meaning. We define the probability of y being the correct label for x based on $M\left(v(y) \mid P(x)\right)$, the probability that M assigns to $v(y)$ at the masked position in $P(x)$. We normalize this probability and set

$$p(y \mid x) \propto \frac{M\left(v(y) \mid P(x)\right)}{M\left(v(y) \mid P(\varepsilon)\right)} \quad (1)$$

with ε denoting an empty sequence following prior work (Brown et al., 2020).

3.3 Priming

Let $\hat{N}_x = \{(x_i, \hat{y}_i)\}_{i=1}^k$ be the selected in-context neighbors with their predicted labels. Based on these semantically similar examples, we want to obtain a prediction for x. In the following, let $\hat{P}(x_i)$ denote $P(x_i)$ with the mask token replaced by $\hat{y}_i$.

Concatenation Priming Previous work usually provides all in-context examples at a time to the LM. That is, all examples are concatenated followed by the test example to obtain the input $c = [\hat{P}(x_1), \hat{P}(x_2), ..., \hat{P}(x_k), P(x)]$, which is provided to the LM to get the final prediction. We refer to this variant as CONCAT priming.

Bag-of-Contexts Priming We propose *bag-of-contexts* (BOC) priming where instead, we only use individual examples for priming and prediction each time and then compute the average of the resulting label distributions as the final prediction. The key advantage of this method lies in the fact that it allows us to use more examples than fit in the context window of the used model.

For each in-context example $x_i \in N$, we construct a corresponding context $c_i = [\hat{P}(x_i); P(x)]$, similar to CONCAT with $k = 1$. For each c_i, we then use the LM to obtain a distribution $q_i(y)$ over possible labels $y \in Y$ for x, where we employ normalization analogous to Eq. 1. Finally, we make use of a weighting function $w(x_i) : N \to \mathbb{R}^+$ and compute

$$q_f(y) = \frac{1}{Z} \cdot \sum_{i=1}^{k} w(x_i) \cdot q_i(y) \quad (2)$$

with $Z = \sum_{i=1}^{k} w(x_i)$. We obtain the final prediction for x as $\hat{y} = \arg\max_{y \in Y} q_f(y)$. We experiment with the following two weighting functions. *uniform*: $w(x_i) = 1$. *similarity-based*: $w(x_i)$ is the cosine similarity between x_i and x.

3.4 Iterative SOUP

We also experiment with an iterative variant of SOUP where the labels for the unlabeled examples in D_U are iteratively refined. To this end, we treat each example $x_u \in D_U$ as a test example: We use SOUP to reclassify x_u with $D_U \setminus \{x_u\}$ as the set of unlabeled examples. This means for each example x, we select in-context neighbors from $D_U \setminus \{x_u\}$ as priming contexts to allow us to refine the prediction for x. We can repeat this process for multiple iterations.

4 Experiments

Datasets We evaluate SOUP on four English datasets: IMDb (Maas et al., 2011) and Yelp Reviews (Zhang et al., 2015) for sentiment analysis as well as AG's News and Yahoo Questions (Zhang et al., 2015) for text categorization. For each dataset, we use one of the the patterns and verbalizers introduced by Schick and Schütze (2021a); further details can be found in Appendix A. For IMDb, the unlabeled in-context examples are selected from the training set of SST-2 (Socher et al., 2013) following Liu et al. (2021). For all other datasets, the in-context examples are obtained from the respective training sets.[2]

Experimental Setup For our main experiments, we use *ALBERT-xlarge-v2* (Lan et al., 2020) as underlying LM and *paraphrase-MiniLM-L6-v2* (Reimers and Gurevych, 2019) as sentence encoder. As the context window of ALBERT is 512 tokens, we truncate each example to 120 tokens for CONCAT. To enable a fair comparison between both priming strategies, we also set the maximum token number for BOC to 120. We compare SOUP to zero-shot performance using only the patterns and verbalizers ("prompt only"), similar to Radford et al. (2019) and Schick et al. (2021). We do not compare to other baselines as we are not aware of other approaches that enable leveraging unlabeled data in zero-shot settings *without finetuning*. For iterative SOUP, we use 3 iterations to improve the labels assigned to unlabeled data.

Results As shown in Table 1, when using CONCAT with $k = 3$, our method clearly performs worse than the prompt-only baseline. However, using our proposed BOC approach consistently out-

[2]To ensure a resource-friendly evaluation, we restrict both the unlabeled sets and the test sets to a maximum of 10,000 randomly selected examples.

	k	$w(x_i)$	AG's	Yahoo	IMDb	Yelp
Prompt only	–	–	66.01	48.04	72.67	43.37
SOUP (CONC.)	3	–	43.88	21.96	54.71	29.56
SOUP (BOC)	3	unif.	68.18	45.64	68.30	40.43
		sim.	68.18	45.57	68.31	40.43
	10	unif.	69.64	49.93	71.03	44.05
		sim.	69.74	49.98	71.01	43.93
	50	unif.	69.70	**52.67**	72.97	**46.21**
		sim.	**70.00**	52.56	72.95	46.20
iSOUP (BOC)	50	unif.	69.88	45.22	**73.78**	45.79

Table 1: Accuracy with zero-shot prompting, SOUP with CONCAT and BOC as well as iterative SOUP (iSOUP) using different numbers of neighbors (k) and both uniform ("unif.") and similarity-based ("sim.") weighting.

Size	Method	AG's	Yahoo	IMDb	Yelp
xlarge	Prompt only	66.01	48.04	72.67	43.37
xlarge	SOUP	69.70	52.67	72.97	**46.21**
xxlarge	Prompt only	73.51	57.89	76.67	45.84
xxlarge	SOUP	**74.89**	**61.82**	**79.54**	41.00

Table 2: Performance of a prompt-only baseline and SOUP with $k = 50$ and uniform weighting using different model sizes

performs not only priming with CONCAT by a large margin, but also leads to consistent improvements over our baseline on three out of four datasets for $k \geq 10$. Moreover, performance grows consistently with the number of in-context examples, with $k = 50$ resulting in improvements for each dataset considered. On average, similarity-based weighting leads to negligible gains over uniform weighting. For our iterative variant of SOUP, we therefore only experiment with uniform weighting; iterative SOUP leads to slight improvements for two tasks, but performs much worse than SOUP for Yahoo.

5 Analysis

We examine the influence of both increasing the language model's size and replacing the Sentence Transformer with different encoders on the performance of SOUP. We also briefly discuss the efficiency of our method.

Model Size We first focus on the impact of model size on the performance of SOUP; to this end, we also evaluate our method (with $k = 50$ and uniform weighting) and the prompt-only baseline using ALBERT-xxlarge-v2 (Lan et al., 2020), a model that is about four times as large as ALBERT-xlarge-v2. As shown in Table 2, for our prompt-only baseline performance consistently improves with model

Sentence Encoder	AG's	Yahoo	IMDb	Yelp
paraphrase-MiniLM-L6-v2	69.70	52.67	72.97	**46.21**
msmarco-bert-base-dot-v5	**69.93**	**53.04**	**74.47**	45.82
unsup-simcse-roberta-large	69.76	52.40	73.90	45.19

Table 3: SOUP (ALBERT-xlarge-v2, $k = 50$, uniform weighting) is robust to choice of sentence encoder.

size for both methods. With exception of ALBERT-xxlarge-v2 on Yelp, for which our method surprisingly leads to worse performance, SOUP consistently outperforms the baseline method.

Sentence Encoder We also investigate the impact of the sentence encoder on downstream task performance. As *paraphrase-MiniLM-L6-v2* was trained on a mixture of tasks that has some overlap with the tasks we evaluate on, we additionally consider *msmarco-bert-base-dot-v5* (Reimers and Gurevych, 2019), a model that was trained exclusively on MS MARCO passages (Bajaj et al., 2018), and *unsup-simcse-roberta-large* (Gao et al., 2021b), an encoder that was trained in a fully unsupervised fashion. As can be seen in Table 3, the choice of sentence encoder has little influence on performance, illustrating that performance improvements do not come from the encoder being pretrained on downstream task data.

Efficiency One disadvantage of our approach is that the number of required forward passes grows linearly with k. After precomputing encodings and labels for U_D, classifying a single example with $k = 3$ took about 0.6s using a single NVIDIA GeForce GTX 1080Ti; for $k = 10$ and $k = 50$, the required times were 1.5s and 6.8s. However, performance can be improved a lot with decoder-only LMs (e.g., Radford et al., 2018, 2019; Brown et al., 2020), as this enables the precomputation of contextualized representations for each $x_u \in U_D$.

6 Conclusion

We have presented SOUP, a method for *unlabeled priming* that classifies inputs by retrieving semantically similar unlabeled examples, classifying these examples in a zero-shot fashion and providing them as additional contexts for in-context learning. Beyond that, we have proposed a new priming strategy that leads to much better performance and scales to more than just a few examples. We have shown that with sufficiently many retrieved examples, SOUP consistently leads to improved performance.

References

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset.

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.

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.

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. 2021a. 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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. Simcse: Simple contrastive learning of sentence embeddings.

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.

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.

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.

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*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *CoRR*, abs/2101.06804.

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.

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.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

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.

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

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.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, 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, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization. *Computing Research Repository*, arXiv:2110.08207.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze questions for few shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, Kyiv, Ukraine (Online). International Committee on Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. Few-shot text generation with pattern-exploiting training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021c. 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.

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*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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.

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.

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 Dataset Details

For each task except IMDb, we use one of the patterns and verbalizers introduced by Schick and Schütze (2021a). In the following, we describe in detail the patterns and verbalizers used.

IMDb For the IMDb Large Movie Review Dataset (Maas et al., 2011), the task is to estimate the binary sentiment of a movie review based on the review's text. We use the following pattern and verbalizer for an input review a:

$$P(a) = a. \text{ The movie is [MASK].}$$
$$v(0) = \text{bad} \quad v(1) = \text{good}$$

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 use the following pattern for an input text a:

$$P(a) = a. \text{ In summary, the restaurant is [MASK].}$$

As a verbalizer v, we define:

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

AG's News AG's News (Zhang et al., 2015) is a task to classify a news article as belonging to one of the categories *World* (1), *Sports* (2), *Business* (3) or *Science/Tech* (4). We define the following pattern for an input news text a:

$$P(a) = a. \text{ News Category: [MASK].}$$

Intuitively, we use a verbalizer that maps 1–4 to "World", "Sports", "Business" and "Science", respectively.

Yahoo Yahoo Questions (Zhang et al., 2015) is a text classification dataset. Given a question and an answer, the text has to be classified to one of ten possible categories. We make use of the following pattern for a input question a and an answer b:

$$P(a, b) = a\ b. \text{ Question Category: [MASK].}$$

Our verbalizer maps labels 1–10 to the tokens "Society", "Science", "Health", "Education", "Computer", "Sports", "Business", "Entertainment", "Relationship" and "Politics".