What the Weight?!
A Unified Framework for Zero-Shot Knowledge Composition

Anonymous ACL submission

Abstract

The knowledge encapsulated in a model is the core factor determining its final performance on downstream tasks. Much research in NLP has focused on efficient methods for storing and adapting different types of knowledge, e.g., in dedicated modularized structures, and on how to effectively combine these modules, e.g., via parameter averaging at test time. However, given the many possible options in composing knowledge, a thorough understanding of the mechanisms involved is missing, and hence it remains unclear which strategies to utilize. In this work, we address this research gap by proposing a novel framework for zero-shot module composition, which encompasses existing and some novel variations for selecting, weighting, and combining parameter modules under a single unified notion. Focusing on the scenario of domain knowledge and adapter layers, our framework provides a systematic unification of concepts, allowing us to conduct the first comprehensive benchmarking study on various zero-shot knowledge composition strategies. In particular, we test two module combination methods (parameter averaging, output ensembling), and five selection and weighting strategies (uniform, and based on entropy, domain prior, TF–IDF, and semantic similarity) for their effectiveness and efficiency on 21 training and 10 evaluation domains across three models. Our results highlight the efficacy of ensembling, but also hint at the power of simple though often-ignored weighting methods. We further conduct various in-depth analyses, that, for instance, allow us to understand the role of weighting vs. top-k selection, and we show that, to a certain extent, the performance of adapter composition can even be predicted.

1 Introduction

Pre-trained language models (PLMs), e.g., the GPT-family (Radford et al., 2019; Brown et al., 2020, *inter alia*), determine the current state-of-the-art

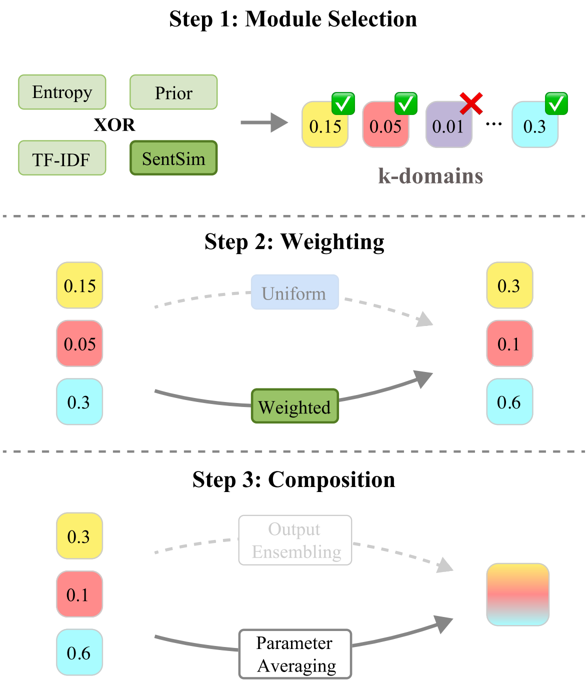


Figure 1: Our unified framework for on-demand module composition consisting of three steps: selection, weighting, and final combination. We show the example of zero-shot domain adaptation with adapter layers.

in Natural Language Processing (NLP), which has often been attributed to the rich knowledge they encapsulate in their parameters (e.g., Tenney et al., 2019). Previous research has heavily focused on utilizing the PLMs' knowledge in various scenarios particularly in a zero-shot setting, e.g., to transfer the knowledge of different source domains to a specific target domain (e.g., Emelin et al., 2022; Hung et al., 2022, *inter alia*).

Besides the numerous practical advantages of knowledge modularization – such as parameter-efficiency (Ponti et al., 2023), avoiding catastrophic forgetting (Ansell et al., 2021), and reducing negative interference (Sun et al., 2020) – researchers have shown the benefits of re-using and re-combining already existing modules (Pfeiffer et al., 2021).

Based on this idea, a particularly attractive scenario is the *on-demand selection and combination of knowledge modules at inference time*. To do so, there exist a plethora of potential strategies: modules can be selected by computing sentence similarities and domain clusters (Chronopoulou et al., 2023), domain priors (Li et al., 2022), and model entropy (Wang et al., 2022). Then, they can be combined with a weight space averaging, following the idea of a "model soup" (Wortsman et al., 2022), or output vector ensembling (Li et al., 2022).

However, despite the existence of a variety of knowledge composition methods, there is (a) no comprehensive overview and evaluation of those methods, and (b) no unified view on knowledge composition that could facilitate this process. The composition methods introduced for various objectives have not been tested in a comparable setup (e.g., Li et al. (2022), do not focus on zero-shot domain adaptation, in contrast to Chronopoulou et al. (2023)), and various factors (e.g., the number of modules to select, and whether to additionally weight each module in the composition) have not been systematically taken into account. We shed light on these, focusing on the specific case of zero-shot domain adaptation with adapter layers. Given a series of adapters originating from domain-specific training, we address the problem of how to choose and combine adapters to improve the performance on unseen evaluation domains.

Contributions. Our contributions are three-fold: **(1)** we present a unified framework for zero-shot knowledge composition (see Figure 1), which provides an interoperable notion on knowledge composition variations proposed for diverse scenarios in the literature. Our framework allows us **(2)** to conduct a large evaluation of knowledge composition strategies for zero-shot domain adaptation to date. Concretely, we test two combination methods (averaging and ensembling), and five selection and weighting strategies (uniform, and based on model entropy, domain prior, semantic sentence similarity, and TF–IDF (which has been previously ignored) across three models (`gpt2-base`, `gpt2-large`, `deberta-base`) using 21 training and 10 evaluation domains. **(3)** We advance our understanding of knowledge composition by proposing and studying a meta-regression method applied to the framework, aiming to predict the optimal combinatorial setting.

Our experiments show that w.r.t. combination strategies, output vector ensembling is often superior to parameter averaging, supporting findings from recent work (Li et al., 2022). Importantly, we observe that corpus-based weighting and selection strategies (TF–IDF and SENTENCE SIMILARITY) often outperform more complex model-based approaches, while also being more efficient. Our study on meta-regression shows that zero-shot domain adaptation performance is partially predictable, particularly for specific adapter combinations. We hope that our work will advance efficient and effective NLP. For full reproducibility, we release all code publicly under `[URL]`.

2 A Unified Composition Framework

In this section, we present our unified framework for knowledge module composition. We base our explanation on the scenario of domain adaptation using adapters as the underlying module. Our framework is, however, generic and can be applied to various composition scenarios.

The problem of composing knowledge boils down to the following: let θ_i be the parameters of n adapters trained via language modeling on n domains $D_1, ..., D_n$ while the original model parameters ϕ are kept frozen. Given an unseen evaluation domain D_{n+1}, the task is to effectively adapt to D_{n+1} via an optimal domain composition. As illustrated in Figure 1, our approach to such a composition relies on three steps: (**1**) identify k suitable adapters; (**2**) apply a weighting to the selected adapters; (**3**) perform the final combination. In the following, we describe the scoring and the combination strategies, implemented in our framework and used for conducting the experiments.

2.1 Scoring Strategy

We examine five scoring strategies. These strategies are utilized for selecting the top-k most suitable adapters (**1**), and/or to compute the weights ω_i per domain (**2**) which will later be used in the combination. Concretely, our framework consists of uniform, two corpus-based, and two model-based scoring approaches, explained in the following.

Uniform. In this simplest method (UNIFORM), the scores follow a uniform distribution with values of $\omega_i = 1/k$. This strategy can not be used for selecting the top-k, but it can be paired with other strategies that provide the top-k best domain adapters, by further weighting these uniformly.

Semantic Sentence Similarity. This is a corpus-based scoring strategy (SENTSIM). In line with Chronopoulou et al. (2023), we compute Sentence-BERT (Reimers and Gurevych, 2019) embeddings for 100 randomly selected sequences of the development set of each of the training domains $D_1, ..., D_n$, and of the unseen evaluation domain D_{n+1}. Next, we compute the averaged cosine similarity for each $D_1, ..., D_n$ across the 100 training embeddings with each of the 100 embeddings from D_{n+1}. We obtain the final SENTSIM scores through normalization, dividing each cosine similarity by the sum of all similarities. The resulting scores are in $[0, 1]$, such that $\sum_{i=1}^{k} \omega_i = 1$.

TF–IDF. In contrast to previous work, we also examine Term Frequency–Inverse Document Frequency (TF–IDF), as another simple corpus-based scoring strategy. Here, we are motivated by the fact that domain differences also manifest in different lexical choices. As before, we extract 100 sequences of the development sets of each of the training domains and of the novel evaluation domain. We then compute TF–IDF vectors for each subset and compute the scores as the normalized average cosine similarity (see above). We provide the exact TF–IDF formulation in the Appendix B.

Domain Prior. Following Gururangan et al. (2022) and Li et al. (2022), here, we consider score estimation as a Bayesian problem (PRIOR): we introduce a domain variable D alongside each sequence x of the evaluation set and define $p(x|D = j)$ as the conditional probability of the last token in the sequence, given the preceding tokens, calculated by applying a softmax over the model output vector. Applying Bayes' rule, we estimate the domain posterior $p(D = j|x)$ (the probability of a sequence belonging to the domain j) as follows:

$$p(D = j|x) = \frac{p(x|D = j) \cdot p(D = j)}{p(x)} = \frac{p(x|D = j) \cdot p(D = j)}{\sum_{j'=1}^{k} p(x|D = j') \cdot p(D = j')} . \quad (1)$$

To estimate the domain prior $P(D = j)$, we compute the exponential moving average (EMA) of the posterior probabilities at the end of each sequence block. We use $N = 100$ sequences of the dev sets with a sequence length of 1024 and an EMA decay of $\lambda = 0.3$, which has been found to result in stable posterior probabilities (Li et al., 2022).

$$p(D = j) = \sum_{i=1}^{N} \lambda^i \cdot p(D = j|x^{(i)}) , \quad (2)$$

with individual input sequences x_i. We then fix the obtained domain priors and use those as scores at inference time. We apply averaging normalization, causing the scores of k adapters to sum up to 1.

Entropy. This method leverages model uncertainty as a scoring strategy (ENTROPY). Our method has conceptual similarities to the one of Wang et al. (2021b), while in contrast instead of running multiple gradient descent iterations, we opt for a more efficient strategy and measure the uncertainty for each adapter on the development sets X with a single pass. Similar to Lesota et al. (2021), we define model uncertainty as the entropy of the predicted probability distribution:

$$H(X) = -\sum_{x \in X} p(x) \cdot \log p(x) , \quad (3)$$

with mini-batches x, and $p(x)$ being the mean probability of the next token given the preceding tokens for all sequences in the batch. For each adapter, we then compute the uncertainty of the model on the evaluation set (that is, the data corresponding to the unseen domain). The resulting uncertainties are then normalized to obtain certainty scores with values in the range of $[0, 1]$. This way, the domain adapter achieving the lowest uncertainty on the evaluation set gets the highest weight assigned.

2.2 Combination Method

Given the weight vector ω we obtained from steps (**1**) and (**2**), we rely on two combination methods to combine the knowledge modules (**3**).

Parameter Averaging. We follow Chronopoulou et al. (2023) and use "model souping" (Wortsman et al., 2022), namely weight space averaging, as our first combination strategy. To ensure consistency, we also treat the parameters of the PLM heads of auto-encoding models as parts of θ_i – the parameters specific to a particular domain D_i, as these appear to have a major impact on the downstream task. Here, we thus average over both the adapter layers and the weight space of the head's parameters. Expanding on the original proposal by Chronopoulou et al. (2023), we also allow for the weighting of the adapters. In particular, we consider $f(x, \phi, \theta_i)$ as a single model with its original parameters ϕ, and the domain-specific adapter and head parameters θ_i operating on the provided textual input x. The new model using the parameter averaging method is hence formulated as:

$$f(x, \phi, \sum_{i=1}^{k} \omega_i * \theta_i)\,, \tag{4}$$

with ω_i as the weight for the domain-specific parameters θ_i, and k the number of selected adapters.

Ensembling. In this method, we ensemble the outputs of k selected models $f(x, \phi, \theta_i)$, each defined with the corresponding domain-specific parameters. This strategy is similar to the one proposed in Li et al. (2022).

$$\sum_{i=1}^{k} \omega_i * f(x, \phi, \theta_i)\,. \tag{5}$$

Compared to averaging, this strategy requires a separate pass through each model of the ensemble.

3 Benchmarking Composition Strategies

We use our framework to benchmark module composition strategies for zero-shot domain adaptation.

3.1 Overall Experimental Setup

Data. We follow Chronopoulou et al. (2023) and resort to defining domains by provenance, i.e., the source of a document. Although the notion of a domain is fuzzy (Plank, 2016; Saunders, 2021), the document sources provide an intuitive segmentation of the corpora while also being common practice in NLP research. We use the same 21 training domains, which correspond to collections of text from 21 websites, and 10 evaluation domains as in (Chronopoulou et al., 2023). 30 of these constitute domains from the 100 most high-resource internet domains from the C4 dataset (Raffel et al., 2020; Dodge et al., 2021). We also add the publicly available `yelp.com` dataset.[1] We show all datasets along with their train-eval split sizes in Table 1.

Models. We evaluate one auto-encoding and two auto-regressive models. To be able to compare our results to Chronopoulou et al. (2023), we use GPT-2 (Radford et al., 2019) in the *base* configuration (`gpt2-base`). Additionally, we evaluate the *large* configuration (`gpt2-large`) and further train domain adapters for the DeBERTa model (He et al., 2021) in the *base* configuration (`deberta-base`). We obtain all models from the Huggingface Transformers library (Wolf et al., 2020).

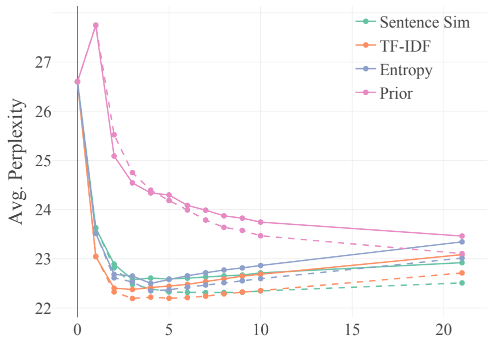
[1] https://www.yelp.com/dataset

Split	Datasets	# Tokens
Train	dailymail.co.uk	23M (3M)
	wired.com	18M(2M)
	express.co.uk	13M (2M)
	npr.org	24M (3M)
	librarything.com	2M (300K)
	instructables.com	24M (3M)
	entrepreneur.com	15M (2M)
	link.springer.com	23M (3M)
	insiderpages.com	6M (700K)
	ign.com	9M (1M)
	eventbrite.com	6M (800K)
	forums.macrumors.com	19M (2M)
	androidheadlines.com	14M (2M)
	glassdoor.com	2M (200K)
	pcworld.com	13M (2M)
	csmonitor.com	22M (3M)
	lonelyplanet.com	4M (500K)
	booking.com	30M (4M)
	journals.plos.org	6M (1M)
	frontiersin.org	31M (4M)
	medium	21M (3M)
Eval	reuters.com	16M (2M)
	techcrunch.com	12M (2M)
	fastcompany.com	13M (2M)
	nme.com	3M (300K)
	fool.com	34M (4M)
	inquisitr.com	13M (2M)
	mashable.com	12M (2M)
	tripadvisor.com	5M (1M)
	ncbi.nlm.nih.gov	21M (3M)
	yelp.com	15M (2M)

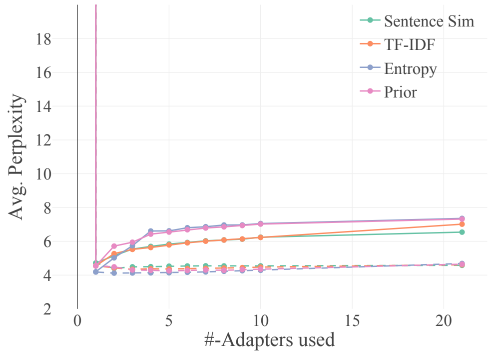
Table 1: Datasets used in our study. We show the 21 training and 10 evaluation domains with their sizes measured in number of tokens (training (eval)).

Adapter Training and Optimization. We train each domain adapter separately via language modeling (masked language modeling or causal language modeling, depending on the model) on a single NVIDIA A6000 GPU with 48 GB RAM. For each adapter, we use a random seed of 5 during training. We train for 20 epochs using the Adam optimizer (Kingma and Ba, 2015) (weight decay = 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \cdot 10^{-6}$, learning rate=$1 \cdot 10^{-4}$). For `deberta-base` and `gpt2-base`, we use an effective batch size of 80, while for the bigger model, `gpt2-large`, we set the effective batch size to 20. To make the results of `gpt2-base` comparable to the results of Chronopoulou et al. (2023), we adopt the adapter architecture proposed by Bapna and Firat (2019), that is, we insert an adapter layer after the transformer feed-forward layer. We set the reduction factor to 12, resulting in a bottleneck size of 64 for `gpt2-base` and `deberta-base`, and 107 for `gpt2-large`.

Evaluation. For each evaluation domain, we measure the models' perplexities obtained after

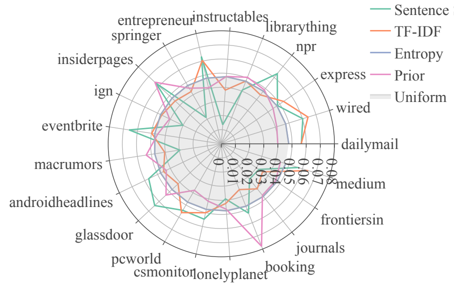


(a) `gpt2-base`

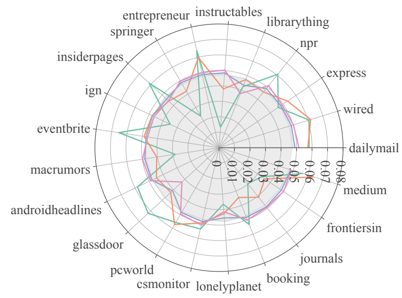


(b) `deberta-base`

Figure 2: Comparison between Parameter Averaging (solid lines) and Ensembling (dashed lines) over different numbers of top-k adapters. We show the mean perplexity results for (a) `gpt2-base`, and (b) `deberta-base` for each of our scoring strategies (SENTSIM, TF–IDF, ENTROPY, PRIOR) averaged across four runs.



(a) `gpt2-base`



(b) `deberta-base`

Figure 3: Adapter weights for all training domains and scoring strategies when using all trained adapters. The light grey shade indicates the uniform weighting.

adapter composition. All evaluations are conducted over 4 different random seeds $(5, 10, 42, 88)$ and averaged to achieve stable results.

3.2 Results

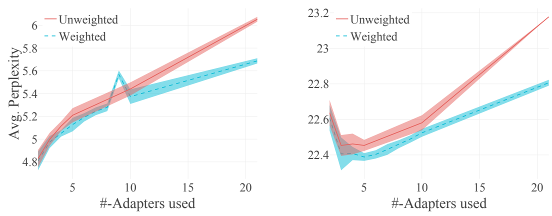
Combination Strategies. We compare the two combination strategies, parameter averaging, and ensembling, coupled with all four scoring strategies, applied for adapter selection and adapter weighting. The perplexities for `gpt2-base` and `deberta-base` are depicted in Figure 2. We show results for `gpt2-large` in the Appendix C. Note that for $k = 0$ and $k = 1$ (no adapter or a single adapter), the combination strategies are equivalent, as we do not need to merge any adapters. Interestingly, `deberta-base` hugely profits from adding a single adapter (improvement of up to -183662.70 in perplexity). Adding a second adapter does, on average, when averaging modules, no longer lead to an improvement. This warrants further investigation on when exactly the knowledge contained in an adapter helps (cf. §4). From $k = 2$ on, ensembling leads to better domain adaptation across most model types and scoring strategies, indicated by lower model perplexities. These findings hold when choosing two adapters only ($k = 2$) and also when increasing k, up to $k = 21$ (all adapters chosen) and are significant at $\alpha = 0.05$ using the Wilcoxon Signed Rank test. With larger k the difference between the combination strategies even increases (from -0.08 for $k = 2$ to -0.41 for $k = 21$ and TF–IDF). The only exception is `prior` for `gpt2-base`, where averaging reaches better performance for smaller k. Overall, we can confirm the recent findings of Li et al. (2022): ensembling typically leads to better performance than module averaging. However, we also conclude that adding more adapters can also harm the performance.

Scoring Strategies. We evaluate the effectiveness of the scoring strategies for weighting all 21 training adapters (see Table 2). Surprisingly, we observe that simpler (and previously ignored) approaches to determine the weighting, e.g., SENTSIM and TF–IDF, often lead to better results compared to more sophisticated approaches. However, for smaller numbers of adapters, the picture can vary (see again Figure 2). To shed more light on this phenomenon, we show the weights obtained through the different scoring strategies in Figure 3:

	Method	Results on the 10 Evaluation Domains (AVG/ENS) reuters	techcru	fastco	nme	fool	inquisitr	mashable	tripadv	ncbi	yelp
♠		21.5	27.7	27.9	28.2	23.8	22.4	27.1	40.4	20.7	36.2
♠	SENTSIM	17.6	22.0	21.3	20.7	22.2	18.4	22.4	36.2	17.6	35.2
gpt2-base		20.2	27.4	27.1	28.4	22.9	21.9	25.7	38.4	19.7	34.4
gpt2-base	UNIFORM	16.9/16.4	23.2/22.6	22.8/21.9	22.8/21.9	21.3/21.3	18.3/17.3	22.2/21.9	34.6/33.8	18.2/18.0	33.3/34.4
gpt2-base	SENTSIM	**16.5/16.1**	**22.8/22.3**	**22.5**/21.7	22.3/**21.5**	**21.2/21.2**	**18.0/17.6**	**21.9/21.6**	**33.7/32.4**	**17.4/17.2**	**32.9/33.7**
gpt2-base	TF–IDF	16.5/16.1	22.8/22.3	22.5/**21.7**	**22.2/21.5**	21.3/21.2	18.0/17.6	22.1/21.7	34.4/33.4	17.8/17.5	33.2/34.1
gpt2-base	ENTROPY	16.8/16.4	23.2/22.6	22.8/21.9	22.8/21.9	21.3/21.3	18.3/17.8	22.3/21.9	34.6/33.8	18.2/18.0	33.3/34.4
gpt2-base	PRIOR	17.1/16.6	23.4/22.8	23.1/22.2	23.1/22.3	21.4/21.4	18.4/18.0	22.4/22.1	34.4/33.6	18.2/18.1	33.2/34.2
gpt2-large		12.2	17.5	17.1	16.6	15.4	14.0	16.7	26.4	12.6	**23.0**
gpt2-large	UNIFORM	11.2/10.6	16.0/15.3	15.5/14.8	14.6/13.7	14.9/14.4	12.7/12.1	15.3/14.6	24.2/23.2	11.9/11.7	24.0/23.5
gpt2-large	SENTSIM	11.1/**10.5**	**15.7/15.0**	15.4/**14.7**	**14.3/13.5**	**14.9/14.4**	**12.5/12.0**	**15.1/14.4**	**23.3/22.2**	**11.4/11.1**	23.3/23.6
gpt2-large	TF–IDF	**11.1/10.5**	15.8/15.1	**15.4**/14.7	14.3/13.5	14.9/14.4	12.5/12.0	15.2/14.5	24.0/22.9	11.7/11.3	23.8/23.9
gpt2-large	ENTROPY	11.2/10.8	16.0/15.5	15.5/15.0	14.6/14.0	14.9/14.6	12.7/12.3	15.3/14.6	24.2/23.2	11.9/11.7	24.0/24.2
gpt2-large	PRIOR	11.2/10.7	16.1/15.4	15.6/14.9	14.7/13.9	14.9/14.5	12.7/12.2	15.3/14.7	24.1/23.0	11.9/11.7	23.9/24.1
deberta-base		116975.5	123763.4	122145.2	117231.9	125070.4	118561.9	118559.0	123046.6	110694.9	125107.5
deberta-base	UNIFORM	6.7/4.1	7.1/4.5	6.4/**4.1**	7.1/4.6	7.1/**4.4**	5.8/3.7	6.8/4.2	9.8/**6.3**	8.8/5.8	8.4/5.5
deberta-base	SENTSIM	**5.9/3.9**	**6.3/4.4**	**5.9/4.1**	**6.2/4.5**	**6.4/4.4**	**5.1/3.5**	**6.1**/4.2	**8.7/6.3**	**7.0/4.6**	**7.9**/5.8
deberta-base	TF–IDF	6.2/4.0	6.6/**4.4**	6.1/**4.1**	6.6/**4.5**	6.8/**4.4**	5.4/3.6	6.5/4.2	9.4/**6.3**	8.4/5.2	8.2/5.5
deberta-base	ENTROPY	6.6/4.0	7.1/**4.4**	6.4/**4.1**	7.0/4.6	7.0/**4.4**	5.7/3.6	6.8/4.2	9.8/**6.3**	8.7/6.3	8.4/5.5
deberta-base	PRIOR	6.6/4.0	6.9/**4.4**	6.4/**4.1**	7.0/**4.5**	7.0/**4.4**	5.6/3.6	6.7/**4.2**	9.8/**6.3**	8.7/5.6	8.4/**5.4**

Table 2: Perplexity results using all trained adapters for prediction and comparison with recent publications as well as different scoring strategies averaged over 4 different initializations. The perplexities marked with ♠ represent the results of Chronopoulou et al. (2023) obtained with `gpt2-base`.

the model-based scoring strategies produce weight distributions closer to the uniform distribution than the two corpus-based ones, where domain differences are more pronounced. We conclude that model-based ones are thus, while providing good results in adapter selection (i.e., when a fixed and smaller k is chosen), less suitable for fine-grained weighting of a larger set of adapters. We are also interested in whether the more advanced scoring strategies should be used as weighting mechanisms or whether uniform weighting leads to superior results. To this end, we compute the perplexities on all evaluation datasets in two variants: (i) when using the different scoring strategies (e.g., TF–IDF) for selection and weighting, and (ii) when only using them for selection and then uniformly weighting the selected adapters. As already indicated by the weight differences depicted in Figure 3, we do not expect big differences for model-based strategies (e.g., ENTROPY). However, for the corpus-based strategies, weighting has a small but visible effect (up to 0.3711 for $k = 21$). We show the average scores obtained across all evaluation datasets and across these strategies (TF–IDF and SENTSIM) in Figure 4: for higher k, weighting generally has a positive impact. It can thus be an alternative to fixing k – removing this additional hyperparameter – for the corpus-based scoring strategies. Yet,



(a) `deberta-base` (b) `gpt2-base`

Figure 4: Comparison between weighting adapters based on their similarity (blue) and assigning them uniform weights (red). We show the mean perplexity results for (a) `deberta-base`, and (b) `gpt2-base` and when using corpus-based scoring strategies (TF–IDF, SENTSIM) averaged over four runs and both combination strategies.

selecting a good number of adapters still stands out as a more crucial factor for optimal performance.

Efficiency. A particular motivation for modularization is the re-usability of the individual modules – leading to a reduction of the environmental impact (Strubell et al., 2020; Hershcovich et al., 2022). Here, we discuss the efficiency of the combination strategies we test within our framework. As pointed out by Li et al. (2022), ensembling is intrinsically more expensive at inference time than averaging – the amount of parameters is linearly increasing with the number of modules added. We

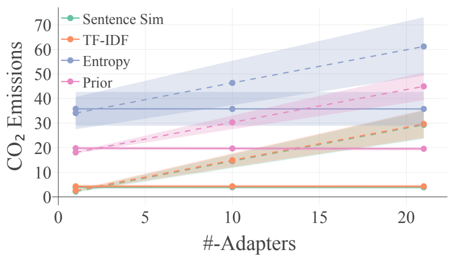


Figure 5: The different scoring and combination strategies with regards to their efficiency. We show the results for `gpt2-base` for Parameter Averaging (solid lines) and Ensembling (dashed lines) paired with each of our four scoring strategies and averaged across four runs.

now measure the expected CO_2 equivalents in our concrete experimental setup. This complements our understanding of the fine-grained differences among the individual scoring strategies. Following Hershcovich et al. (2022), we compute the CO_2 equivalents in gram (gCO_2eq) as follows:

$$\begin{aligned} gCO_2eq = \\ \text{ComputationTime (hours)}\times \\ \text{Power(kW)}\times \\ \text{EnergyMix } (gCO_2eq/kWh) \end{aligned} \tag{6}$$

We estimate these by measuring the computation time needed for each selection paired with each selection strategy. All experiments are carried out on a single NVIDIA A6000 GPU (TDP 300W) except for the score calculations with TF–IDF and SENTSIM. These were run on a single AMD EPYC 7313 CPU (TDP 155W). We employ a private server infrastructure located in [ANONYMIZED] with a carbon intensity of 470g.[2] We compute the mean carbon emission across 4 initialization seeds and display the results in Figure 5.

As expected, we measure a linear increase for ensembling, while averaging does not result in increased CO_2 equivalents. Unsurprisingly, the model-based strategies are more expensive than the corpus-based ones. Here, ENTROPY-based selection results in the highest amount of estimated carbon emissions (up to 61.17 gCO_2 vs. 3.91 for TF–IDF and ensembling).

4 Meta-Regression

In §3, we have shown that adding more adapters (i.e., increasing k) often does not lead to performance gains, and that the effectiveness of the scoring strategies varies across models and evaluation domains. Motivated by these results, here, we analyze to what extent we are able to predict the expected performance for particular compositions.

[2]Estimate from https://app.electricitymaps.com/zone/[ANONYMIZED]

4.1 Experimental Setup

Dataset and Evaluation. We run a meta-regression on our results obtained for each base model in §3. We pre-process the data as follows: to account for variations in the scores, we average over the results obtained from the four random seeds for each evaluation domain. We account for the base differences in perplexity among the evaluation domains by computing the delta between the original model performance on this dataset and the perplexity obtained by using the composition, normalized by the original perplexity. We use 10-fold cross-validation and report the results in terms of Pearson and Spearman Correlation.

Features. Each instance is represented by five feature groups: *Adapter* – the weights assigned to particular training adapters (0 if not chosen); *Number of Adapters* – the number of adapters involved in the composition; *Combination Strategy* – one-hot encoding of average or ensembling; *Scoring Strategy* – one-hot encodings of the scoring strategies (e.g., TF–IDF); and *Evaluation Dataset* – one-hot encodings of the target domain.

Models and Baselines. We experiment with Linear and Ridge regression. For Ridge, we perform hyperparameter tuning (α), leading to $\alpha = 0$ for `gpt2-base`, $\alpha = 0.17$ for `deberta-base` and $\alpha = 0.06$ for `gpt2-large`. We compare the results with a baseline predicting the mean relative difference per evaluation dataset. We hypothesize this to be a strong baseline, as the effectiveness of an adapter combination is highly dependent on the target domain.

Results. Both models surpass the baseline (see Table 3), which, as expected, already reaches high scores. The highest scores are achieved with Ridge regression on the `gpt2-base` results (0.9641 Spearman). The results on `deberta-base` are the lowest, indicating the model type to be a relevant factor. Overall, we conclude that dependent on the PLM, we are able to predict the effectiveness of domain adaptation with various compositions. We believe that this result warrants new research on selecting an optimal number and combination of modules.

Model	Regression	PearsonC	SpearmanC
gpt2-base	Mean Diff.	0.8247*	0.8152*
	Linear	0.9472*	0.9640*
	Ridge	0.9472*	0.9641*
deberta-base	Mean Diff.	0.6584*	0.6142*
	Linear	0.9127*	0.9151*
	Ridge	0.9168*	0.9225*
gpt2-large	Mean Diff.	0.8630*	0.6857*
	Linear	0.9636*	0.9526*
	Ridge	0.9683*	0.9577*

Table 3: Results of our meta-regression (mean correlation scores (Pearson and Spearman) obtained via 10-fold cross-validation, *statistically significant at $\alpha < 0.05$).

5 Related Work

For a thorough overview of modular deep learning, we refer to Pfeiffer et al. (2023).

Modularizing Knowledge. Famously, Houlsby et al. (2019) proposed to use adapter layers (Rebuffi et al., 2017) as a more efficient alternative to full task-specific fine-tuning. Subsequently, researchers in NLP explored adapters for various purposes, e.g., domain adaptation (e.g., Glavaš et al., 2021; Cooper Stickland et al., 2021; Hung et al., 2022; Malik et al., 2023), bias mitigation (e.g., Lauscher et al., 2021; Holtermann et al., 2022), language adaptation (e.g., Philip et al., 2020; Üstün et al., 2022), and storage of various other types of knowledge, such as common sense (Lauscher et al., 2020), factual (Wang et al., 2021a), and sociodemographic knowledge (Hung et al., 2023).

Similarly, much effort has been spent designing new adapter variants with the aim of further increasing their efficiency or effectiveness (e.g., Pfeiffer et al., 2021; Mahabadi et al., 2021; Zeng et al., 2023). Alternatives to adapters that support modularity include subnetworks (Guo et al., 2021) obtained via sparse fine-tuning, prefix tuning (Li and Liang, 2021), and mixture-of-expert (MoE; Jacobs et al., 1991) models.

The latter, exemplified by Switch Transformers (Fedus et al., 2022), integrate a learned gating mechanism to channel inputs to appropriate expert modules. Like other modularization techniques, MoEs have been studied extensively for a wide range of problems (e.g., Lepikhin et al., 2021; Kudugunta et al., 2021; Team et al., 2022; Ponti et al., 2023). Most relevant to us, they have also been used to modularize different types of domain knowledge (Guo et al., 2018; Zhong et al., 2023). In this context, recent studies have considered experts as entirely autonomous models, challenging prevailing efficiency paradigms (Gururangan et al., 2022; Li et al., 2022; Gururangan et al., 2023).

Composing Knowledge. The composition of knowledge modules can be conducted via optimizing additional parameters (e.g., Pfeiffer et al., 2021), or in a zero-shot manner (e.g., Chronopoulou et al., 2023). Falling under the first category of approaches, Pfeiffer et al. (2021) proposed the fusion of adapters based on weights obtained via learned attention matrices. The same mechanism has been adopted by Lu et al. (2021), dubbed knowledge controller. In a similar vein, Wang et al. (2021b) ensemble the output vectors of multiple language adapters and optimize the respective ensemble weights. Wang et al. (2022) and Muqeeth et al. (2023) compose MoE models by learning to route the input to the right modules. Most recently, Frohmann et al. (2023) propose to directly learn scaling parameters for efficient knowledge composition in task transfer.

In this work, we are interested in zero-shot knowledge composition. In this realm, Chronopoulou et al. (2023) rely on weight space averaging and simple selection strategies. Li et al. (2022) and Gururangan et al. (2023) compare ensembling and averaging for composing domain PLMs, relying on domain prior for selection. Until now, a unified view is missing.

6 Conclusion

We proposed a unified framework providing an interoperable notion of zero-shot knowledge composition. Using our framework, we analyzed the effectiveness of different module selection, weighting, and combination strategies. We studied the problem of domain adaptation with adapters and showed, for instance, that ensembling generally yields better results than parameter averaging. Examining five different scoring strategies, we found that even simple approaches can deliver strong results. Our findings also suggest that the number of adapters selected is generally more important than the weights assigned to them. Overall, our results will fuel future research in effective knowledge composition by providing a consolidated perspective on zero-shot module composition.

Limitations

Naturally, our work comes with a number of limitations. Most importantly, we conducted our experiments on the C4 dataset only. However, we strongly believe our main findings to hold also for other corpora designed for testing domain adaptation methods. Related to this aspect, our notion of domains follows the one employed in C4 and is restricted to source websites as domain representatives. Previous research has shown that this definition is not always sufficient to clearly delineate domain knowledge (e.g., Gururangan et al., 2023). Therefore, we advise practitioners to carefully choose the criteria for discriminating among domains that are most useful in their particular application scenario. Additionally, our validation relies primarily on perplexity as a measure for general NLU of PLMs. While perplexity provides a robust initial measure, it does not encapsulate all facets of language understanding and generation, and only serves as a proxy for the final downstream performance of the models. Last, we resorted to adapters as the, arguably, most popular modularization technique in our experiments. We did not test other modularization approaches (e.g., MoEs) due to the large number of additional experiments required and related environmental considerations. However, our framework is general enough to provide useful guidance for the composition of various types of modules proposed in the literature.

Ethical Considerations

We also like to point to the ethical aspects touched by our work. First, as the large body of previous work on bias measurement demonstrates, PLMs are prone to encode and propagate stereotypical and exclusive biases present in their training data (e.g., Bolukbasi et al., 2016; Blodgett et al., 2020). The models we used in our experiments are not spared from this issue (Tal et al., 2022; Narayanan Venkit et al., 2023). We advise practitioners to use these models with the appropriate care and we refer to existing works (Liang et al., 2021; Lauscher et al., 2021) for discussions on bias mitigation. Second, central to our work are environmental considerations: experimentation with deep learning models potentially entails large amounts of CO_2 emissions (Strubell et al., 2020). With our work, we hope to encourage further research on efficient NLP, in particular on modular learning and module composition, and, hence, to contribute to greener AI.

References

Alan Ansell, Edoardo Maria Ponti, Jonas Pfeiffer, Sebastian Ruder, Goran Glavaš, Ivan Vulić, and Anna Korhonen. 2021. MAD-G: Multilingual adapter generation for efficient cross-lingual transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4762–4781, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. 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 1538–1548, Hong Kong, China. Association for Computational Linguistics.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *CoRR*, abs/1607.06520.

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.

Alexandra Chronopoulou, Matthew E. Peters, Alexander Fraser, and Jesse Dodge. 2023. Adaptersoup: Weight averaging to improve generalization of pretrained language models.

Asa Cooper Stickland, Alexandre Berard, and Vassilina Nikoulina. 2021. Multilingual domain adaptation for NMT: Decoupling language and domain information with adapters. In *Proceedings of the Sixth Conference on Machine Translation*, pages 578–598, Online. Association for Computational Linguistics.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conference on Empirical Methods in Natural*

Language Processing, pages 1286–1305, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Denis Emelin, Daniele Bonadiman, Sawsan Alqahtani, Yi Zhang, and Saab Mansour. 2022. Injecting domain knowledge in language models for task-oriented dialogue systems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11962–11974, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Markus Frohmann, Carolin Holtermann, Shahed Masoudian, Anne Lauscher, and Navid Rekabsaz. 2023. Scalearn: Simple and highly parameter-efficient task transfer by learning to scale. *arXiv preprint arXiv:2310.01217*.

Goran Glavaš, Ananya Ganesh, and Swapna Somasundaran. 2021. Training and domain adaptation for supervised text segmentation. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 110–116, Online. Association for Computational Linguistics.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. 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 4884–4896, Online. Association for Computational Linguistics.

Jiang Guo, Darsh Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4694–4703, Brussels, Belgium. Association for Computational Linguistics.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. 2022. DEMix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5557–5576, Seattle, United States. Association for Computational Linguistics.

Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2023. Scaling expert language models with unsupervised domain discovery.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Daniel Hershcovich, Nicolas Webersinke, Mathias Kraus, Julia Bingler, and Markus Leippold. 2022. Towards climate awareness in NLP research. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2480–2494, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Carolin Holtermann, Anne Lauscher, and Simone Ponzetto. 2022. Fair and argumentative language modeling for computational argumentation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7841–7861, Dublin, Ireland. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp.

Chia-Chien Hung, Anne Lauscher, Dirk Hovy, Simone Paolo Ponzetto, and Goran Glavaš. 2023. Can demographic factors improve text classification? revisiting demographic adaptation in the age of transformers. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1565–1580, Dubrovnik, Croatia. Association for Computational Linguistics.

Chia-Chien Hung, Anne Lauscher, Simone Ponzetto, and Goran Glavaš. 2022. DS-TOD: Efficient domain specialization for task-oriented dialog. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 891–904, Dublin, Ireland. Association for Computational Linguistics.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. 2021. Beyond distillation: Task-level mixture-of-experts for efficient inference. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3577–3599, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Anne Lauscher, Tobias Lueken, and Goran Glavaš. 2021. Sustainable modular debiasing of language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4782–4797, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Anne Lauscher, Olga Majewska, Leonardo F. R. Ribeiro, Iryna Gurevych, Nikolai Rozanov, and Goran Glavaš.

2020. Common sense or world knowledge? investigating adapter-based knowledge injection into pretrained transformers. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 43–49, Online. Association for Computational Linguistics.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus Antonius Grasserbauer, Carsten Eickhoff, and Markus Schedl. 2021. A modern perspective on query likelihood with deep generative retrieval models. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '21, page 185–195, New York, NY, USA. Association for Computing Machinery.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models.

Xiang Lisa Li and Percy Liang. 2021. 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)*, pages 4582–4597, Online. Association for Computational Linguistics.

Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*, pages 6565–6576. PMLR.

Qiuhao Lu, Dejing Dou, and Thien Huu Nguyen. 2021. Parameter-efficient domain knowledge integration from multiple sources for biomedical pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3855–3865, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1022–1035.

Bhavitvya Malik, Abhinav Ramesh Kashyap, Min-Yen Kan, and Soujanya Poria. 2023. UDAPTER - efficient domain adaptation using adapters. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2249–2263, Dubrovnik, Croatia. Association for Computational Linguistics.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. 2023. Soft merging of experts with adaptive routing. *arXiv preprint arXiv:2306.03745*.

Pranav Narayanan Venkit, Sanjana Gautam, Ruchi Panchanadikar, Ting-Hao Huang, and Shomir Wilson. 2023. Nationality bias in text generation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–122, Dubrovnik, Croatia. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Sebastian Ruder, Ivan Vulic, and Edoardo Maria Ponti. 2023. Modular deep learning. *CoRR*, abs/2302.11529.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, Online. Association for Computational Linguistics.

Barbara Plank. 2016. What to do about non-standard (or non-canonical) language in NLP. *ArXiv preprint*, abs/1608.07836.

Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. 2023. Combining parameter-efficient modules for task-level generalisation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 687–702, Dubrovnik, Croatia. Association for Computational Linguistics.

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

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference* *on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 506–516.

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.

Danielle Saunders. 2021. Domain adaptation and multi-domain adaptation for neural machine translation: A survey. *ArXiv preprint*, abs/2104.06951.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13693–13696.

Tianxiang Sun, Yunfan Shao, Xiaonan Li, Pengfei Liu, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. Learning sparse sharing architectures for multiple tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8936–8943.

Yarden Tal, Inbal Magar, and Roy Schwartz. 2022. Fewer errors, but more stereotypes? the effect of model size on gender bias. In *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, pages 112–120, Seattle, Washington. Association for Computational Linguistics.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2022. UDapter: Typology-based language adapters for multilingual dependency parsing and sequence labeling. *Computational Linguistics*, 48(3):555–592.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021a. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1405–1418. Association for Computational Linguistics.

Xinyi Wang, Yulia Tsvetkov, Sebastian Ruder, and Graham Neubig. 2021b. Efficient test time adapter ensembling for low-resource language varieties. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 730–737, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. AdaMix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5744–5760, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time.

Guangtao Zeng, Peiyuan Zhang, and Wei Lu. 2023. One network, many masks: Towards more parameter-efficient transfer learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7564–7580, Toronto, Canada. Association for Computational Linguistics.

Tao Zhong, Zhixiang Chi, Li Gu, Yang Wang, Yuanhao Yu, and Jin Tang. 2023. Meta-dmoe: Adapting to domain shift by meta-distillation from mixture-of-experts.

Appendix

A Link to Data, Models, Code Bases

In Table 4, we provide all information and links to the data, models, frameworks, and code bases we use in our work. All artifacts were used according to their intended use, as described in their licenses. Upon release, we will also release our code publicly under the MIT License.

Purpose	Name	URL	Details
Code Base	Language Modeling MLM	https://github.com/adapter-hub/adapter-transformers/blob/master/examples/pytorch/language-modeling/run_mlm.py	
	Language Modeling CLM	https://github.com/adapter-hub/adapter-transformers/blob/master/examples/pytorch/language-modeling/run_clm.py	
Models	gpt2-base	https://huggingface.co/gpt2	12-layers, 768-hidden, 12-heads, 117M parameters
	gpt2-large	https://huggingface.co/gpt2-large	36-layers, 1280-hidden, 20-heads, 774M parameters
	deberta-base	https://huggingface.co/microsoft/deberta-base	12-layers, 768-hidden, 12-heads
	SentenceBert	https://github.com/UKPLab/sentence-transformers	Configuration: all-mpnet-base-v2
Frameworks	nltk==3.7		We use NLTK for punctuation removal, stemming and tokenization before creating the TF-IDF vectors.
	adapter-transformers==3.2.1 huggingface-hub==0.13.4 torch==2.0.0 torchaudio==2.0.1 torchvision==0.15.1 transformers==4.28.1 datasets==2.11.0		
Datasets	C4	https://github.com/allenai/c4-documentation	License: ODC-BY
	yelp.com	https://www.yelp.com/dataset	Licence: https://s3-media0.fl.yelpcdn.com/assets/srv0/engineering_pages/f64cb2d3efcc/assets/vendor/Dataset_User_Agreement.pdf

Table 4: Links and explanations to code bases, datasets, models and frameworks used in our work.

B TF–IDF Equation

We determine the TF–IDF scores by:

$$tfidf(t,d) = tf(t,d) * idf(t)$$

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$idf(t) = log\left(\frac{1+N}{1+df(t)} + 1\right),$$

where N is the total number of documents.

C Comparison of Combination Strategies

We evaluate the combination strategies for three different models. In Figure 6, we present the results for ensembling and parameter averaging for `gpt2-large`. Compared to the results for `gpt2-base` and `deberta-base`, which we showed in Figure 2, we did not run the experiments for all values for k between [0,10] because of the size of the model. However, we find very similar patterns in the variation of perplexity across the different strategies and number of adapters added as for `gpt2-base`. This reinforces the validity of our findings.

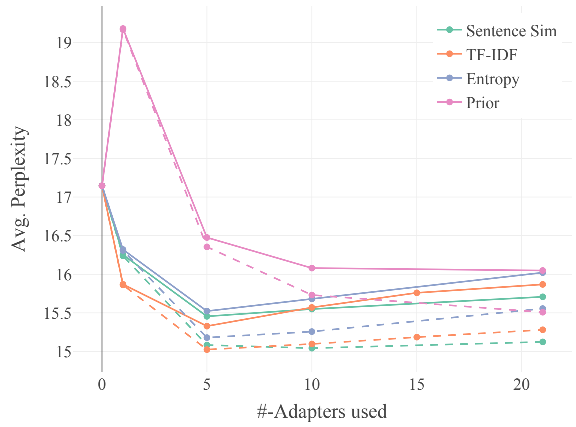
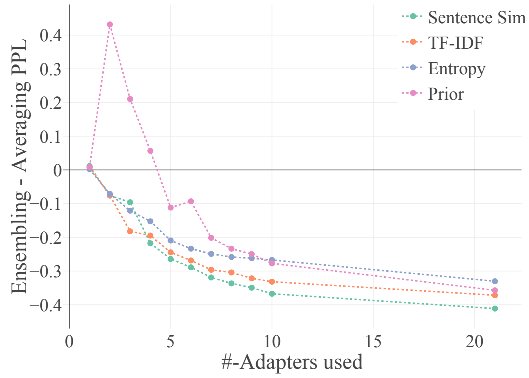


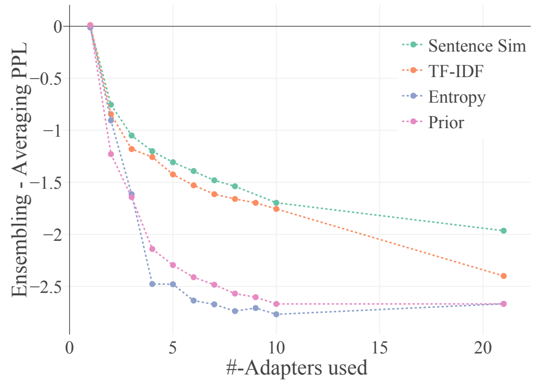
Figure 6: Comparison between Parameter Averaging (solid lines) and Ensembling (dashed lines) for `gpt2-large` over different numbers of top-k adapters. We show the mean perplexity results when using each of our four scoring strategies (SENTSIM, TF–IDF, ENTROPY, PRIOR) averaged across four runs.

Figure 7 additionally shows the perplexity difference between parameter averaging and ensembling for the different scoring strategies. A negative value indicates that ensembling provides lower perplexity values than parameter averaging.

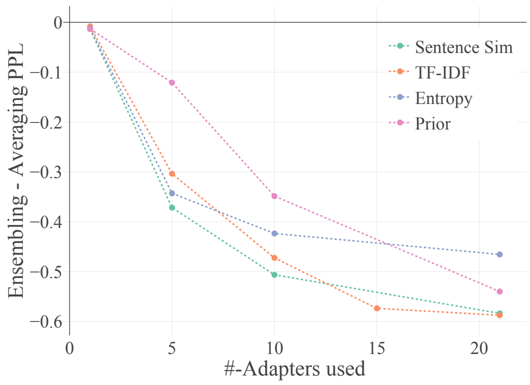
Interestingly, we can see the same tendency for all three models. With an increasing value of k, the difference between parameter averaging and ensembling increases as well, although this effect flattens for $k > 10$. For `deberta-base`, this effect can be seen more strongly. Interestingly, while for `deberta-base`, the difference is larger for model-based approaches, we see an exact opposite effect for the GPT-models.



(a) `gpt2-base`



(b) `deberta-base`



(c) `gpt2-large`

Figure 7: Difference between Ensembling - Parameter Averaging over different numbers of top-k adapters. We show the mean perplexity differences for (a) `gpt2-base`, and (b) `deberta-base` (c) `gpt2-large` when using each of our four scoring strategies (SENTSIM, TF–IDF, ENTROPY, PRIOR) averaged across four runs.

D Meta Regression

We present the coefficients of linear regression for `gpt2-base`, `deberta-base` and `gpt2-large`. We do not include coefficients with an importance value between [-0.1, 0.1].

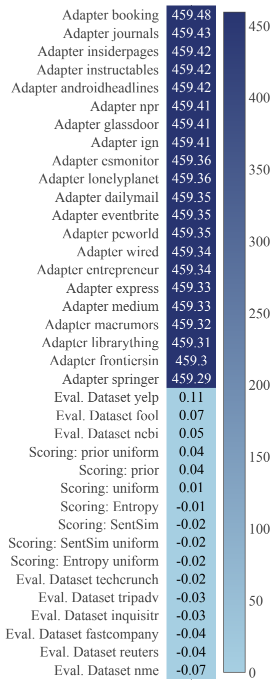


Figure 8: Heatmap of the coefficients of the Linear Regression for `gpt2-base`

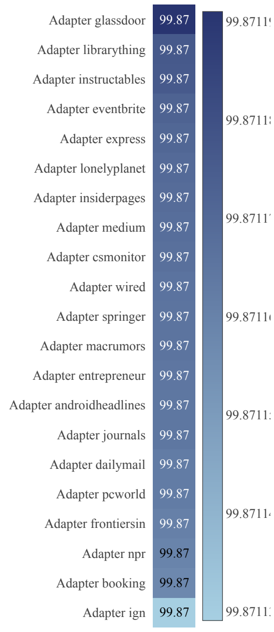


Figure 9: Heatmap of the coefficients of the Linear Regression for `deberta-base`

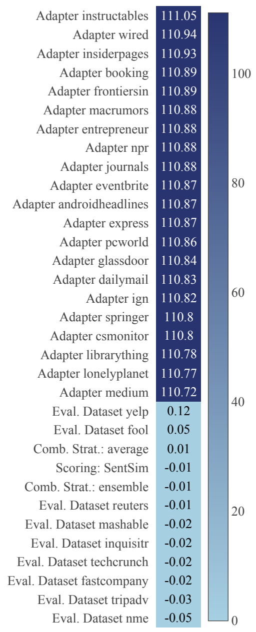
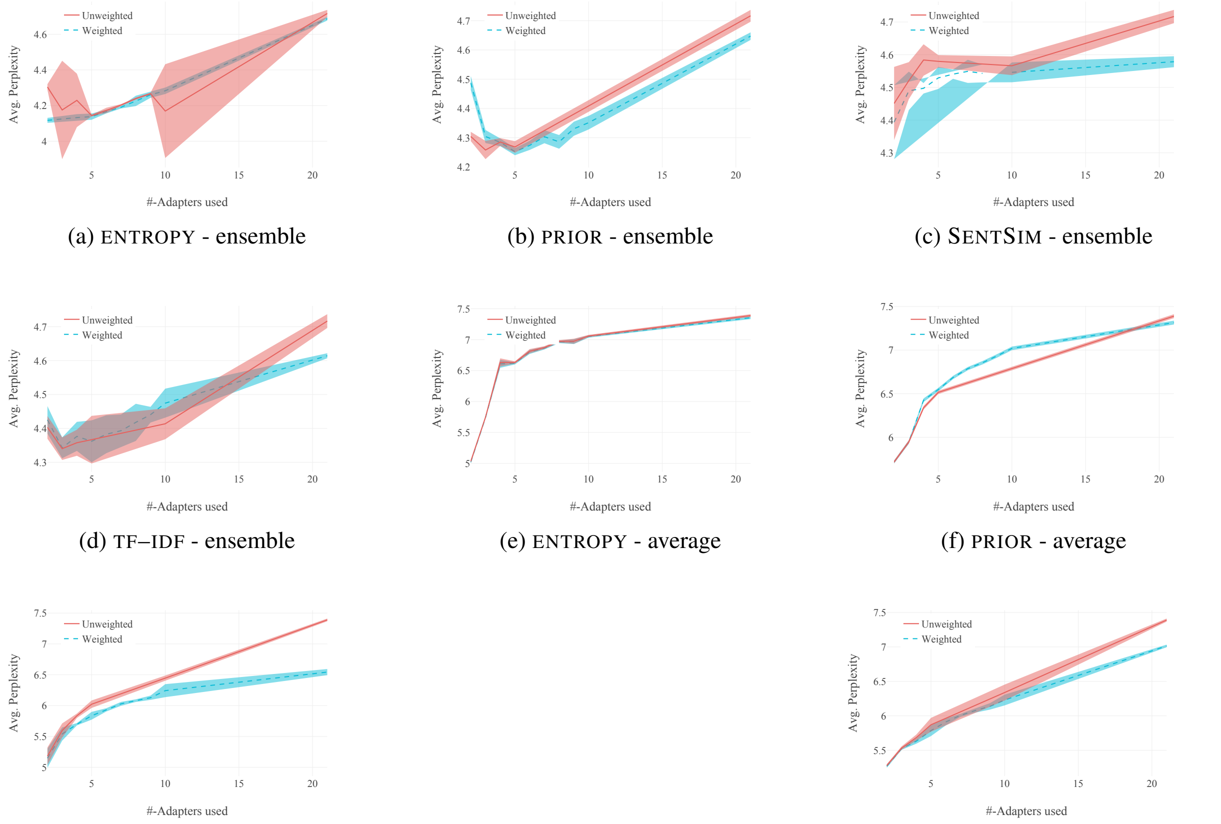


Figure 10: Heatmap of the coefficients of the Linear Regression for `gpt2-large`

E Further Evaluation of Adapter Scorings

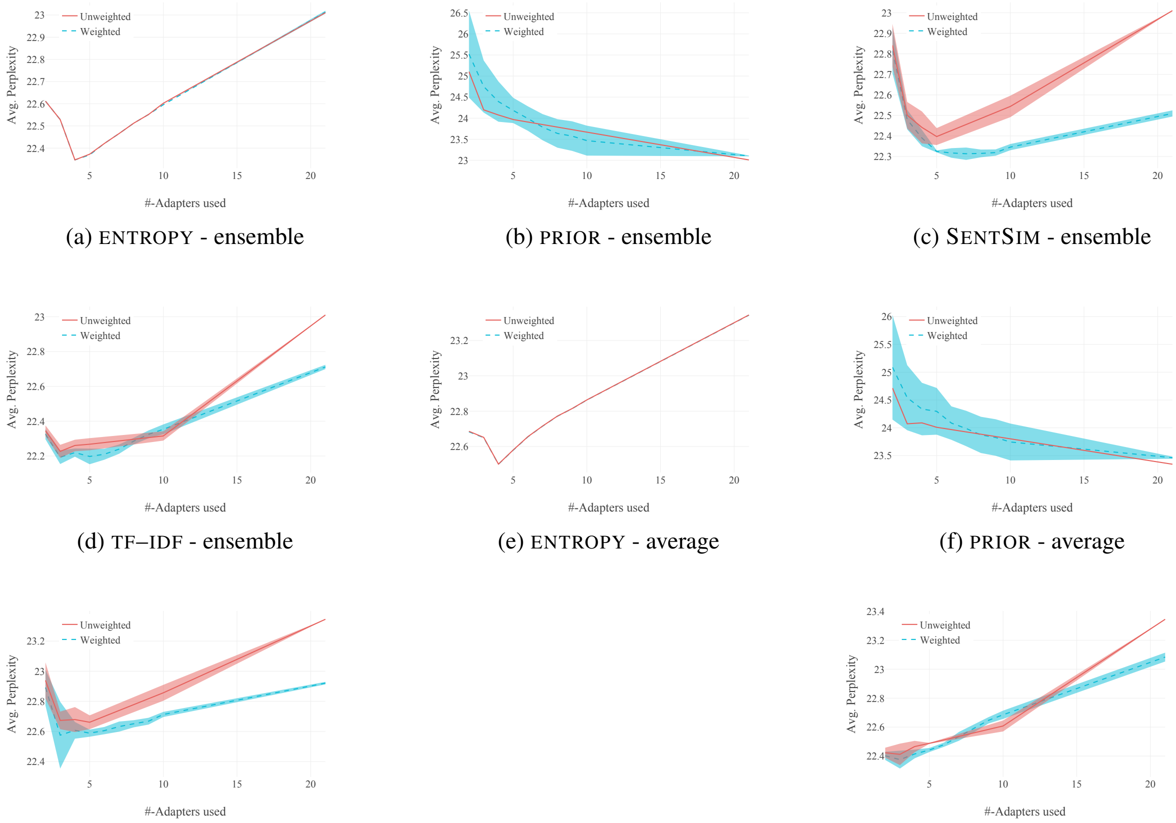


(a) ENTROPY - ensemble (b) PRIOR - ensemble (c) SENTSIM - ensemble

(d) TF–IDF - ensemble (e) ENTROPY - average (f) PRIOR - average

(g) SENTSIM - average (h) TF–IDF - average

Figure 11: Comparison between weighting the selected adapters based on their similarity (blue) and assigning them uniform weights (red). We show the mean perplexity results averaged over all evaluation datasets and across four runs for `deberta-base` when using different pairings of scoring and combination strategies of our framework.



(a) ENTROPY - ensemble

(b) PRIOR - ensemble

(c) SENTSIM - ensemble

(d) TF–IDF - ensemble

(e) ENTROPY - average

(f) PRIOR - average

(g) SENTSIM - average

(h) TF–IDF - average

Figure 12: Comparison between weighting the selected adapters based on their similarity (blue) and assigning them uniform weights (red). We show the mean perplexity results averaged over all evaluation datasets and across four runs for `gpt2-base` when using different pairings of scoring and combination strategies of our framework.

F Efficiency of DeBERTa

We present the results of the efficiency calculations for `deberta-base` in Figure 13. As expected, the plot shows the same pattern as for `gpt2-base`, with a linear increase in CO_2Emissions for a higher number of k.

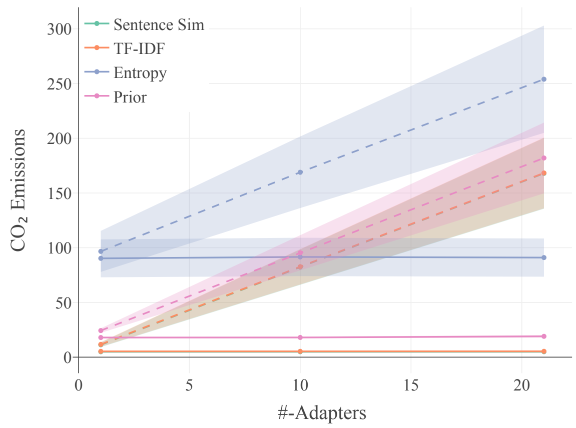


Figure 13: Comparison between the different selection and composition strategies with regards to their efficiency. We present the average CO_2Emissions for experiments where we conducted Parameter Averaging (solid lines) and Ensembling (dashed lines) over different numbers of top-k adapters. We show the results for `deberta-base` when using each of our four scoring strategies (SENTSIM, TF–IDF, ENTROPY, PRIOR) averaged across four runs.

G Threshold Tuning via Early Stopping

In this additional experiment, we tried to estimate the optimal number of adapters to select by applying an early stopping algorithm, whenever we see a sudden drop in adapter similarity.

For this experiment, we use the weighting strategies using TF–IDF and SENTSIM, since these exhibited the largest variation in similarity weights. We then sort these weights from largest to smallest representing the adapter with the respective importance for the novel evaluation domain. We then iterate over the adapter weights and stop if the difference between the weights is larger than a certain threshold. We illustrate this procedure in Figure 14. We run several experiments with different values set for the stopping threshold (see Table 5) and find that with a threshold of 0.004, we are able to obtain on average over all datasets and combination strategies 79% of the optimal model performance.

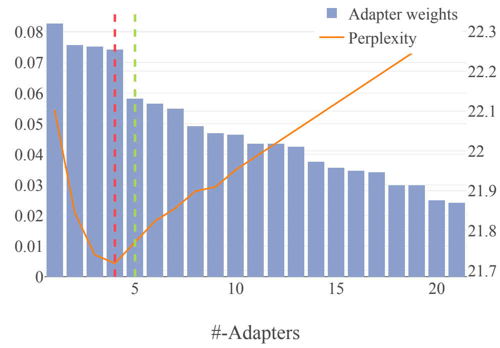


Figure 14: Visualization of the early stopping approach. The red vertical line marks the adapter combination leading to the result with the lowest perplexity. The vertical green line marks the number of adapters that would be chosen when applying the early stopping mechanism. The orange line shows the perplexity change when adding more adapters for this strategy. In this case, we show the results for `gpt2-base` on the techcrunch domain using TF–IDF and ensemble the output.

Threshold	SENTSIM - average	TF–IDF - average	average	SENTSIM - ensemble	TF–IDF - ensemble	ensemble	Total
0.001	0.64	0.84	0.74	0.55	0.73	0.64	0.69
0.002	0.64	0.84	0.74	0.55	0.73	0.64	0.69
0.003	0.67	**0.88**	0.77	0.57	0.79	0.68	0.73
0.004	0.78	**0.88**	**0.83**	0.70	**0.80**	**0.75**	**0.79**
0.005	**0.79**	0.82	0.80	**0.73**	0.77	**0.75**	0.78
0.006	0.74	0.79	0.77	0.69	0.78	0.74	0.75
0.007	0.74	0.74	0.74	0.69	0.73	0.71	0.73
0.008	0.73	0.65	0.69	0.69	0.68	0.69	0.69
0.009	0.73	0.42	0.57	0.69	0.47	0.58	0.58
0.01	0.75	0.42	0.58	0.72	0.47	0.60	0.59

Table 5: Results for threshold tuning for an automatic selection of the best value for k. We show the percentage of how close we can get to the optimal value of k with the respective threshold. We present the average of this percentage over each scoring strategy (TF–IDF and SENTSIM) paired with each combination strategy, each combination strategy alone, and overall (Total).