Multi-Task Pattern Training for Task Generalization

Asaf Maman
Blavatnik School of Computer Science, Tel Aviv University
asafmaman@mail.tau.ac.il

Abstract

In recent years, the appearance of large language models paved the way to calculate predictions via prompts. One key feature of this mechanism is the ability to encode information about the task inside the prompt. This can be accomplished by rewriting input sentences as cloze-style phrases and determining predictions based on the probabilities the model assigns. This method allows high flexibility for multi-tasking, which raises several questions that we aim to answer in this work. We found evidence that training models for multi-tasking, when done right, can be used without harming each task's performance. Furthermore, when training on semantically similar tasks, multi-task training can, to some extent, generalize to unseen tasks.

1 Introduction

When handling classification tasks, the standard method is to fine-tune the model with a task-specific classification head. Recently, when large pre-trained language models such as GPT (Radford and Narasimhan, 2018), BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) started becoming prevalent, utilizing the knowledge attained from pre-training via prompting has become feasible. Providing a task description together with examples has proven successful (Radford et al., 2019) also for zero-shot and few-shot learning, which requires some understanding of the task.

This approach unveils an alternative way to deal with classification tasks: reformulating the input examples as cloze-style phrases. Each example is plugged into a fixed sentence structure, called *pattern*, containing *exactly* one mask token. We chose the prediction according to the scores the model assigns to a subset of tokens, called *verbalizers*, each associated with one of the classes. The model will assign a given input to the class whose verbalizer received the highest score. The method is presented visually in Figure 1.

There are a few advantages to using this mechanism over traditional methods. First, as mentioned before, it provides some sense of task description encoded in the *pattern-verbalizer pair* (PVP). That way, the model is fed with information about the task and may be able to infer it from the PVP. By simply changing the PVP, the model might be able to adapt itself to new tasks without needing a large amount of new-task examples or having to train and initialize task-specific weights. Furthermore, deciding according to MLM predictions makes the fine-tuning closer to the pre-training task and leverages the language modeling knowledge attained for the downstream task.

In this work, we investigate how the performance of masked language models (MLMs) is affected when trained on multiple classification tasks. In addition, we would like to examine whether providing a task description in the form of PVP can help MLMs generalize to *unseen tasks*. Finally, we will test these questions in a few-shot setting using PET pipeline.

Our experiments show that it is possible to train models for multi-tasking using PVPs, and in some cases, training a model for multi-tasking improves performance on unseen tasks. However, the model's ability to infer tasks from PVPs appears to be restricted to tasks that are semantically similar to train tasks.

2 Method

In general, the use of *pattern-verbalizer pair* (PVP) can be formalized as follows: for a given masked language model M, vocabulary V and set of labels $\mathcal{L}$, we define a pattern $P(\mathbf{x})$ that takes as input the

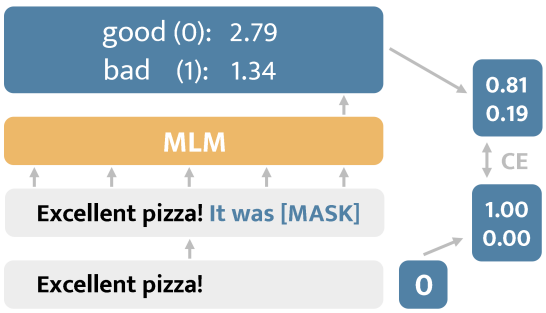

Figure 1: Classification using cloze test — reformulating the sample as a new phrase with exactly one mask token. Then using the scores the language model assigns to the mask token and picking the class with the highest score.

raw sample $\mathbf{x} \in V^*$ and outputs a phrase $P(\mathbf{x}) \in V^*$ that contains exactly one mask token. For each label in $\mathcal{L}$ we define the set of *verbalizers* using the function $v : \mathcal{L} \rightarrow V$ that maps each label to a corresponding token.

We will apply the pattern $P(\mathbf{x})$ on each input $\mathbf{x}$ that we wish to evaluate. Then we will use M to receive the score it assigns to each token $t \in V$ at the masked token position. We denote this unnormalized score as $M(t \mid P(\mathbf{x}))$. Then we'll calculate the unnormalized score for each class label $l \in \mathcal{L}$, that is, $M(v(l) \mid P(\mathbf{x}))$ and denote it by $s_{\mathbf{p}}(l \mid \mathbf{x})$ as the score our language model gives to class label l providing $\mathbf{p} = (P, v)$ as the PVP. We normalize and calculate the probability distribution over the labels using softmax:

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

Finally, we then use the cross-entropy between $q_{\mathbf{p}}(l \mid \mathbf{x})$ and the true one-hot encoded label to calculate the loss over all examples.

3 Experimental Setup

We conducted several experiments to test the questions raised above. In this section, we will describe three setups: Single-task, Multi-task and Few-shot Multi-task training.

The datasets used in the following experiments are AG's News, Yahoo!, Yelp, MNLI and QQP. When selecting datasets, we intended to cover a variety of tasks: text categorization, sentiment analysis, NLI and paraphrase detection. Further details about the datasets are deferred to Appendix A.1.

For each task, we manually picked a set of three PVPs. We present the PVPs for each task in Appendix A.3. To provide further analysis and possibly more insights about the method's behavior, we evaluated each model not only on the PVP that it was trained on but also on all three possible PVPs of the evaluated task.

Additional implementation details can be found in Appendix A.2. We published the code to the GitHub repository[1].

3.1 Unsupervised Baselines. To see how fine-tuning affects each task's performance, we evaluated the zero-shot performance of the pre-trained model as a baseline. The difference between the reported scores of the fine-tuned model and the baseline demonstrates precisely the effect of the fine-tuning on the train tasks.

3.2 Single-task Training. We trained a supervised model for each classification task. Then we evaluated each model on the task it was trained on, but also on all the other tasks. This indicates how training on one single task benefits when switching to a different task in evaluation time.

3.3 Multi-task Training. To test whether multi-tasking will enable generalization to unseen tasks, we combined the training sets of all the datasets and trained a model omitting *one task* each time. The missing task was the one that we aimed to generalize to. We trained each example with a PVP corresponding to the task it belongs to and evaluated all models on each task separately.

3.4 Few-shot Multi-task Training. In Schick and Schütze (2020), the authors presented a method for few-shot training called PET. As an extension for the supervised multi-tasking training, we used PET to train multi-task models with a few examples. For each task, we used 1000 examples to train an ensemble of models and generate a soft-labeled dataset. Finally, we trained one classifier on a 20K size soft-labeled data. This differs from Schick and Schütze (2020) first by training on *multiple* tasks but also by using the method described in Section 2 to train the classifier.

4 Results

Results for single-task training are reported in Table 1, multi-task training in Table 2 and few-shot multi-task training in Table 3.

[1] https://github.com/asafmaman101/amnlp_final_project_code

	AG's News	Yahoo!	Yelp Full	Yelp Polarity	MNLI	QQP	All	Unsup.
AG's News	**93.7** ± 0.06	**81.7** ± 1.78	62.6 ± 6.10	62.7 ± 3.62	57.6 ± 5.06	63.9 ± 4.35	**92.6** ± 0.45	66.7 ± 7.55
Yahoo!	**56.1** ± 1.88	**74.8** ± 0.25	39.3 ± 5.59	38.1 ± 7.25	24.4 ± 2.25	34.0 ± 5.39	**74.1** ± 0.68	43.2 ± 7.66
Yelp Full	27.9 ± 6.25	32.0 ± 2.23	**67.7** ± 0.53	**44.3** ± 3.94	34.1 ± 5.23	35.6 ± 13.9	**66.6** ± 1.58	33.5 ± 11.6
Yelp Polarity	61.9 ± 10.4	59.6 ± 8.50	**97.7** ± 0.26	**97.7** ± 0.05	75.4 ± 4.52	75.5 ± 22.3	**97.3** ± 0.29	67.3 ± 15.7
MNLI	36.0 ± 1.31	35.6 ± 0.42	39.4 ± 9.67	42.1 ± 8.47	**88.4** ± 0.07	**51.8** ± 1.95	**75.6** ± 16.3	38.4 ± 4.34
QQP	44.7 ± 14.5	43.7 ± 12.4	45.7 ± 15.2	41.1 ± 6.71	**77.9** ± 0.74	**85.9** ± 0.31	**68.2** ± 27.7	43.9 ± 9.57

Table 1: **Single-task training performence on all tasks** — reported accuracy results are averaged over all three possible patterns with standard deviation reported aside. Different training tasks are arranged in columns and evaluation tasks in rows. The "All" column corresponds to training on all tasks together and "Unsup." column represents the unsupervised baseline. We emphasized scores that show a significant improvement over the baseline and underlined significant improvement for experiments with different train and test tasks.

4.1 Single-task Training. First, when training and evaluating on the same task, the models gain reasonable scores compared to standard benchmarks. When evaluating single-task models on tasks different from what they were trained on, the results are mixed. Some evaluations show improvement over the unsupervised baseline (*i.e.* training on MNLI and testing on QQP) while some reduced performance (for instance, training on MNLI and testing on Yahoo!).

AG's News on Yahoo! The scores on one of these tasks improved after training on the other. However, this does not directly imply generalization. Even though AG's News and Yahoo! have different distributions and classes, there is an artifact to consider when comparing them — they share two classes in common: Sports and Business. Therefore, we cannot conclude that the model learned to generalize to unseen tasks. This result can be seen more like a domain adaptation than a task generalization.

MNLI on QQP To overcome the issue raised above, we tested paraphrase detection, which has some semantic relationship with NLI. Two phrases with entailment connection might in some cases be paraphrases, while natural and contradiction pairs are not. Although there is some semantic overlap between these two tasks, their labels and the tasks themselves differ.

The performance of the model trained on MNLI when evaluating on QQP shows an average score of 77.9%, *i.e.* 34% more than the baseline on this task. In addition, training a model on QQP achieved 85.9%, meaning that testing the MNLI model on QQP got only 8% less than fine-tuning a model on this task. While this is still far from concluding that the model is inferring tasks from the description, we believe that such results are meaningful considering that we trained the models on entirely different tasks and without seeing examples from the target distribution.

PVP Variance Training and evaluating different PVPs produce very different results for some tasks. For example, QQP was not consistent with different PVPs on the Yelp Polarity task (can be seen in extended results in Table 4), we can see that two of the PVPs scored roughly 88%, but one PVP got 49%. This emphasizes the importance of choosing the right PVP to boost performance and generalization.

Combining all tasks One question we asked was how multi-task training would affect single-task performance. We trained a model on all tasks simultaneously, and the results showed that multi-tasking was not harmful. The model achieves slightly the same results on each task as if it was fine-tuned to this task specifically. On most tasks, the multi-task training reduced the score by roughly 0.5-1.0% per task. The only exception is when evaluating on MNLI and QQP, the average score for this pair is low, but one can notice that the variance is high and that the low average score is a result of one low-performing PVP. Excluding "bad" PVPs, by using a validation set, for instance, can prevent these cases. The performance of the multi-task model without the "bad" PVPs is also very close to the single-task performance.

4.2 Multi-task Training. Testing on the missing task showed different results among the tasks.

AG's News and Yahoo! When omitting AG's News, the model succeeded in getting 82.0% accuracy, an improvement over the baseline. That is, training on other tasks enhances performance over AG's News when never observing this task. However, when comparing this score with the single-

	wo/AG's News	wo/Yahoo!	wo/Yelp Full	wo/MNLI	wo/QQP	All	Unsup.
AG's News	**82.0** ±0.25	93.6 ±0.23	93.4 ±0.16	93.4 ±0.27	92.8 ± 0.85	**92.6** ±0.45	66.7 ±7.55
Yahoo!	75.3 ±0.23	**51.8** ±2.50	75.1 ±0.25	75.1 ±0.12	75.1 ±0.26	**74.1** ± 0.68	43.2 ±7.66
Yelp Full	67.9 ±0.43	67.9 ±0.54	30.5 ±8.51	68.1 ±0.15	68.1 ± 0.37	**66.6** ± 1.58	33.5 ±11.6
MNLI	88.2 ±0.09	88.1 ±0.37	85.2 ±4.37	35.3 ±0.88	88.1± 0.22	**75.6** ± 16.3	38.4 ±4.34
QQP	86.2 ± 0.24	86.1 ± 0.08	86.6 ± 0.19	86.2 ± 0.33	**76.9**± 0.52	**68.2** ± 27.6	43.87 ±9.57

Table 2: **Multi-task training performence on all tasks** — results of models trained on all tasks exept one. Reported accuracy results are averaged over all three patterns with standard deviation reported aside. Omitting different training tasks are arranged in columns (*i.e.* "wo/AG's News" means training on all tasks **except** AG's News) and evaluation tasks in rows. We emphasized scores that show a significant improvement over the baseline.

task training, we see almost the same result (81.7%) for a model trained *only* on Yahoo!. Thus, most of the knowledge learned in the multi-task training got from the Yahoo! task and not from the remaining tasks. Moreover, the similarity artifacts between AG's News and Yahoo! are also taking part in this setup.

Yelp and MNLI On these two tasks, we notice that the scores of the models trained on all other tasks are worse than the baseline model. This probably stems from the fact that the training tasks are too semantically different, and the model struggles to learn to be completely agnostic to the task.

QQP Similar to single-task training, testing multi-task performance on QQP showed a significant improvement over the baseline. Considering the model was not asked to detect paraphrases during training and has not seen paraphrases before evaluation, it is reasonable to claim that the model succeeded in generalizing to this task.

To some extent, training a model for multi-tasks benefits with performance on unseen tasks. However, it still seems that the model's ability to infer the task from PVP is limited to semantically close tasks. Although the models failed to generalize to any particular task in the experiments, this limitation is somewhat expected.

	wo/AG's News	wo/Yahoo!	wo/Yelp Full	wo/MNLI
AG's News	**78.8** ±2.53	91.9 ±0.08	91.9 ±0.09	91.7 ±0.10
Yahoo!	72.3 ±0.33	**52.2** ±1.72	72.1 ±0.13	72.2 ±0.20
Yelp Full	65.5 ±0.20	65.6 ±0.25	33.8 ±5.12	65.5 ±0.24
MNLI	85.7 ±0.20	85.8 ±0.11	85.7 ±0.27	37.3 ±3.27

Table 3: **Few-shot Multi-task training using PET** — results of models trained using PET on all tasks exept one. Reported accuracy results are averaged over all three patterns with standard deviation reported aside. Omitting different training tasks are arranged in columns (*i.e.* "wo/AG's News" means training on all tasks **except** agnews) and evaluation tasks in rows.

4.3 Few-shot Multi-task Training. In this setup, we restricted experiments to four tasks due to the limitation of resources. As in the previous experiment, here too there are mixed results, but similar relationships exist between the tasks.

When omitting AG's news or Yahoo!, for example, the model improves over the baseline as in the last two setups. However, for Yelp Full and MNLI, the model fails to perform without seeing the task at train time.

On the other hand, when comparing the results to those in Table 2, the low performance does not seem to stem from the small number of train examples. Moreover, in comparison to supervised results, PET performs only slightly worse. This result is not surprising and mostly follows the results revealed in Schick and Schütze (2020), but testing it here proves that it can also get the benefits of multi-tasking using the method suggested in this work.

We believe that our result opens the way for exploring ways to use PVPs, for example, adjusting a supervised model for a new task with only a few examples.

5 Conclusion

In this work, we tested how training and evaluating using PVPs affects the model's ability to multi-task and generalize to unseen tasks. The results show that training for multi-tasking is feasible for achieving almost the same performance as single-task fine-tuning for a specific task. However, the ability to generalize to unseen tasks is limited, at least in the experimental setup of the current work. Scores on different tasks did improve when training on entirely different tasks, but we cannot affiliate this improvement with the ability to infer the new task and use existing knowledge for solving it. Still, the presented improvement on unseen tasks can be beneficial in some applications of such models in real-world use cases.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

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.

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.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

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

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2639–2649, Florence, Italy. Association for Computational Linguistics.

Alec Radford and Karthik Narasimhan. 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.

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

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.

A Appendices

A.1 Datasets. The following datasets were chosen to examine the questions raised above. The consideration when picking datasets was to cover different tasks but also to have tasks that share common semantic characteristics. Some datasets were also picked from Schick and Schütze (2020) to control performance on the different tasks.

AG's News A dataset for text categorization which contains 120K train samples and 7.6K test samples. It contains articles formatted as title and text body. The task is to classify each article to one of four categories: (1) World, (2) Sports, (3) Business, (4) Tech.

Yahoo! Text categorization dataset. It contains 1.46M pairs of questions and answers. The train set contains 1.4M samples and the test set 60K samples. The task is to classify each pair to its subject out of 10 possible options: (1) Society (2) Science (3) Health (4) Education (5) Computer (6) Sports (7) Business (8) Entertainment (9) Relationship (10) Politics.

Yelp Yelp Reviews dataset contains 650K train samples and 50K test samples. There are two versions of tasks for this dataset. One, called "Yelp Full" is to classify a review for a 5-star rating. The other version, named "Yelp Polarity" is to predict whether the review sentiment was positive or negative.

MNLI The MNLI dataset contains pairs of sentences to classify between three possible NLI labels: (1) entailment (2) natural (3) contradiction. The dataset has 392K train and 9.8K test samples.

Quora Question Pairs QQP contains 363K train samples and 40K test samples. The task is to predict whether two sentences are paraphrases one of the other.

A.2 Implementation Details. Experiments were implemented in PyTorch (Paszke et al., 2019). We chose the following hyperparameters for the experiments in Section 3. We chose to use Roberta Large across all experiments. It enables us to compare results with Schick and Schütze

(2020). We trained the model with a max sequence size of 256 tokens. We chose the batch size to fit into 11GB GPU infrastructure, which led to a batch size of 2. To simulate larger batch sizes, we accumulated gradients for eight steps to have an effective batch size of 32. We chose the learning rate to be 1e-5. We trained each model for three epochs on the whole train set. The temperature was tuned to a value of 2, weight decay to 0.01, and adam epsilon to 1e-8.

A.3 Patterns. We chose the following datasets to evaluate performance.

AG's News Each sample **x** in the dataset consists of title and text body. That is $\mathbf{x} = (a, b)$ where a and b represents the title and text body respectively. For the experiments, we used the following three patterns:

$P_1(x) =$ ____: a b $\quad P_2(x) =$ ____ News: a b

$P_3(x) = a$ (____) b

The verbalizers used for each class were World, Sports, Business and Tech, corresponding to the dataset categories.

Yahoo Each sample **x** in the dataset consists of a question and an answer. That is $\mathbf{x} = (a, b)$ where a and b represents the question and answer respectively. For the experiments, we used the following three patterns:

$P_1(x) =$ ____: a b $\quad P_2(x) =$ ____ Question: a b

$P_3(x) = a$ (____) b

The verbalizers used for each class were "Society", "Science", "Health", "Education", "Computer", "Sports", "Business", "Entertainment", "Relationship" and "Politics", corresponding to the dataset categories.

Yelp Each sample **x** in the dataset consists of a review a. For the experiments, we used the following three patterns:

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

$P_2(x) = a$. All i n all, it was ____.

$P_3(x) =$ Just ____! a

The Yelp Full and Yelp Polarity were differentiated by the verbalizers used in each task. For Yelp Full, the 1-5 rating scale was affiliated with "terrible", "bad", "okay", "good", "great" verbalizers. Yelp Polarity was flattened to "bad" and "good" only.

MNLI Each sample **x** in the dataset consists of two sentences a and b. The three patterns used for the experiments were:

$P_1(x) =$"a"? ____, "b" $\quad P_2(x) = a$? ____, b

$P_3(x) =$"a"? ____, "b"

And the verbalizers used for each were "Wrong", "Right" and "Maybe" for Contradiction, Entailment and Natural classes, respectively. P_3 differs from P_1 by its verbalizers, it was used with "No", "Yes" and "Maybe" instead.

Quora Question Pairs Each sample **x** in the dataset consists of two sentences a and b. The three patterns used for the experiments were:

$P_1(x) =$"a"? ____, "b" $\quad P_2(x) = a$? ____, b

$P_3(x) =$"a"? ____, "b"

And the verbalizers used for each were "Wrong" and "Right" for sentences that are not paraphrases and are paraphrases, respectively. P_3 differs from P_1 by its verbalizers, it was used with "No" and "Yes" instead.

B Full Results Report

The exact score for each single experiment is described in and Table 4, Table 5 and Table 6.

		AG's News			Yahoo!			Yelp Full			Yelp Polarity			MNLI			QQP			All			Unsup.
		0	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	
	0	93.72	93.58	93.00	80.18	82.94	81.54	64.64	66.84	63.68	58.60	69.34	64.46	59.22	52.60	52.38	58.94	63.00	65.58	93.08	92.66	93.68	65.48
AG's News	**1**	93.36	93.74	90.64	81.24	83.68	82.50	66.58	67.50	66.10	72.34	65.40	72.92	67.62	61.60	65.62	73.98	67.20	78.32	93.54	92.28	92.78	74.72
	2	92.54	90.96	93.62	81.88	81.94	81.38	61.34	63.46	55.80	53.62	64.16	64.14	59.66	49.24	51.90	62.24	52.64	65.46	90.22	93.58	92.34	59.76
	0	58.06	53.86	55.52	75.08	74.98	74.72	37.94	39.66	39.30	34.14	42.72	40.22	25.66	18.56	16.58	29.74	30.20	41.44	74.66	74.30	74.86	42.22
Yahoo!	**1**	57.32	54.30	53.22	74.58	74.82	74.52	43.24	45.48	45.72	37.76	46.42	43.08	32.94	25.66	23.80	41.48	32.30	40.96	75.44	74.18	74.54	51.26
	2	56.26	49.48	55.96	73.76	72.84	74.58	30.12	33.00	34.56	26.86	33.56	33.60	18.02	14.54	21.76	29.82	18.96	40.08	67.82	74.58	73.32	36.02
	0	31.04	30.86	34.32	31.56	35.24	27.88	68.36	67.23	67.42	46.10	46.34	49.44	37.24	41.28	41.74	39.50	47.08	30.02	68.36	67.80	67.94	40.08
Yelp Full	**1**	30.88	32.00	31.28	31.26	34.46	33.10	64.72	67.40	63.52	46.86	46.94	48.22	36.52	37.02	38.88	42.04	47.12	32.78	68.82	66.00	65.40	40.34
	2	20.12	20.22	20.72	20.08	20.08	30.08	41.72	33.36	67.48	38.44	31.38	39.74	24.08	25.44	28.08	27.50	20.60	20.08	64.98	68.62	65.36	20.08
	0	68.88	65.76	76.34	63.92	69.48	59.06	97.82	97.46	97.66	97.68	97.50	97.60	79.48	79.46	82.52	88.98	92.44	66.38	97.64	97.74	98.04	80.24
Yelp Polarity	**1**	67.60	67.00	69.36	59.26	65.14	69.26	97.54	97.40	97.52	97.52	97.68	97.36	77.70	70.56	82.74	83.48	87.78	69.58	97.92	97.20	97.50	71.78
	2	49.80	49.80	49.92	49.84	49.80	49.84	78.04	73.90	97.88	96.76	78.18	97.76	67.02	73.06	76.24	75.26	52.26	49.80	97.46	97.90	97.10	49.86
	0	34.84	33.94	35.42	35.56	35.56	35.60	31.13	31.94	32.56	38.30	36.02	34.62	88.36	87.40	61.46	52.06	50.64	48.40	88.08	60.12	88.52	35.62
MNLI	**1**	35.84	35.86	35.48	35.94	36.04	35.70	35.68	37.14	37.06	38.40	36.20	38.90	86.60	88.36	57.02	51.36	49.74	41.14	88.06	57.16	87.94	36.22
	2	37.44	35.44	37.44	33.94	34.92	35.20	49.24	50.70	50.06	48.18	47.86	51.80	82.12	77.06	88.48	32.66	32.70	53.62	81.78	88.20	81.54	43.42
	0	36.34	36.36	36.66	36.66	36.36	36.34	37.48	37.12	37.00	37.82	37.24	36.90	78.78	76.80	36.46	85.94	85.38	61.06	85.92	36.34	85.60	54.74
QQP	**1**	36.34	36.34	36.38	36.38	36.34	36.34	36.34	36.36	36.60	36.96	36.64	36.52	78.44	77.38	36.34	85.80	85.64	54.86	85.70	36.34	85.44	36.68
	2	55.36	41.44	61.48	60.56	60.74	58.04	57.72	45.58	63.24	45.04	47.14	48.80	75.60	71.58	77.64	63.66	63.64	86.26	85.00	86.16	82.44	40.20

Table 4: **Single-task training full results** — reported accuracy results for each train and evaluation PVP combination. Different training tasks and PVPs are arranged in columns and evaluation tasks and PVPs in rows. The "All" column corresponds to training on all tasks together and "Unsup." column represents the unsupervised baseline without any fine-tuning.

		wo/AG's News			wo/Yahoo!			wo/Yelp Full			wo/MNLI			wo/QQP			Unsup.
		0	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	
	0	81.86	82.64	80.86	93.54	93.20	92.68	93.52	93.26	92.02	93.76	93.34	92.70	93.14	92.84	92.14	65.48
AG's News	**1**	81.88	81.92	79.80	93.16	93.90	91.40	93.06	93.52	90.66	93.16	93.50	92.00	92.70	93.56	93.36	74.72
	2	82.42	80.16	82.32	92.76	90.08	93.48	92.24	91.16	93.24	92.62	91.86	93.22	92.92	91.60	91.92	59.76
	0	75.06	75.14	74.60	54.70	41.98	53.58	75.20	75.06	74.46	75.22	75.16	74.98	74.88	74.28	73.78	42.22
Yahoo!	**1**	74.70	75.48	74.00	54.72	49.98	55.18	75.02	75.42	73.48	74.90	75.26	74.52	74.90	75.20	73.60	51.26
	2	73.92	71.64	75.44	44.68	27.12	50.92	73.86	70.72	74.92	74.62	73.82	75.04	74.24	69.50	75.40	36.02
	0	68.28	66.78	67.98	68.02	66.76	67.72	33.98	40.10	30.94	68.02	68.04	67.52	68.48	67.76	67.76	40.08
Yelp Full	**1**	65.98	67.44	66.36	66.24	67.34	64.90	31.48	36.80	30.24	64.84	68.22	66.74	65.18	68.00	65.64	40.34
	2	64.02	63.00	68.00	66.42	64.00	68.40	20.42	29.46	20.86	54.00	39.48	68.32	63.20	56.42	68.32	20.08
	0	88.22	88.06	52.78	88.30	88.04	58.52	87.60	87.82	53.46	35.50	31.74	34.70	88.14	87.98	57.58	35.62
MNLI	**1**	85.00	88.38	48.34	86.64	88.46	52.98	86.12	87.88	49.50	35.14	34.36	35.34	86.48	87.86	54.80	36.22
	2	81.58	79.76	88.22	81.28	81.28	87.76	81.40	79.54	80.18	36.86	45.66	36.08	83.84	81.58	88.30	43.42
	0	85.96	86.12	36.34	86.14	86.08	36.34	86.54	86.12	36.52	85.84	86.18	38.92	77.40	77.30	36.44	54.74
QQP	**1**	85.24	86.18	36.35	85.92	86.16	36.35	85.46	86.84	36.38	85.52	86.30	36.40	72.16	76.38	36.38	36.68
	2	74.70	80.70	86.44	78.88	81.42	86.28	83.02	80.94	86.48	77.78	84.76	86.48	78.48	77.52	77.10	40.20

Table 5: **Multi-task training full results** — reported accuracy results for each train and evaluation PVP combination. Omitting different training tasks and PVPs are arranged in columns and evaluation tasks and PVPs in rows. The "Unsup." column represents the unsupervised baseline without any fine-tuning.

		wo/AG's News			wo/Yahoo!			wo/Yelp Full			wo/MNLI			Unsup.
		0	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	**0**	**1**	**2**	
AG's News	**0**	77.16	79.32	76.22	91.94	91.66	91.86	91.78	91.80	91.84	91.62	91.72	91.68	65.48
	1	80.70	81.68	79.74	91.88	92.00	91.74	92.00	91.88	91.82	91.98	91.76	91.68	74.72
	2	78.34	80.14	77.44	91.96	91.86	91.84	92.16	92.00	91.96	91.94	91.72	91.82	59.76
Yahoo!	**0**	72.60	71.80	71.92	54.18	46.40	52.98	72.22	72.24	72.02	72.28	71.88	72.48	42.22
	1	72.38	71.94	72.16	56.38	51.30	54.94	72.22	71.98	72.06	72.42	71.96	72.26	51.26
	2	70.86	68.60	72.28	41.30	31.88	51.10	70.70	69.56	72.00	71.70	71.44	72.34	36.02
Yelp Full	**0**	65.72	65.00	65.40	65.72	65.16	65.50	36.22	37.30	34.24	65.82	65.50	65.54	40.08
	1	64.50	65.46	64.42	64.48	65.84	65.00	36.90	37.22	35.34	65.02	65.40	65.28	40.34
	2	64.82	64.18	65.32	65.18	64.60	65.36	22.70	31.20	27.90	61.34	52.50	65.40	20.08
MNLI	**0**	85.82	85.54	62.28	85.84	85.36	66.48	85.92	85.56	64.84	35.14	33.16	35.28	35.62
	1	84.76	85.84	58.04	84.40	85.94	62.62	84.04	85.80	61.02	35.98	35.76	35.88	36.22
	2	81.02	79.78	85.48	81.76	81.50	85.72	81.52	80.80	85.40	41.42	42.70	41.08	43.42

Table 6: **Few-shot Multi-task training using PET** — results of models trained using PET on all tasks exept one. Omitting different training tasks are arranged in columns (*i.e.* "wo/AG's News" means training on all tasks **except** agnews) and evaluation tasks in rows. The "Unsup." column represents the unsupervised baseline without any fine-tuning.