



Uncovering the Token Splitting Effect in Soft Prompts for Multi-Model LLM Training

Raphael Reimann ¹ and Frederic Sadrieh ¹

Abstract: Prompt tuning in natural language processing enables efficient utilization of Large Language Models (LLMs), but soft prompts often struggle with interpretability. This study introduces a novel multi-model training methodology for soft prompts, validated across the MultiBERTs collection using IMDb, Emotion, and MNLI datasets. We uncover the token splitting effect in soft prompts, a phenomenon where individual prompt tokens align with specific models within their embedding spaces, significantly impacting performance. Our findings reveal that post-training prompt compression enhances efficiency with minimal performance loss. We thereby advance the understanding of soft prompt behavior in multi-model settings, offering pathways for resource-efficient optimization and strategic compression in Large Language Models.²



Keywords: soft prompts, large language models, multi-model training, interpretability

1 Introduction

Traditionally, pre-trained language models have been used to perform a downstream task by fine-tuning the existing language model on the task, consuming significant computational resources since each task requires a separate model instance [Ra18]. Parameter-efficient fine-tuning provides an alternative to this challenge by fine-tuning a selected subset of model weights or introducing and optimizing a smaller number of new weights trained on the downstream task. Most parameters of the model remain unchanged, making the approach more resource-efficient.

In-context learning offers an alternative by using manually crafted input-output pairs without optimizing model weights. However, it is sensitive to the nuances of prompt design and requires extensive human labor and labeled data to identify the most effective prompts [Li21]. To address these challenges, soft prompts have been introduced, as a technique that optimizes prompts in the model’s embedding space [LAC21]. By design, soft prompts require much fewer parameters than traditional fine-tuning methods but have been shown to match the performance of full model fine-tuning.

The exploration of soft prompts raises several research questions that this paper seeks to address. We investigate the interpretability of soft prompts when trained on multiple distinct language models. We contribute to the broader discourse on optimizing language

1 Hasso-Plattner Institut, Prof.-Dr.-Helmert-Straße 2-3, 14482 Potsdam,
raphael.reimann@student.hpi.de,  <https://orcid.org/0009-0003-4847-3579>;
frederic.sadrieh@student.hpi.de,  <https://orcid.org/0009-0006-3938-8067>

2 Our code is available at <https://github.com/FSadrieh/explainable-soft-prompts>

models in a resource-constrained setting, highlighting the potential of soft prompts as a parameter-efficient approach.

Our main contributions include identifying the token splitting effect and its implications for soft prompt behavior across multiple models, examining the relationship between a token’s position within the prompt and its importance to the model’s performance. We demonstrate the feasibility of post-training prompt compression, which results in a performance penalty but provides an approach for further efficiency gains.

2 Related Work

Prompt Engineering Approaches Li; Liang [LL21] propose prefix-tuning, which freezes language model parameters and trains a continuous prefix for each downstream task. The prefix acts as "virtual tokens" that the model can attend to in each layer, optimizing them across the entire embedding space. Lester et al. [LAC21] suggest prompt-tuning as a simplification, using a single prompt representation by prepending k d -dimensional vectors to the input, where d is the embedding space dimensionality of the pre-trained model, thereby reducing the number of parameters.

Interpretability of Soft Prompts Lester et al. [LAC21] explore the interpretability of soft prompts by mapping soft tokens to their nearest-neighbor natural language token. They find that while the sequence of learned prompts shows little interpretability, some tokens are close to lexically similar clusters from the training task, suggesting that the prompt may prime the model to interpret input in a specific domain. Bailey et al. [Ba23] note that soft prompts occupy distinct regions in the embedding space from natural language tokens, making direct comparisons misleading. Khashabi et al. [Kh22] show that the back-translation of soft prompts can be controlled by optimizing a new loss function, which confines the soft prompt to the neighborhood of a hard prompt with minimal performance loss, allowing one to freely control the interpretability of the soft prompt.

Multi-Model training Zou et al. [Zo23] demonstrate attacking large language models (LLMs) with certain hard prompt suffixes, leading to adverse behavior. Using open-source LLMs like Vicuna, they train suffixes with “Greedy Coordinate Gradient” and “Universal Prompt Optimization” to improve attack performance on black-box models. Training across multiple models results in a suffix that performs well on both trained and out-of-distribution models Zou et al. [Zo23], which motivates exploring multi-model training for soft prompts.

3 Methods

Multi-Model Training We achieve multi-model training by prepending the same soft prompt to the input of multiple models, calculating the average loss across all models, and using this aggregated feedback to update the soft prompt. This process assumes uniformity in embedding dimensions across the selected models.

Dimensionality Reduction and Prompt Token Model Mapping Soft prompts exist in high-dimensional spaces and are not directly interpretable. Dimensionality reduction techniques like t-SNE [VH08], UMAP [MHM18], and PCA [Pe01] reduce this space to 2D for visualization. We first reduce dimensions to 50 using PCA, then apply UMAP or t-SNE for the final reduction to two dimensions. We use a k-nearest neighbor algorithm [FH89] to map prompt tokens to embedding spaces, providing a quantitative approach to evaluate soft prompts. Calculating the k nearest neighbors for each soft prompt token across all embedding spaces, a majority vote assigns each token to an embedding space. We use both cosine similarity and Euclidean distance with $k = 7$.

Token masking To assess token importance, we use token masking by individually replacing each soft prompt token t_i with a zero vector z and measuring the model’s output loss for each replacement. We define token importance for t_i in model m as $\text{importance}(t_i, m) = \text{Loss}_m([t_{<i}, z, t_{>i}]; \text{input})$. A loss increase indicates the token’s importance to the model since it performs poorly without the token. The process is repeated for each model $m \in M$ and token $t \in T$ and thus has a complexity of $O(M \cdot T)$.

Using token importance, we can mask unimportant tokens. Omitting the unimportant tokens completely, results in prompt compression. Compression allows for more input into the model, by condensing the important soft prompt tokens.

4 Experimental Results

Model To enable comparable multi-model training we use the MultiBERT collection [Se22], comprising 25 BERT-base uncased models pre-trained with different seeds on English data, using Masked Language Modeling and Next Sentence Prediction objectives. This allows generalization beyond specific model instances due to variations in embedding spaces and weights but identical architecture [Se22].

Dataset We evaluate our findings using three classification datasets:

The IMDb movie review dataset [Ma11]. The dataset contains movie reviews categorized into positive or negative sentiment (binary classification). We have generated a split of

15,000; 5,000; 5,000 (train, validation, test) from the original 25,000 training instances available through huggingface datasets repository [Lh21].

The emotion dataset [Sa18]. The dataset contains English tweets and assigns one of six emotions to each tweet (“sadness, disgust, anger, joy, surprise, and fear” [Sa18]). The multi-class nature of the dataset allows us to analyze our methods in a different setting compared to IMDb. We have generated a split of 6,000; 5,000; 5,000 (train, validation, test) from the 16,000 train split of the dataset downloaded from huggingface datasets [Lh21].

The mnli dataset, which is part of the glue benchmark [WNB18]. The dataset contains a premise and a hypothesis sentence. We feed them into our Multibert model by concatenating them with a semi-colon (i.e., premise;hypothesis). The label shows whether the premise entails the hypothesis, is neutral, or contradicts it [WNB18]. We have generated a split of 10,000; 5,000; 5,000 (train, validation, test) from a sample of the 393,000 train split of the dataset downloaded from huggingface datasets [Lh21].

Task and Training setup We use masked language modeling for classification, appending a [MASK] token to each example. The label is the tokenized version of the class name. Parts of our training and evaluation pipeline build on a template [DSZ23], incorporating best practices from previous work.

Since performance is not the focus, we limit hyper-parameter tuning. We tuned batch size, learning rate, warmup period, weight decay, beta1, and beta2 on the IMDb dataset using 10 runs with a Bayesian sweep. For other datasets, we adjusted only the learning rate and batch size. All final hyper-parameter configurations can be found in Table 1.

	batch size	learning rate	warmup period	weight decay	beta1	beta2
IMDb	168	0.811	0.04	0.173	0.803	0.939
emotion	180	0.128	0.04	0.173	0.803	0.939
mnli	304	0.266	0.04	0.173	0.803	0.938

Tab. 1: The tuned hyper-parameters per dataset. Note only batch size and learning rate are tuned per model.

Following Lester et al. [LAC21] we prepend 16 tunable tokens to the embedded input and feed it into the model, bypassing the embedding layer. The 16 768-dimensional vectors (MultiBERTs embedding dimension) are seeded and randomly initialized. In addition, we seed the training.

We train soft prompts in 1, 2, 5, and 10 model configurations on each dataset, maintaining the same initialization, training seed, and hyper-parameters while varying model seeds. We also train soft prompts on 5-model configurations with different training and initialization seeds but fixed model seeds. This results in six categories: 1-model, 2-model, 5-model, 5-model-init, 5-model-train, and 10-model. The 1-model category always has 25 soft

prompts, one per model. For IMDB, we trained 5 soft prompts for each category; for other datasets, we trained 3 per category. The first trained soft prompt is the same across 5-model, 5-model-init, and 5-model-train categories.

All reported results are on the test set. We report the mean and standard deviation across multiple runs within each soft prompt category. The best results and those within 2 standard deviations of the best are bolded (following Raffel et al. [Ra20]). All values are rounded to three decimal points.

4.1 Multi-model training

#Model	1	2	5	10
IMDb	0.360 \pm 0.093	0.332 \pm 0.035	0.325 \pm 0.022	0.353 \pm 0.010
emotion	0.494 \pm 0.121	0.506 \pm 0.015	0.735 \pm 0.003	0.886 \pm 0.007
mnli	1.098 \pm 0.014	1.086 \pm 0.005	1.108 \pm 0.009	1.134 \pm 0.011

Tab. 2: The soft prompt test loss on the trained on models based on the amount of models the soft prompt was trained on. Bold are the best losses and those within two standard deviations per dataset.

The difference in test loss between a single-model and a multi-model configuration is small as shown by Table 2. Despite maintaining the same prompt length and increasing the number of models the prompt is trained on, we observe minimal to no significant performance drops when training on multiple models. This is noteworthy because each multi-model soft prompt has fewer tokens per model.

4.2 Token splitting effect

Dimensionality Reduction We apply t-SNE (scikit-learn, random state 1, cosine metric) and UMAP (umap library, n_neighbors 50, cosine metric) for dimensionality reduction of soft prompts and their corresponding model embedding spaces. Figure 1 shows a clear separation of embedding spaces and distribution of soft prompt tokens within these spaces. In addition, Figure 1 shows the prompt tokens only within trained-on models’ embedding spaces. To determine if a token’s location within a model’s embedding space indicates its significance, we use token masking, to assess their impact on model performance.

Token masking Dropping individual tokens in a soft prompt reveals their importance to a model (section 3). Table 3 provides a token importance matrix example. Each model relies on a distinct subset of important tokens, indicated by a considerable loss increase when these tokens are masked. Interestingly, each token is crucial for only one model, and each model has a similar number of important tokens. The effect is strong, with a significant difference between the worst loss for each token and the rest. Unimportant tokens, when masked, have a negligible effect on loss, confirming model-specific token dependencies.

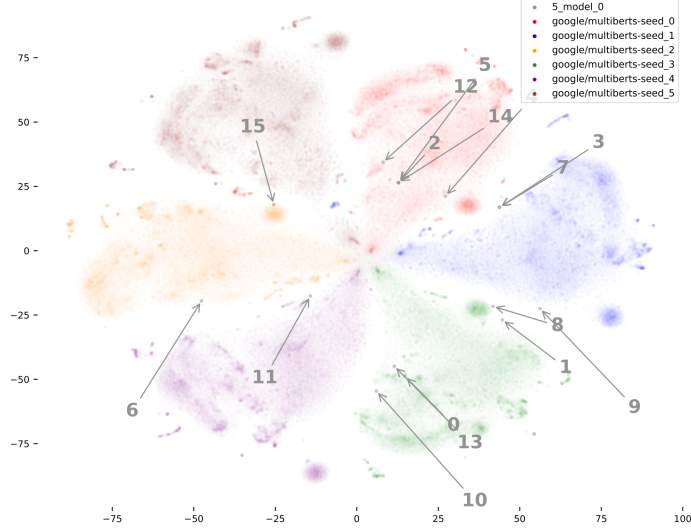


Fig. 1: A dimensionality reduction of the embedding spaces of MultiBERTs 0, 1, 2, 3, 4, 5 and the prompt tokens of 5_model1_0 (trained on IMDb). Note that 5_model1_0 was only trained on the first five models while MultiBERT 5 is an out-of-distribution model. The dimensionality reduction was done using PCA and t-SNE.

Token	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
Model 0	0.314	0.322	0.308	0.307	1.975	0.342	0.305	0.306	0.312	0.305	0.304	0.308	0.305	0.311	0.388	0.304
Model 1	1.064	0.309	0.306	1.779	0.307	0.307	0.307	1.891	0.307	5.025	0.306	0.305	0.307	0.305	0.305	0.305
Model 2	0.316	0.312	0.308	0.309	0.310	0.310	2.732	0.309	0.310	0.310	0.309	0.310	0.310	0.369	0.310	3.310
Model 3	0.315	0.651	0.316	0.315	0.315	0.316	0.313	0.313	0.876	0.317	3.715	0.311	0.311	0.313	0.313	0.315
Model 4	0.305	0.302	1.272	0.301	0.299	0.300	0.299	0.300	0.300	0.300	0.300	3.568	0.369	0.362	0.301	0.301

Tab. 3: The test loss for the soft prompt 5_multibert_0 (trained on IMDb), when each token is masked separately and evaluated on each model. It is clear to which model each token belongs since the test loss (importance) is high for only one model.

Scaling the token splitting effect As shown in Table 4, increasing the number of models while keeping prompt length constant strengthens the token-splitting effect. Fewer tokens lead to higher importance for each token.

Token location vs token performance We evaluate if important tokens lie within their model’s embedding space by aligning token importance (via token masking) with token position (k-nearest neighbor vote). Alignment analysis Table 5, using cosine similarity and Euclidean distance, shows modest values but exceeds the random baseline significantly, especially with more models. The alignment between a token’s position and its significance to the model’s performance increases as the token splitting effect increases.

Dataset	2 Models	5 Models	10 Models
IMDb	0.631 ± 0.237	2.065 ± 0.684	4.777 ± 0.387
emotion	0.75 ± 0.042	1.384 ± 0.05	2.543 ± 0.006
mnli	1.145 ± 0.017	1.251 ± 0.032	1.68 ± 0.089

Tab. 4: The test loss the important tokens achieve when masked, grouped by the number of models the soft prompt was trained on and datasets. The more models share a prompt, with the same prompt length, the higher the importance of the individual tokens.

#Model	random	IMDb		emotion		mnli	
		cos	euc	cos	euc	cos	euc
2	0.5	0.575 ± 0.16	0.55 ± 0.17	0.625 ± 0.051	0.625 ± 0.088	0.625 ± 0.184	0.604 ± 0.193
5	0.2	0.713 ± 0.064	0.725 ± 0.075	0.375 ± 0.135	0.438 ± 0.135	0.354 ± 0.078	0.333 ± 0.078
10	0.1	0.65 ± 0.135	0.638 ± 0.127	0.417 ± 0.029	0.417 ± 0.029	0.271 ± 0.029	0.188 ± 0.051

Tab. 5: Alignment of the token position and performance based on the nearest neighbor vote and token masking. The random column is the score a random classifier would achieve when trying to predict to which model a token should belong. Bold are all results, which beat the random classifier significantly.

#Model	IMDb		emotion		mnli	
	masked	shortend	masked	shortend	masked	shortend
2	2.162 ± 1.756	2.821 ± 2.041	2.393 ± 0.603	2.237 ± 1.287	6.302 ± 0.883	6.597 ± 4.603
5	1.1 ± 0.523	1.416 ± 0.714	5.425 ± 0.512	5.350 ± 2.482	7.88 ± 0.266	8.056 ± 3.992
10	1.972 ± 0.399	4.405 ± 0.861	6.826 ± 0.341	6.830 ± 2.704	10.15 ± 0.429	10.938 ± 2.976

Tab. 6: Test loss of the soft prompts, when the unimportant tokens are masked or removed. Masking inflicts a performance drop, which is quite high depending on the dataset. Prompt compression performs similarly to prompt masking.

Prompt masking and compression Token masking reveals that there are only a few important tokens per model, allowing us to mask or remove unimportant tokens. As shown in Table 6, masking or compressing unimportant tokens increases model loss, but masked soft prompts perform better than untrained soft prompts (loss of 9 to 11).

For comparison, we train a soft prompt with a length of 2 on each MultiBERT model with the IMDb dataset. These can be compared to the IMDb 10_model soft prompts from Table 6, as they have roughly 2 unmasked tokens (with a prompt length of 16, training on 10 models, each model gets 1.6 tokens). The shorter single model prompts achieve an average test loss of **0.43 ± 0.085** , significantly better than the 1.972 ± 0.399 of the masked 10_model prompts.

Despite this, the masking heuristic outperforms random masking. Randomly masking all but 2 tokens for the IMDb 10_model prompts results in a test loss of 9.598 ± 2.865 across 3 seeds, much worse than our heuristic.

However, variability exists, with some soft prompts showing model-specific performance differences due to unequal token allocation. For example, one model may receive favorable token assignments, leading to good performance, while another loses an important token. Prompt masking and compression are feasible but require careful token selection.

Effects on transferability The transferability of soft prompts is hindered by the model-specific nature of the token splitting effect. Soft prompts trained on multiple models do not display generalizability but rather specialize within the embedding spaces of the trained-on models. This specialization hinders their application to models beyond the original training distribution, which diverges from the generalization capabilities of multi-model training shown in previous research like Zou et al. [Zo23].

4.3 Factors influencing the token splitting effect

Dimension	IMDb	emotion	mnli
Models	2.065 \pm 0.684	1.384 \pm 0.05	1.251 \pm 0.032
Train seed	1.605 \pm 0.192	1.438 \pm 0.071	1.289 \pm 0.011
Init seed	1.645 \pm 0.254	1.574 \pm 0.306	1.274 \pm 0.02

Tab. 7: The test loss the important tokens achieve when masked, categorized by soft prompt category and dataset. Choosing different models, initializations, or training seeds has no impact on the strength of the token-splitting effect.

Influence of training and initialization seed The training and initialization seed does not significantly impact the token importance or the alignment within models (see Table 7). The seeds do have a substantial impact on the token assignment. When comparing token masking assignments from the same models with different training and initialization seeds (see Table 3; with different training seed Table 8; and with different initialization seed Table 9) the tokens are assigned to different models and the soft prompts perform differently the models. There seems to be no predictability of which tokens get chosen from which model.

Model	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	None
model 0	0.306	0.311	0.302	0.296	0.294	0.295	0.292	0.301	0.296	0.341	0.299	0.309	0.300	0.319	0.483	0.311	0.298
model 1	0.321	0.314	0.309	1.819	0.308	0.309	2.290	1.931	0.308	0.308	0.309	0.309	0.309	0.308	0.308	0.307	0.307
model 2	3.216	0.325	0.316	0.316	0.319	4.638	0.318	0.318	0.317	0.317	0.339	0.319	0.320	0.318	0.319	0.321	0.315
model 3	0.287	3.296	0.283	0.324	0.284	0.284	0.284	0.284	0.309	0.284	0.972	0.646	0.284	0.284	0.285	0.284	0.284
model 4	0.308	0.301	0.434	0.300	1.690	0.300	0.299	0.299	0.379	0.299	0.378	0.300	5.632	0.300	0.299	0.300	0.3

Tab. 8: The test loss for the soft prompt 5_multibert_train_1 (trained on IMDb), when each token is masked separately and evaluated on each model.

Model	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	None
model 0	1.173	0.370	0.347	0.347	0.340	0.344	0.374	0.393	0.344	0.348	0.355	0.352	0.348	0.338	0.354	0.343	0.35
model 1	0.317	3.172	0.715	0.311	0.311	0.310	0.310	0.313	0.311	0.312	0.715	0.311	0.311	0.312	0.310	1.267	0.31
model 2	0.305	0.300	0.363	0.299	1.594	0.308	0.300	0.300	1.977	0.300	0.302	0.303	0.303	0.390	0.357	0.300	0.299
model 3	0.297	0.296	0.295	0.298	0.526	0.419	0.300	0.299	0.298	0.299	0.299	0.298	3.467	0.301	0.300	0.298	0.297
model 4	0.303	0.295	0.296	2.026	0.296	0.295	0.296	0.296	0.296	0.327	0.332	1.881	0.296	0.296	0.340	0.295	0.297

Tab. 9: The test loss for the soft prompt 5_multibert_init_1 (trained on IMDb), when each token is masked separately and evaluated on each model.

Influence of training duration The soft prompts perform worst on the mnli dataset. We evaluate if this is due to under-training by training these soft prompts for 2, 5, 10, and 20 epochs (see Table 10). Extended training periods result in a notable reduction in loss across models and an increase in the significance of important tokens. It should be noted, that the importance of the soft prompts trained on 2 epochs is relatively high due to a higher baseline loss, which is a limitation of our current importance metric as discussed in section 5. We find that the token-splitting effect is stronger with increased training.

#Epochs	2	5	10	20
Normal loss	1.153 ± 0.015	1.108 ± 0.009	1.078 ± 0.004	1.052 ± 0.005
Important token loss	1.298 ± 0.039	1.251 ± 0.032	1.259 ± 0.011	1.363 ± 0.048

Tab. 10: The test loss the important tokens achieve, when masked, grouped after a number of epochs. These are only the soft prompts trained on mnli. The more epochs a model trains the better its performance and the higher the important token loss.

5 Limitations

Our work is limited by the exclusive use of the MultiBERT models, which means our findings on the token splitting effect and soft prompts’ performance may not extend to models with different architectures or scales. Our results are also constrained by focusing on three datasets, so our insights may not generalize across different data domains or task types. Future research should validate these findings on a broader range of datasets and tasks.

We did not investigate the balance between prompt length and available input space. While our experiments used a fixed prompt length, task-specific characteristics might need adjustments of this parameter, which should be explored further.

The importance metric defined in section 3 is simple but has flaws. For example, Table 9 shows model 0 achieving a better loss when token 9 is masked, yet the token is still assigned to the model. The metric works best if the soft prompt has a similar loss across all models. A potential improvement would be to include the soft prompt’s performance on the model

without masking: $\text{importance}(t_i, m) = \frac{\text{Loss}_m([t_{<i}, z, t_{>i}]; \text{input})}{\text{Loss}_m(p; \text{input})}$ where p is the prompt, t_i the prompt token at index i , m the model, and z a zero vector of the soft prompt dimension.

6 Conclusion

In our research, we extend the discourse on the characteristics of soft prompts within the context of pre-trained language models. Based on a series of experiments we analyze the dynamics of soft prompts and their relation with the embedding spaces of pre-trained language models.

Our main contribution is identifying the Token Splitting Effect, where specific tokens are critically important across different models and lie within their respective embedding spaces. Models distribute these tokens roughly equally, and masking important tokens significantly decreases performance. This effect intensifies with more models involved in training while maintaining a constant prompt length, especially among high-performing models. Changes in training and initialization seeds result in a similar effect strength but different token distributions. Prompt masking leads to performance drops and performs worse than training smaller prompts but better than random masking. Prompt compression is similar to token masking but results in shorter prompts.

Future research should broaden the evaluation of the Token Splitting Effect to include a wider range of model architectures to determine its consistency and limitations. Additionally, exploring the impact of changing prompt lengths based on the number of models could optimize soft prompts more effectively. Investigating the use of soft prompts in creating adversarial scenarios is another interesting area, building on studies like Khashabi et al. [Kh22] and Bailey et al. [Ba23]. It would be valuable to examine if the Token Splitting Effect can generate tokens that enhance model performance in specific contexts while presenting challenges or adversarial behavior in others.

7 Computational budget

We use the 25 MultiBERT models [Se22] which are trained based on the architecture of BERT Base and thus have 110 million parameters [De19]. The soft prompts have a prompt length of 16 with an embedding dimension of 768, resulting in 12,288 tunable parameters.

We logged about 5 days of training runs with wandb on an NVIDIA GeForce RTX 3090 with one GPU and 24 CPUs. We trained for 5 hours on an NVIDIA RTX A6000 with one GPU and 24 CPUs. An evaluation requires about 20 seconds of testing (or validation since the dataset sizes are the same) on the RTX 3090. For the token splitting effect, we need one test epoch per model per prompt token. This results in 160 test epochs for every 10 model configuration and a run time of about one hour. We estimate to have spent around 3 days of compute for the evaluation. This results in 8 compute days on the RTX 3090 and 5 hours on the NVIDIA RTX A6000.

References

- [Ba23] Bailey, L.; Ahdriz, G.; Kleiman, A.; Swaroop, S.; Doshi-Velez, F.; Pan, W.: Soft prompting might be a bug, not a feature. In: Workshop on Challenges in Deployable Generative AI at International Conference on Machine Learning (ICML). Honolulu, Hawaii, 2023.
- [De19] Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In (Burstin, J.; Doran, C.; Solorio, T., eds.): Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, NAACL-HLT 2019, minneapolis, MN, USA, june 2-7, 2019, volume 1 (long and short papers). Association for Computational Linguistics, pp. 4171–4186, 2019, DOI: 10.18653/v1/N19-1423, URL: <https://doi.org/10.18653/v1/n19-1423>.
- [DSZ23] Dobler, K.; Schall, M.; Zimmermann, O.: nlp-research-template, 2023, URL: <https://github.com/konstantinjdobler/nlp-research-template>, visited on: 10/19/2023.
- [FH89] Fix, E.; Hodges, J. L.: Discriminatory analysis. Nonparametric discrimination: Consistency properties. International Statistical Review/Revue Internationale de Statistique 57 (3), Publisher: JSTOR, pp. 238–247, 1989.
- [Kh22] Khashabi, D.; Lyu, X.; Min, S.; Qin, L.; Richardson, K.; Welleck, S.; Hajishirzi, H.; Khot, T.; Sabharwal, A.; Singh, S.; Choi, Y.: Prompt Waywardness: The Curious Case of Discretized Interpretation of Continuous Prompts. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, Seattle, United States, pp. 3631–3643, 2022, DOI: 10.18653/v1/2022.naacl-main.266, URL: <https://aclanthology.org/2022.naacl-main.266>.
- [LAC21] Lester, B.; Al-Rfou, R.; Constant, N.: The Power of Scale for Parameter-Efficient Prompt Tuning. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, pp. 3045–3059, 2021, DOI: 10.18653/v1/2021.emnlp-main.243, URL: <https://aclanthology.org/2021.emnlp-main.243>.
- [Lh21] 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 empirical methods in natural language processing: System demonstrations. arXiv: 2109.02846 [cs.CL], Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, pp. 175–184, 2021, URL: <https://aclanthology.org/2021.emnlp-demo.21>.
- [Li21] Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; Neubig, G.: Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, arXiv:2107.13586 [cs], 2021, URL: <http://arxiv.org/abs/2107.13586>, visited on: 11/22/2023.
- [LL21] Li, X. L.; Liang, P.: Prefix-Tuning: Optimizing Continuous Prompts for Generation. 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). Association for Computational Linguistics, Online, pp. 4582–4597, 2021, DOI: 10.18653/v1/2021.acl-long.353, URL: <https://aclanthology.org/2021.acl-long.353>.

- [Ma11] Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies. Dataset retrieved from Hugging Face: <https://huggingface.co/datasets/imdb>, Association for Computational Linguistics, Portland, Oregon, USA, pp. 142–150, 2011, doi: 10.5555/2002472.2002491, URL: <http://www.aclweb.org/anthology/P11-1015>.
- [MHM18] McInnes, L.; Healy, J.; Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2018.
- [Pe01] Pearson, K.: LIII. *On lines and planes of closest fit to systems of points in space*. en, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2 (11), pp. 559–572, 1901, ISSN: 1941-5982, 1941-5990, doi: 10.1080/14786440109462720, URL: <https://www.tandfonline.com/doi/full/10.1080/14786440109462720>, visited on: 02/27/2024.
- [Ra18] Radford, A.; Narasimhan, K.; Salimans, T.; Sutskeve, I.: Improving Language Understanding by Generative Pre-Training, 2018.
- [Ra20] Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P. J.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research 21 (140), pp. 1–67, 2020, URL: <http://jmlr.org/papers/v21/20-074.html>.
- [Sa18] Saravia, E.; Liu, H.-C. T.; Huang, Y.-H.; Wu, J.; Chen, Y.-S.: CARER: Contextualized affect representations for emotion recognition. In: Proceedings of the 2018 conference on empirical methods in natural language processing. Dataset retrieved from Hugging Face: <https://huggingface.co/datasets/dair-ai/emotion>, Association for Computational Linguistics, Brussels, Belgium, pp. 3687–3697, 2018, doi: 10.18653/v1/D18-1404, URL: <https://www.aclweb.org/anthology/D18-1404>.
- [Se22] Sellam, T.; Yadlowsky, S.; Tenney, I.; Wei, J.; Saphra, N.; D’Amour, A. N.; Linzen, T.; Bastings, J.; Turc, I. R.; Eisenstein, J.; Das, D.; Pavlick, E.: The MultiBERTs: BERT Reproductions for Robustness Analysis, Publication Title: ICLR 2022 Models retrieved from Hugging Face: For example for seed 0: https://huggingface.co/google/multiberts-seed_0, 2022, URL: <https://arxiv.org/abs/2106.16163>.
- [VH08] Van der Maaten, L.; Hinton, G.: Visualizing data using t-SNE. Journal of machine learning research 9 (11), 2008.
- [WNB18] Williams, A.; Nangia, N.; Bowman, S.: A broad-coverage challenge corpus for sentence understanding through inference. In (Walker, M.; Ji, H.; Stent, A., eds.): Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long papers). Dataset retrieved from Hugging Face: <https://huggingface.co/datasets/glue/viewer/mnli>, Association for Computational Linguistics, New Orleans, Louisiana, pp. 1112–1122, 2018, doi: 10.18653/v1/N18-1101, URL: <https://aclanthology.org/N18-1101>.
- [Zo23] Zou, A.; Wang, Z.; Kolter, J. Z.; Fredrikson, M.: Universal and Transferable Adversarial Attacks on Aligned Language Models, Publisher: arXiv Version Number: 1, 2023, doi: 10.48550/ARXIV.2307.15043, URL: <https://arxiv.org/abs/2307.15043>, visited on: 10/24/2023.