A Ethical Considerations and Impact Statement

The experiments in this paper were conducted with publicly available datasets MNLI, QQP, QNLI, SST2, STS-B, MRPC, RTE, CoLA, SciTail, SNLI, DBPedia, TREC, AG News, Yahoo Answers, Yelp Polarity, SST5, IMDB, SQuADv2, and MATH, citing the original authors. As we were not able to determine the license for all used datasets, we have opted to use them in the limited form possible, adhering to the terms of use of the GLUE benchmark for all of the mentioned datasets. As the datasets are commonly used in other related works and were published in scientific works that went through an established review process, we do not check for the presence of any offensive content, as it was already removed by the authors of these publicly available datasets. In addition, we do not utilize any personally identifiable information or offensive content, and we do not perform crowdsourcing in any form for data annotation. To our knowledge, we are not aware of any potential ethical harms or negative societal impacts of our work, apart from the ones related to the field of Machine Learning (i.e., the use of computational resources that are consuming energy and producing heat with indirect CO2 emission production). We follow the license terms for the T5-base model we use – all models and datasets allow their use as part of the research. As we transform conditional generation into the classification problem (generating only labels), we minimize the problem of generating offensive or biased content.

Impact Statement: CO2 Emissions Related to Experiments The experiments in this paper require a significant amount of GPU computing resources as we train and evaluate 1 model over multiple random initializations (10) for different methods (4) and datasets (12). Overall, the experiments, including evaluations (which did not require training, but still used GPU resources for inference) and preliminary experiments (which are not reported in the scope of our work), were conducted using a private infrastructure, which has a carbon efficiency of 0.432 kgCO_2eq/kWh. Approximately 1200 hours of computation were performed on hardware of type A100 PCIe 40GB (TDP of 250W). Total emissions are estimated to be 240.24 kgCO_2eq, of which 0 percent were directly offset. These estimations were conducted using the CodeCarbon [1] Python module. Whenever possible, we tried to reduce the computational costs. Because our method is built upon the prompt tuning PEFT method, we always trained only a small part of the model parameters (76800 parameters, which is around 0.2% of the T5-base model parameters), and training the model fully will probably require more GPU hours and create more CO2 emissions.

B Experimental setup: Further Details

Implementation details. For implementing all of our experiments, we utilize *Python 3.11.8* with the *PyTorch* [5] framework and Huggingface modules (*transformers* [7] for model loading and training, *peft* [4] for PEFT methods initialization, *datasets* [2] for data loading, and *evaluate* for evaluation). We create a single data structure for task prompt vectors that is capable of performing arithmetic operations with soft prompts.

Data splits. We take 1000 samples from the train set and use it as a validation set, and make the test set from the original validation set for datasets that contain over 10000 samples. For datasets with less than or equal to 10000 samples, we do not modify the training set and split the validation set into 2 halves for validation and test sets. We keep the same random seed for subsampling and splitting for all of our experiments.

Hyperparamters setings. We provide all of our configurations in the config directory of our repository. We set soft prompt length to 100 tokens, learning rate to 0.3, and lower the weight decay of the AdamW optimizer [3] to 1×10^{-5} for the T5-base model. We also utilize the Seq2SeqTrainer class from the Huggingface transformers Python module. We train all models on all data sets for 10 epochs, except for TREC, where we train for 50 epochs due to the tendency of models to underfit here. We set different hyperparameters for prompt tuning, and the hyperparameters for zero- or few-shot evaluation do not differ much from those for prompt tuning. We train for 1000 update steps while keeping a batch size of 2 for 5, 10, and 25 shots, 8 for 50, 100, and 250 shots, and 16 for 500, 750, and 1000 shots. In general, we chose the maximum token length for labels by searching the dataset for the maximum token length (in our configs, we set default *max_target_lenght* to 128 if the dataset requires generating sentences). For the inputs, we pad the token sequences to 256 tokens with the *max_target_lenght* parameter. We use a learning rate of 0.3 for the *AdamW* optimizer, with weight decay of 1×10^{-5} and 500 warmup steps for 10 epochs (with an exception for the TREC dataset) with the batch size of 32. We evaluate, log, and save after each 100 training steps and keep only the best model at the end of the training. In our configs, we set the number of tokens to 50, but in reality, the Hugging Face *peft* library doubles the number for encoder-decoder models like T5.

For training the soft prompts with the LLaMa-3.1-8B-Instruct and DeepSeek-LLM-7B-Chat models, we utilized similar hyperparameter settings as for the T5-base model, with the exception of using the cosine function for the learning rate scheduler and the usage of the SFTTrainer class from the Huggingface trl Python module [6] for training.

Since we are utilizing the T5-base for conditional generation, we are computing exact match instead of accuracy for classification. Because we are generating labels also for classification tasks, the exact match is equivalent to accuracy in the sequence classification task. In the scope of our work, we refer to a single dataset as a task.

For the training of multi-task ATTEMPT, we have used hyperparameters and a training environment based on the original implementation. Full hyperparameter settings can be found in the repository. [1] of our replication study of ATTEMPT in the configs directory (files **attempt_tvp*.toml**).

C Additional results: Task Prompt Vectors and Task Prompt Cosine Similarities

In this section, we provide more detailed and disaggregated results from Section 4.2. Figure 1 shows the comparison of cosine similarities across different random initializations of task prompts from prompt tuning. We can see that for all task combinations,

[1] https://github.com/DisAI-Replication-Challenge/ATTEMPT

the highest cosine similarity is for the equal random initializations. Additionally, when comparing different tasks and different random initializations, the cosine similarities are the lowest, which only confirms our finding from Section 4.2.

We repeat the same process of comparing cosine similarities across different random initializations for task prompt vectors in Figure 2. Similarly to task prompts, the highest cosine similarity is for the equal random initializations. We can see that for task prompt vectors, the cosine similarities between different random initializations are higher than task prompts in Figure 1. Similarly to our findings in Section 4.2, we can see that certain task combinations have higher cosine similarities than others. For both of these figures, we can see that task prompts and task prompt vectors from different initializations usually end up at different points in the task sub-space.

In addition, we provide a comparison of aggregated cosine similarities of task prompt and task prompt vectors for s for the LLaMa-3.1-8 B-Instruct and DeepSeek-LLM-7 B-Chat models in Figures 3a, 3b, 4a, and 4b. We can see that the differences between task prompts and task prompt vectors are much smaller compared to the T5 comparison from Section 4.2, which may be caused by relatively small (close to zero) vocabulary embeddings (from which we initialized the soft-prompts).

References

1. Courty, B., Schmidt, V., Luccioni, S., Goyal-Kamal, MarionCoutarel, Feld, B., Lecourt, J., LiamConnell, Saboni, A., Inimaz, supatomic, Léval, M., Blanche, L., Cruveiller, A., ouminasara, Zhao, F., Joshi, A., Bogroff, A., de Lavoreille, H., Laskaris, N., Abati, E., Blank, D., Wang, Z., Catovic, A., Alencon, M., Stęchły, M., Bauer, C., Lucas-Otavio, JPW, MinervaBooks: mlco2/codecarbon: v2.4.1 (May 2024). https://doi.org/10.5281/zenodo.11171501, https://doi.org/10.5281/zenodo.11171501
2. Lhoest, Q., Villanova del Moral, A., Jernite, Y., Thakur, A., von Platen, P., Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L., Davison, J., Šaško, M., Chhablani, G., Malik, B., Brandeis, S., Le Scao, T., Sanh, V., Xu, C., Patry, N., McMillan-Major, A., Schmid, P., Gugger, S., Delangue, C., Matussière, T., Debut, L., Bekman, S., Cistac, P., Goehringer, T., Mustar, V., Lagunas, F., Rush, A., Wolf, T.: Datasets: A community library for natural language processing. In: Proceedings of the 2021 Conference on EMNLP: System Demonstrations. pp. 175–184. ACL, Online and Punta Cana, Dominican Republic (Nov 2021)
3. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019), https://openreview.net/forum?id=Bkg6RiCqY7
4. Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., Bossan, B.: Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft (2022)
5. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
6. von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., Gallouédec, Q.: Trl: Transformer reinforcement learning. https://github.com/huggingface/trl (2020)
7. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art

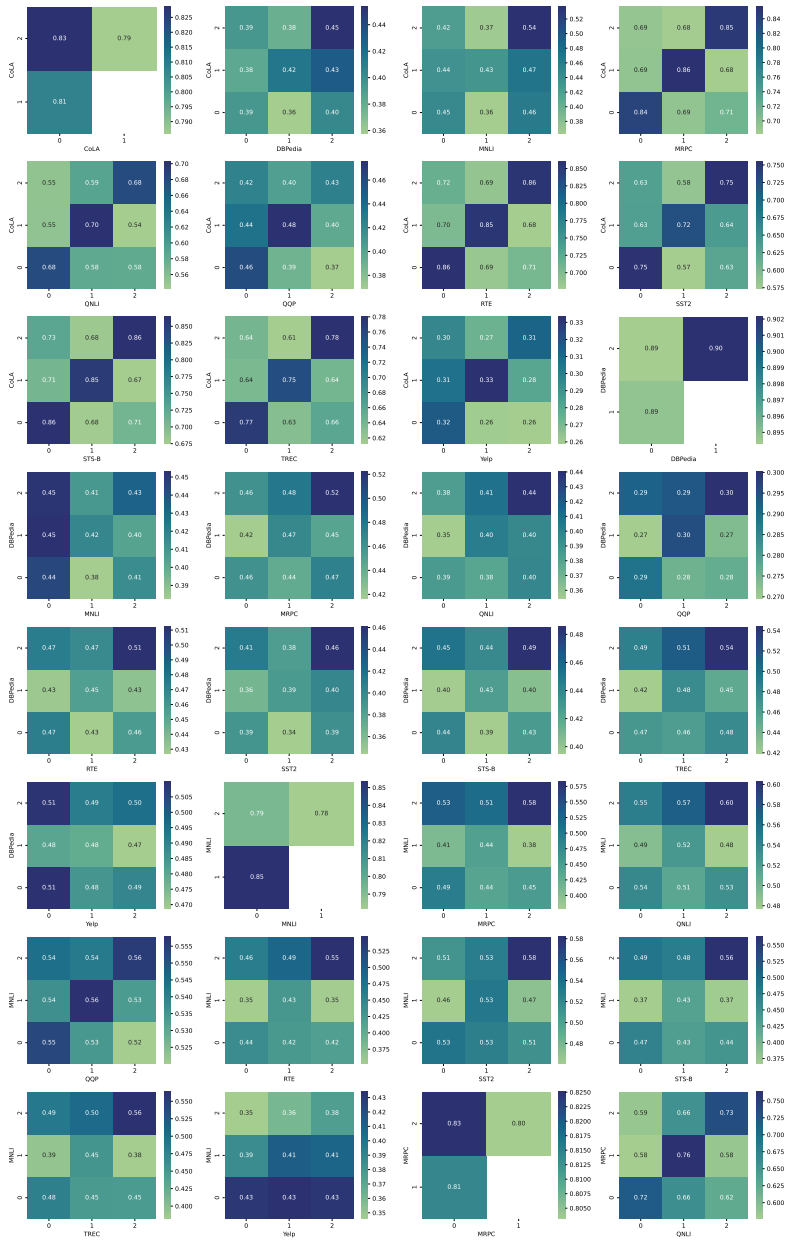


Fig. 1: Comparisons of cosine similarities of *task prompts* fine-tuned on different tasks for T5-base model. Each heatmap represents a different task combination. We calculate the cosine similarities for all combinations of 3 random initializations, omitting the combinations of random initializations where cosine similarity is equal to 1 (single-task comparisons). Each heatmap is represented as a single field in Figure 2a by averaging all values. The x and y axes represent the number of random initializations.

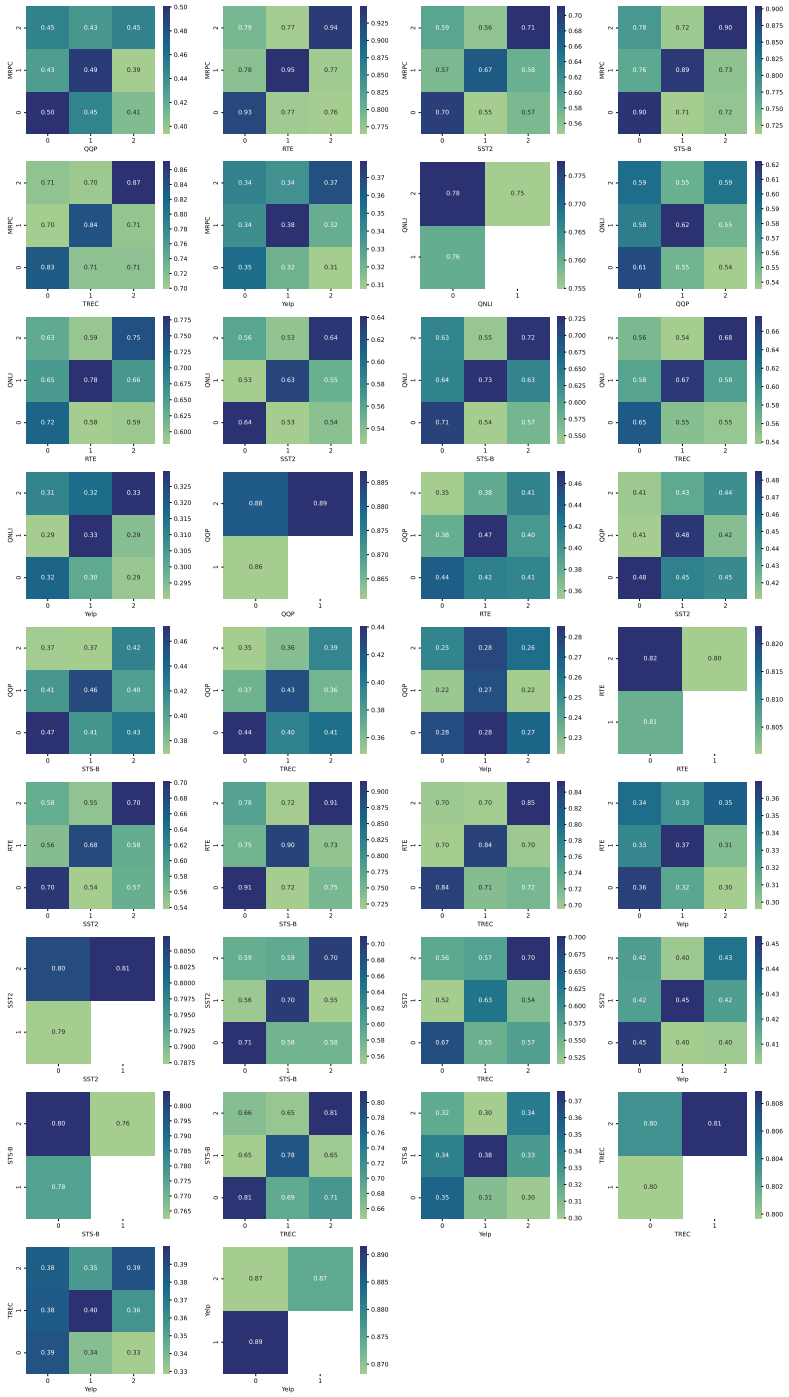


Fig. 1 (cont.): Continuation of Figure 1 for additional tasks.

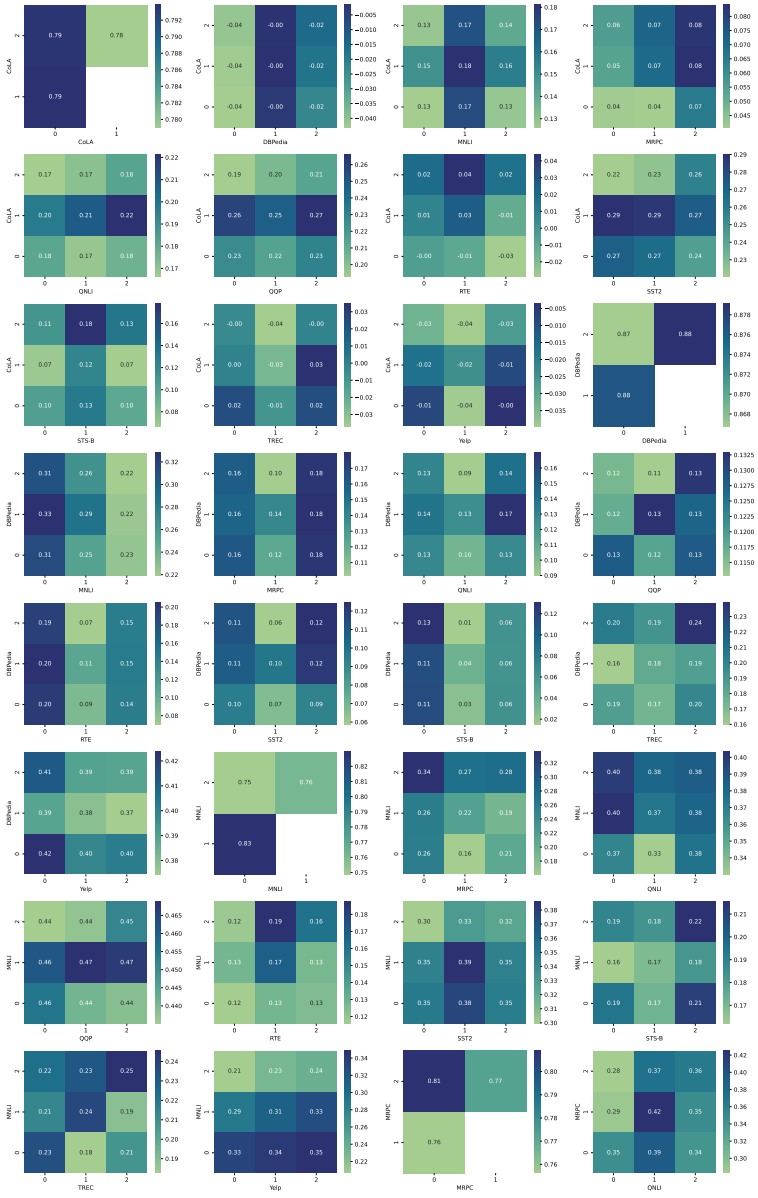


Fig. 2: Comparisons of average cosine similarities of *task prompt vectors* for T5-base model. The averages are calculated similarly to Figure 1 but with task prompt vectors created from different task prompts. Each heatmap represents a different task combination. We calculate the cosine similarities for all combinations of 3 random initializations, omitting the combinations of random initializations where cosine similarity is equal to 1 (single-task comparisons). Each heatmap is represented as a single field in Figure 2b by averaging all values. The x and y axes represent the number of random initializations.

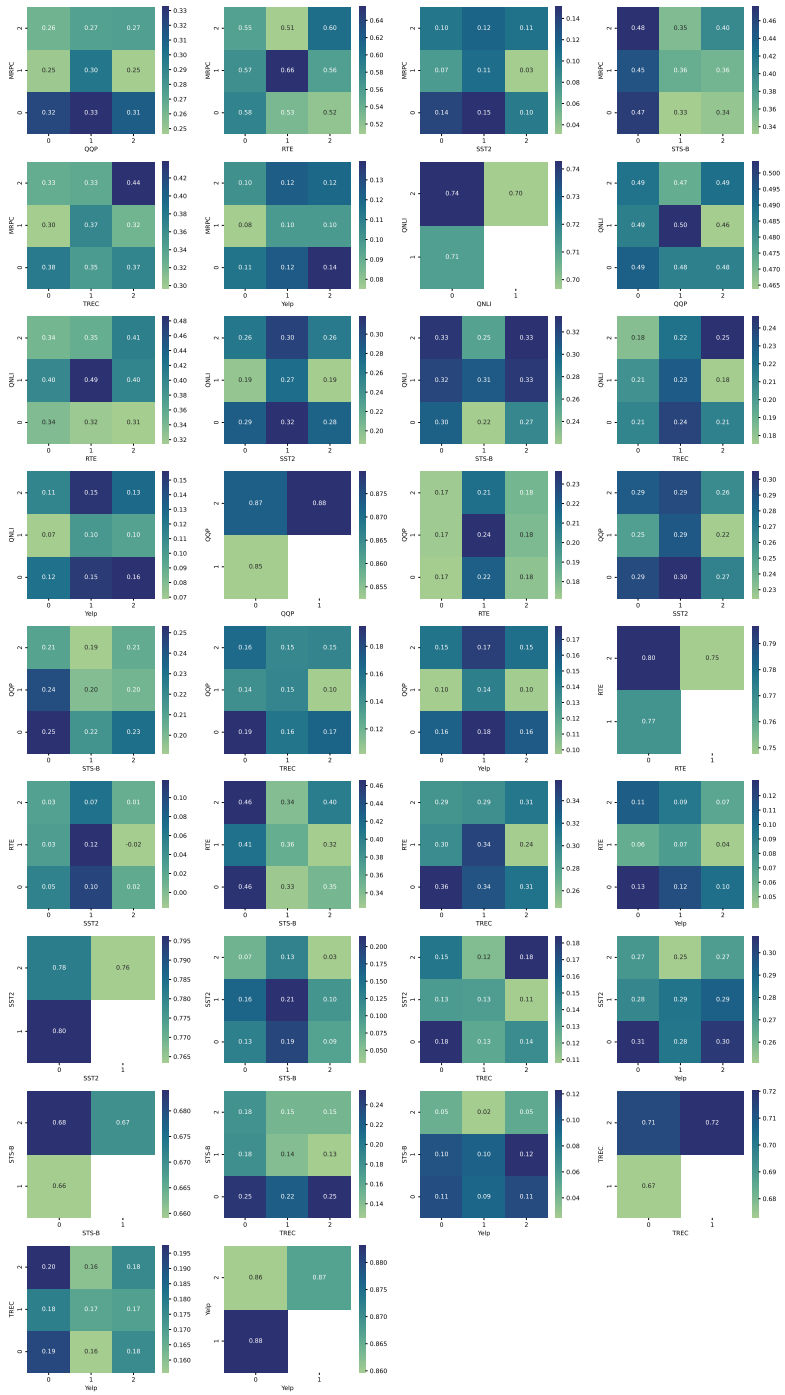
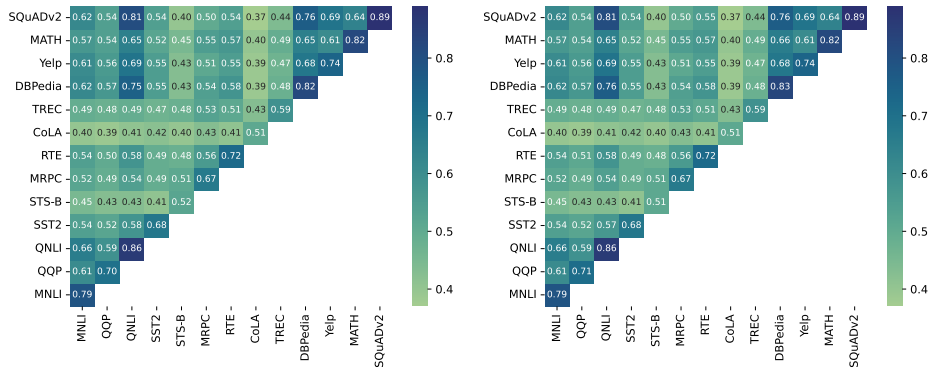
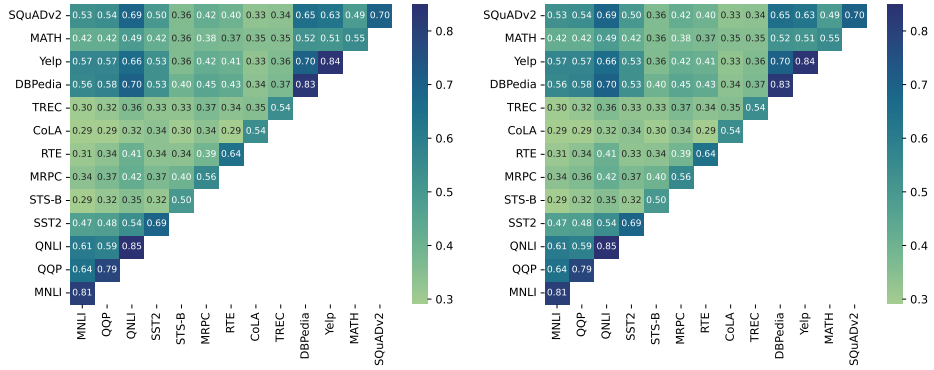


Fig. 2 (cont.): Continuation of Figure 2 for additional tasks.



(a) Cosine similarities of *task prompts*. (b) Cosine similarities of *task prompt vectors*.

Fig. 3: Comparison of average cosine similarities between task prompts and task prompt vectors fine-tuned on different tasks for the LLaMa-3.1-8B-Instruct model. The average is calculated across all combinations of 3 random initializations (i.e., row QNLI column MNLI was calculated as the average of all cosine similarities between MNLI and QNLI task prompts for all random initialization combinations, omitting the combinations where cosine similarity is equal to 1). The diagonal represents the cosine similarities of the same tasks, and it represents the maximum value of cosine similarity across different random initializations.



(a) Cosine similarities of *task prompts*. (b) Cosine similarities of *task prompt vectors*.

Fig. 4: Comparison of average cosine similarities between task prompts and task prompt vectors fine-tuned on different tasks for the DeepSeek-LLM-7B-Chat model. The average is calculated across all combinations of 3 random initializations (i.e., row QNLI column MNLI was calculated as the average of all cosine similarities between MNLI and QNLI task prompts for all random initialization combinations, omitting the combinations where cosine similarity is equal to 1). The diagonal represents the cosine similarities of the same tasks, and it represents the maximum value of cosine similarity across different random initializations.

natural language processing. In: Proceedings of the 2020 Conference on EMNLP: System Demonstrations. pp. 38–45. ACL, Online (Oct 2020)