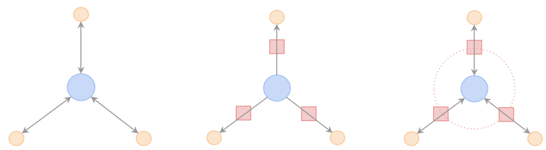
# Searching for Effective Multilingual Fine-Tuning Methods: A Case Study in Summarization

**Anonymous ACL submission**


## Abstract

Recently, a large number of tuning strategies have been proposed to adapt pre-trained language models to downstream tasks. In this paper, we perform an extensive empirical evaluation of various tuning strategies for multilingual learning, particularly in the context of text summarization. Specifically, we explore the relative advantages of three families of multilingual tuning strategies (a total of five models) and empirically evaluate them for summarization over 45 languages. Experimentally, we not only established a new state-of-the-art on the XL−Sum dataset but also derive a series of observations that hopefully can provide hints for future research on the design of multilingual tuning strategies.[1]

(a) (b) (c)

Figure 1: Different frameworks for multilingual learning, where orange circles represent different languages and blue circles denote pre-trained language models (PLMs). Red boxes refer to additional learnable parameters, such as adapters or prefixes. Double sided arrows represent that the parameters of PLMs are tunable.

## 1 Introduction

Methods that perform fine-tuning of pre-trained language models (PLMs) now represent the state-of-the-art across a wide variety of NLP tasks (Howard and Ruder, 2018; Han et al., 2021). Because there are a myriad of methods for tackling this important task of fine-tuning LMs, there is an increasing body of research investigating the empirical strengths and weaknesses of different tuning strategies across several tasks (Peters et al., 2019; Mahabadi et al., 2021; Karimi Mahabadi et al., 2021; Li and Liang, 2021; Mao et al., 2021; Hu et al., 2021; Min et al., 2021; He et al., 2021). One of the major design dimensions of these works revolves around *which set of model parameters are updated*; should fine-tuning only adjust a few additional parameters that are not part of the initial LMs (e.g., Adapters (Houlsby et al., 2019), or Prefix Tuning (Li and Liang, 2021; Xue et al., 2021)), update all parameters of the pre-trained models (Dai and Le, 2015; Devlin et al., 2019; Schick and Schütze, 2021), or update only a subset of parameters (Guo et al., 2020)?

At the same time, there has been much progress in multilingual models based on pre-trained LMs (Lample and Conneau, 2019; Conneau et al., 2020; Liu et al., 2020) However, there is a notable gap in the literature – to our knowledge, there are no comprehensive comparative studies on how different tuning strategies behave in multi-lingual scenarios – there is significant work on multilingual adapters (Pfeiffer et al., 2020b; Ansell et al., 2021) and parameter tying across languages (Sachan and Neubig, 2018; Lin et al., 2021), but few studies comparing different families of methods.

In this paper, we try to fill this gap by performing a comprehensive study of different parameter tuning techniques in the context of text summarization (Rush et al., 2015; Nallapati et al., 2016; Chopra et al., 2016; Lewis et al., 2020; Zhang et al., 2020; Dou et al., 2021). We focus particularly on summarization as previous work on parameter-efficient tuning has noted that the differences between tuning techniques are particularly salient in more complex generative tasks such as summarization, as opposed to text classification (He et al., 2021). We draw on the techniques examined in the monolingual scenario and combine the unique characteristics of the multilingual scenario (e.g., shared features across languages) to derive different architectures for multilingual learning. These

[1] Code at https://github.com/anonymous-717/Multi-Sum

frameworks encompass some existing works on multilingual learning, but also allow us to propose new learning methods and perform comparisons between different frameworks.

Figure 1 (a) shows a commonly-used framework (Hasan et al., 2021) in which one tunable pre-trained model is shared by different languages. Figure 1 (b) introduces a language-specific module with learnable parameters while keeping the PLM's parameter frozen, i.e. parameter-efficient tuning (Mao et al., 2021; He et al., 2021). In practice, the language-specific module could be instantiated as an adapter, prefix, or other variety of extra parameters. Notably, this kind of language-specific module is independent for each language and cannot share information, thus low-resource languages cannot benefit from other related languages. Figure 1 (c) tries to alleviate this problem in two ways: making parameters of pre-trained models tunable or introducing additional modules whose parameters can be shared by different languages.

Using this framework, we ask these questions:

Q1: How well do different parameter-efficient tuning methods (Figure 1-b) perform compared to PLM fine-tuning models (Figure 1-a) in multilingual summarization? Q2: Will supervised transfer, a commonly used technique in multi-lingual learning, be helpful for parameter-efficient tuning? Q3: Could better results be achieved by enabling information exchange between different languages? How do different choices of parameter-efficient tuning methods interact with this sharing?

We explore these questions by performing extensive experiments over 45 different languages . Our quantitative and qualitative analyses find several observations, such as:
(1) Compared to PLM fine-tuning, both parameter-efficient tuning methods (prefix- and adapter-tuning) are advantageous in low-resource scenarios. Particularly, prefix-tuning outperforms adapter-tuning with extremely few samples over different languages §3.1. (2) Parameter-efficient tuning is possible to fail in the *supervised transfer* setting (§3.2), where pre-trained language models are fine-tuned on the source languages whose scripts are distant from the target language's. (3) Adding language specific adapters or prefixes while additionally tuning the PLM's parameters, can maintain multi-lingual PLM fine-tuning's advantage of sharing information among languages, as well as preserving private parameters for each language to reduce the negative effect of the limited capacity of one LM shared by all languages. We achieve a new state-of-the-art performance with such a multilingual tuning strategy.

## 2 Preliminaries

### 2.1 Task Formulation

Abstractive summarization can be formulated as a conditional generation task where the input $D$ is a document, and the output $S$ is a short summary. The majority of state-of-the-art models for abstractive summarization use encoder-decoder models (Sutskever et al., 2014), where an encoder generates representations for the source document $\mathbf{D} = [\mathbf{d}_1, ..., \mathbf{d}_m]$, and a decoder outputs the summary $\mathbf{S} = [\mathbf{s}_1, ..., \mathbf{s}_n]$ one target token at a time. The conditional probability of a single sample is modeled as $p(\mathbf{s}^i|\mathbf{d}^i;\theta)$, and hence parameters $\theta$ are obtained by maximum likelihood estimation

$$\underset{\theta}{\operatorname{argmax}} \sum_{(\mathbf{d}^i,\mathbf{s}^i)\in(D,S)} \log p(\mathbf{s}^i|\mathbf{d}^i;\theta), \qquad (1)$$

where $(D, S)$ is the parallel training corpus. For multilingual text summarization, $D$ and $S$ can be in any of a number of languages.

### 2.2 Tuning Strategy

Recently, applying pre-trained language models (PLMs) to abstractive summarization tasks equipped with diverse tuning strategies has achieved a great success, which can be formulated as below:

$$h_i = \text{PLM}(D, s_i, h_{<i}; \theta_{\text{plm}}, \theta_{\text{add}}) \qquad (2)$$

where PLM is a sequence to sequence pre-traiend LMs (e.g., T5 (Raffel et al., 2019) or BART (Lewis et al., 2020)), $\theta_{\text{plm}}$ represents the original PLM parameter and $\theta_{\text{add}}$ denotes the additional parameters added by different tuning strategies.

Based on whether and when parameters $\theta_{\text{plm}}$ and $\theta_{\text{add}}$ will be tuned, different tuning strategies as illustrated in Fig. 2 have been explored, which we will detail below for the better introduction of multilingual tuning strategies.

**PLM Fine-tuning** This is one of the most common tuning strategies that aim to tune all of the parameters $\theta_{\text{plm}}$. While PLM fine-tuning has achieved strong performance on many benchmarks, one major limitation lies in the requirement of large training samples, which is not feasible in the low-resource scenario.

To alleviate this issue, parameter-efficient tuning has been extensively explored recently, among which we select two representative methods which are initially designed for generation tasks, consistent with our goal.

**Adapter-tuning** Adapter-tuning adds additional lightweight layers between the layers of an existing PLM. Although there is a variety of ways to define "adapter", we adopt the definition of (Bapna et al., 2019). Specifically, the adapter block consists of (1) a layer normalization $\mathrm{LN}(\cdot)$ for the input of the adapters, (2) an autoencoder whose inner dimension can be adjusted according to the complexity of the target task with a down projection layer, an up projection layer, and a nonlinear activation function between them, and (3) a residual connection. Formally, given $h_i \in \mathbb{R}^d$ be the output of i-th layer, the adapter is formulated:

$$\textsc{Adapter}(h_i) = (\mathrm{ReLU}(\mathrm{LN}(h_i)\mathbf{W}_i^{db}))\mathbf{W}_i^{bd} + h_i,$$

where b is the inner dimension, $\mathbf{W}_i^{db} \in \mathbb{R}^{d\times b}$ is the weight of down projection layer and $\mathbf{W}_i^{bd} \in \mathbb{R}^{b\times d}$ is the weight of up projection layer.

**Prefix-tuning** Prefix-tuning (Li and Liang, 2021) prepends a prefix for every layer of a LM. Let $\mathbf{H}_i^{LM} \in \mathbb{R}^{t\times d}$, where $d$ is the hidden dimension of LM, $t$ is the input sequence length, denote the hidden representation of the i-th layer. We prepend prefixes at each layer to obtain $\mathbf{H}_i = [\mathrm{Prefix}_i; \mathbf{H}_i^{LM}] \in \mathbb{R}^{(t+l)\times d}$, where $l$ is the prefix length, $\mathrm{Prefix}_i \in \mathbb{R}^{l\times d}$ is prepended prefix.

We can look up trainable matrix $\mathbf{P}_\theta \in \mathbb{R}^{l\times(d\times n)}$, where $n$ is the number of layers of the LM, to get $\mathrm{Prefix}_i$. However, according to (Li and Liang, 2021), reparameterization has better performance than directly updating $\mathbf{P}_\theta$ in practice. So we reparametrize the matrix $\mathbf{P}_\theta = \mathrm{MLP}(\mathbf{P}'_\theta)$, where $\mathrm{MLP}(\cdot)$ has the structure of an autoencoder with a tunable middle dimension size, and $\mathbf{P}'_\theta$ is a smaller matrix with dimension $l \times d$.

### 2.3 Multilingual Tuning Methods

Based on the above-mentioned tuning strategies in single language scenarios, we investigate three different multilingual learning frameworks and explore their applicable scenarios in detail.

**Multilingual PLM Fine-tuning (MPF)** This is a commonly-used setting (Hasan et al., 2021) when training samples from different languages are provided. Summarization systems share one multilingual pre-trained language model whose parameters can be updated by any system.

**Multilingual Parameter-efficient Tuning (MPE)** In this framework, additional private parameterized modules such as prefix or adapter are introduced for each system besides one shared multilingual pre-trained language model, whose parameters keep frozen. Some existing works (Bapna et al., 2019) follow this framework but mainly focus on the use of adapters.

**Multilingual Private-shared Tuning (MPS)** In the above method, although systems of different languages share one pre-training model, their parameters cannot be modified, which results in the lack of information interaction across languages and the difficulty in mining the shared knowledge. In this framework, parameters from both additional modules and pre-trained models can be updated.

## 3 Experiments

**Dataset** As our evaluation testbed, we use the `XL-Sum` corpus (Hasan et al., 2021),[2] which is a news dataset containing 1.1 million article-summary pairs in 45 languages. The dataset is collected from the British Broadcasting Corporation (BBC) website, using a bold paragraph at the beginning of each article as the summary and the rest of the article as the input text. We choose XL-sum for its: (1) high language coverage, including low resource, medium resource, and high resource languages, (2) similar intrinsic characteristics, e.g. novel n-gram ratio, abstractivity, and compression among all samples, allowing our analysis to focus on the differences across languages, other than different intrinsic features across samples.

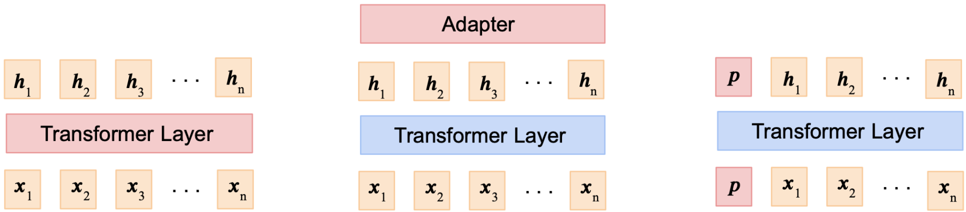
**Evaluation Metric** As is standard in summarization, we use ROUGE (Lin, 2004) as our evaluation metric, which computes the n-gram similarity between the gold and the generated summary.[3]

### 3.1 Exp-I

To answer the question (Q1) of how well different parameter-efficient tuning methods behave compared to standard LM fine-tuning in the multilingual setting, we study the performance of three

[2]License: CC BY-NC-SA 4.0.

[3]We use the rouge package provided by `XL-Sum`(Hasan et al., 2021) to support multiple languages.




(a) PLM Fine-tuning (b) Adapter-tuning (c) Prefix-tuning

Figure 2: Different tuning methods. Red indicates trainable parameters; blue indicates frozen parameters.

tuning methods: prefix-tuning, adapter-tuning and PLM fine-tuning on different languages.

#### 3.1.1 Experiment Details

**Settings and Hyper-parameters** We use the base version of multilingual T5 (Xue et al., 2021) as a backbone, which covers most languages in `XL-Sum` dataset and is the same as (Hasan et al., 2021), allowing us to make a fair comparison. In our experiment, MLP of prefix-tuning is two linear layers with an inner dimension as a hyper-parameter. For prefix-tuning, the hyper-parameters we tune[4] are the same as (Li and Liang, 2021). For adapter-tuning, the hyper-parameters remain the same as prefix-tuning except the prefix length that is not needed for adapter-tuning. More details are in the appendix. For both adapter-tuning and prefix-tuning, these hyper-parameters lead to about 8% additional parameters compared to the LM's total parameters, which are tuned during training while the LM's parameters are frozen. To study whether language features will influence the choice of tuning methods, we choose five languages from different language families: `English` (Germanic), `Chinese simplified` (Sino Tibetan), `Spanish` (Romance) , `Ukrainian` (Balto Slavic) and `Urdu` (Indo Iranian).

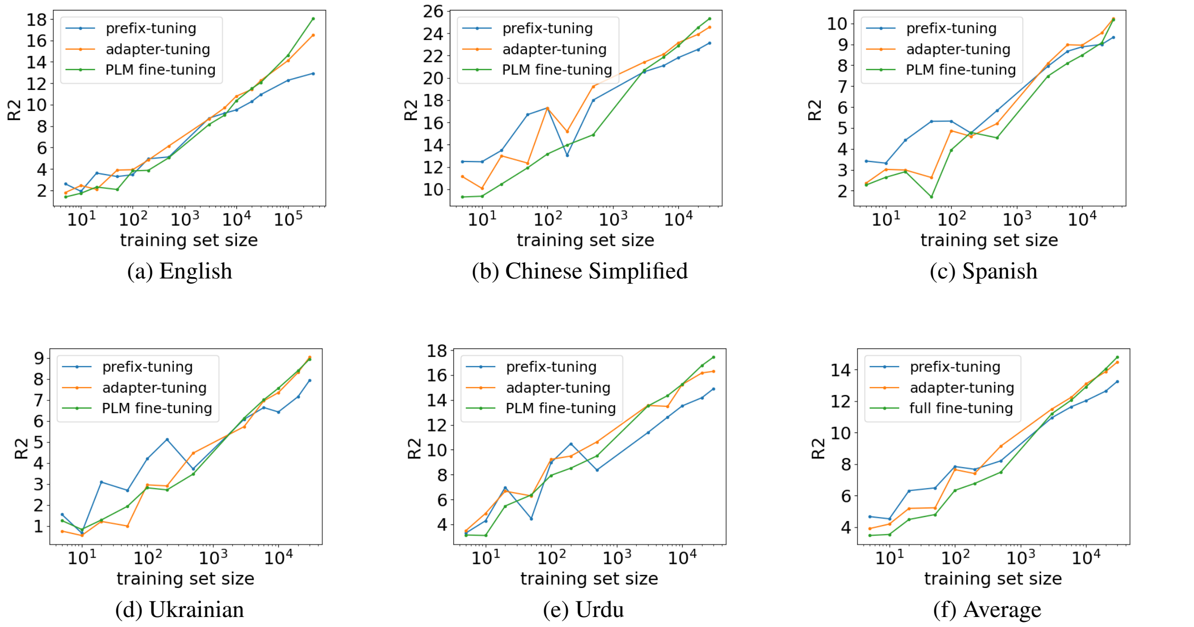
We subsample the full dataset of each language to obtain sub-datasets of various sizes,[5] and sub-datasets of size $\leq$ 500 are considered "few-shot" experiments. For each few-shot experiment, we randomly sample 3 different training sets and development set (with dev size = 20% training set size). The reported result is the average of 3 experiments on the full test set of the chosen language. The performance of few shot experiments is influenced by the training samples chosen (Zhao et al., 2021), so we keep the sampled training set and development set the same for the three tuning methods to have a fair comparison. For non-few shot experiments, each size has one experiment and is tested on the full test set of the chosen language. The hyperparameters are chosen from a single language (Japanese) for each tuning method and applied as is to all languages. We use the result from the checkpoint with the best validation set performance over all training epochs.

#### 3.1.2 Results and Analysis

**Results** Fig.3 illustrates the performance of three different tuning methods with respect to the available training samples, observations are:
(1) In general when the sample number is less than 200 prefix-tuning achieves the best performance. Between 200 and 10k adapter-tuning is superior, and greater than 10k PLM fine-tuning surpasses the other two. This indicates that *regarding both the performance and parameter efficiency (only tune 8% of the parameters of PLM fine-tuning), prefix-tuning is the best choice when we have extremely few samples, while adapter-tuning is the winner in medium resource settings*. (2) As the training set size increases from few shot to high resource, PLM fine-tuning has the largest performance improvement, while prefix-tuning has the least performance improvement and adapter-tuning is the middle. (3) Compared to PLM fine-tuning, which is almost monotonically increasing with the training set size, both adapter-tuning and prefix-tuning have some fluctuations. From preliminary experiments, we find that adapter-tuning and prefix-tuning are more sensitive to learning rate than PLM fine-tuning. Fixing two separate learning rates for few shot and non-few shot experiments for all languages, a simplification of the normal training process to find

[4] Specifically, the number of epochs, batch size, learning rate, prefix length and inner dimension during training, beam search size and length penalty during inference.

[5] Concretely, {5, 10, 20, 50, 100, 200, 500, 3000, 6000, 10000, 20000, 30000}. For English, which has far more training samples than other languages, we add two training set size 100000 and 300000.




(a) English (b) Chinese Simplified (c) Spanish

(d) Ukrainian (e) Urdu (f) Average

Figure 3: Performance of prefix-tuning, adapter-tuning and PLM fine-tuning on five languages over training set sizes. The x axis is the number of training samples at log scale, the y axis is the ROUGE-2 score.

an optimal learning rate for each training set size and each language on the development set, might cause the unstable performance of the tuning methods that are more sensitive to learning rate. (4) All above observations are roughly true for every language and is especially clear in the average plot.

**Discussion & Takeaways** The possible explanation of the different behavior of adapter-tuning and prefix-tuning is their structural discrepancy, in which adapter-tuning adds parameters between two transformer layers, while prefix-tuning uses these parameters to generate prefixes and append the prefixes at the front of each transformer layer. Similar to prolong the input, prefix-tuning does not touch the PLM's architecture, preserving the knowledge in PLM. Thus, by utilizing the PLM better, prefix-tuning has better performance at few shot when there are not enough samples to learn new knowledge. However, while the training set size becomes larger, more flexible structures are needed to learn from these samples, leading to better performance of adapter-tuning. The possible reason why both prefix-tuning and adapter-tuning outperform PLM fine-tuning at the left side is that prefix-tuning and adapter have only 8% parameters to tune, which avoid overfitting when the training samples are not enough.

From this experiment, we can see that prefix-tuning has both advantages of utilizing the knowledge of PLM better and fewer parameters to tune, while adapter-tuning only benefits from the latter. This reminds us that while designing a prompt, one important thing is to keep the PLM's architecture and make the prompt as natural as possible to better extract knowledge from PLM.

### 3.2 Exp-II

When designing models in multi-lingual scenarios, one crucial question is how to make modules of different languages communicate efficiently so that shared knowledge can be fully utilized. There are two ways to do this: (1) by transferring: first fine-tune PLMs on multiple languages except the one we concerned about (a.k.a target language), and then adapt the fine-tuned multi-lingual model to the target language; (2) by multitask learning: jointly train all languages together. In this experiment, we study method 1 and try to answer the Q2: will supervised transfer be helpful for parameter-efficient tuning. In Exp.3.3, we study method 2.

#### 3.2.1 Experiment Details

**Settings and Hyper-parameters** We first divide `XL-Sum` dataset into two parts, one of which including 34 languages (75% of total languages) is used to fine-tune PLM jointly on multiple languages to obtain a single multi-lingual model,

| Language | Script | mt5-base | | | | | | mt5-base34 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | prefix-tuing | | | adapter-tuing | | | prefix-tuing | | | adapter-tuing | | |
| | | R1 | R2 | RL | R1 | R2 | RL | R1 | R2 | RL | R1 | R2 | RL |
| amharic | Geez | 15.33 | 5.42 | 13.8 | 16.58 | 5.88 | 14.88 | 16.97 | 5.88 | 15.12 | 17.85 | 6.2 | 16.01 |
| azerbaijani | Cyrillic | 15.72 | 6.3 | 14.47 | 17.81 | 7.34 | 16.13 | 11.9 | 3.4 | 10.93 | 12.16 | 3.37 | 10.96 |
| bengali | Brahmic | 23.76 | 9.11 | 20.66 | 25.99 | 10.07 | 22.21 | 0 | 0 | 0 | 0 | 0 | 0 |
| burmese | Brahmic | 12.74 | 3.46 | 11.51 | 14.07 | 4.09 | 12.54 | 1.57 | 0.38 | 1.51 | 1.14 | 0.22 | 1.08 |
| igbo | Latin | 23.27 | 5.25 | 17.36 | 25.22 | 7.08 | 19.38 | 28.1 | 7.98 | 21.19 | 28.33 | 7.93 | 21.72 |
| japanese | Kan,Hi,Kat | 41.23 | 18.89 | 32.42 | 45.6 | 21.87 | 35.02 | 11.39 | 2.82 | 9.05 | 11.84 | 3.15 | 9.12 |
| scottish_gaelic | Latin | 24.05 | 7.73 | 19.45 | 20.42 | 4.31 | 17.02 | 24.36 | 7.16 | 18.87 | 26.37 | 8.09 | 20.22 |
| spanish | Latin | 28.09 | 9.05 | 21.14 | 29.28 | 10.3 | 22.23 | 29.75 | 9.48 | 22.18 | 30.26 | 9.83 | 22.42 |
| tamil | Brahmic | 16.45 | 6.58 | 15 | 19.81 | 8.83 | 18.12 | 0.42 | 0.02 | 0.42 | 0.44 | 0.02 | 0.43 |
| ukrainian | Cyrillic | 18.73 | 7.07 | 16.43 | 21.84 | 8.92 | 19.08 | 16.46 | 4.41 | 13.99 | 16.96 | 4.61 | 14.32 |
| urdu | Arabic | 35.05 | 14.5 | 28.76 | 38.81 | 17.69 | 32.08 | 20.19 | 4.48 | 15.66 | 19.57 | 4.58 | 15.08 |

Table 1: R1, R2, and RL scores of prefix-tuning, adapter-tuning of mt5-base and mt5-base34 for 11 languages. "Kan, Hi, Kat" is the abbreviation of Kanji, Hiragana, and Katakana.

while another part including 11 languages (25% of total languages) is used to investigate the fine-tuned PLM's ability to generalize to new languages. The 11 left-out languages that do not participate in fine-tuning are chosen according the principle that they are from different language family and have different training set size.[6] The hyper parameters used to fine-tune PLM is the same as the Multilingual training of `XL-Sum` (Hasan et al., 2021).
We refer to mt5-base as the original PLM without fine-tuning and mt5-base34 as the fine-tuned version on 34 languages. We then performed prefix-tuning and adapter-tuning on mt5-base and mt5-base34 for 11 left-out languages. Hyper parameters used for these tuning methods remain the same as those in non-few shot experiments of Sec.(3.1).

#### 3.2.2 Results and Analysis

**Results** Table.1 illustrates the performance of multi-lingual models mt5-base and mt5-base34 to adapt to 11 new languages by prefix-tuning and adapter-tuning. The main observations in Table.1 are as follows:
(1) For four languages, Amharic, Igbo, Scottish Gaelic, and Spanish, mt5-base34 will bring gains against their counterparts by 0.3 to 6.0 R1 score.
(2) Fine-tuning mt5-base on 34 languages of `XL-Sum` dataset jeopardizes the performance of seven languages, among which Bengali, Burmese, Tamil's R1 score becomes near zero.
(3) Three of the four languages with performance improvement adapted from fine-tuned PLM are of Latin script, while all three languages with dramatic performance drop down are of Brahmic script, indicating the important role script plays to determine whether supervised transfer is helpful for parameter-efficient tuning.

**Discussion & Takeaways** Although intuitively new languages will benefit from PLM fine-tuned on `XL-Sum` dataset, the practical results shows that not all languages will obtain improvements. Transfer learning in such a way might cause catastrophic forgetting of the previously acquired knowledge in PLM (McCloskey and Cohen, 1989; Santoro et al., 2016). If there are not enough training samples of a certain script during PLM fine-tuning, the PLM might lose the ability generalizing to languages of this script by parameter-efficient tuning methods and freezing PLM. This indicates that *the effectiveness of parameter-efficient tuning methods under multi-lingual scenarios is highly dependent on the multi-lingual model we use and under some situations, parameter-efficient tuning might lose their adaptivity*. We leave how to alleviate this problem for future work.

### 3.3 Exp-III

In this experiment, we study jointly multi-lingual training to see if different languages can benefit from each other and how does adding private parameters for each language influence the performance of multi-lingual training (Q3).

#### 3.3.1 Experiment Details

With respect to Fig.1, we have 6 different settings to compare single language training with multi-lingual training.
**PLM Fine-tuning (PLF)**: mt5-base is fine-tuned on all languages to obtain separate models for each language as the baseline. **Multi-lingual**

[6] Concretely, 7 languages (Amharic, Azerbaijani, Bengali, Burmese, Igbo, Japanese, Scottish Gaelic, Spanish, Tami) are low resource ($< 15,000$ training samples), 2 languages (Spanish, Tamil) are medium resource ($15,000 \sim 40,000$) and 2 languages (Ukrainian, Urdu) are high resource ($> 40,000$).

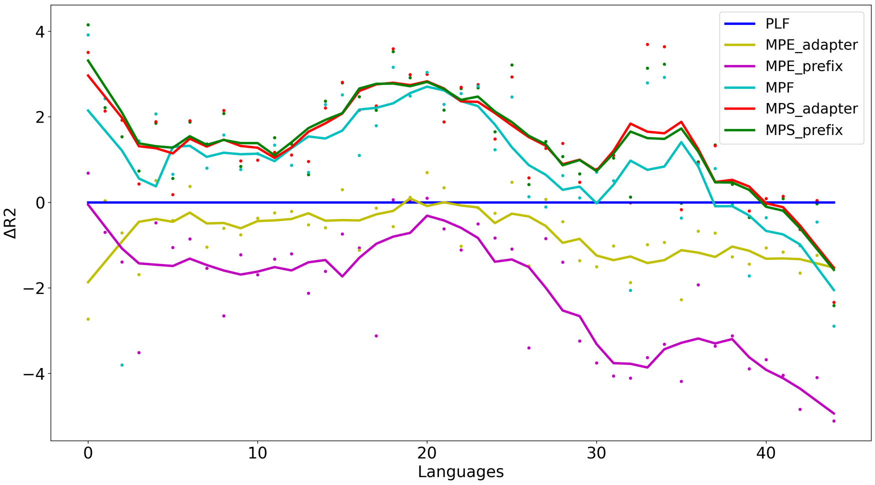



Figure 4: Trend-lines depicting performance improvement. X-axis is the languages, which are arranged in increasing order of available training data from left to right. Y-axis depicts the R2 score relative to the singular language PLM fine-tuning baseline.

**Parameter-efficient (Adapter/Prefix) Tuning (MPE_adapter/MPE_prefix):** We add adapters/prefixes (with parameters = 8% parameters of LM) for each language and tune adapters/prefixes separately for each language while freezing mt5-base which is shared by all languages. **Multi-lingual PLM Fine-tuning (MPF):** A single model is trained with training samples from multiple languages. The training strategy proposed by (Lample and Conneau, 2019) to use a smoothing factor (alpha) of 0.5 to balance the sampling rate of low resource languages and high resource languages is followed by every multilingual setting.[7] **Multi-lingual Private-shared (Adapter/Prefix) Tuning (MPS_adapter/MPS_prefix):** Private adapters/prefixes (with parameters = 2% parameters of LM) for each language (in total 45 languages in `XL-Sum` dataset × 2% parameters for each languages = 90% additional parameters) are added to a single LM. The LM is tuned jointly for multiple languages, while the adapters/prefixes are tuned separately for each language. [8]

[7]We use the results of `XL-Sum`(Hasan et al., 2021). In order to have a fair comparison and remove the influence of different rouge packages, we use their model-generated outputs on test set to calculate the rouge score, instead of using their reported rouge score directly.

[8]One thing to notice is that during training, in each iteration we sample from one language with a certain probability. The LM shared by all languages is tuned every iteration, while the private parameters for each language are tuned whenever the language is sampled.

#### 3.3.2 Results and Analysis

The summarization performance on different languages with different settings is plotted in Figure 4. With combinations of different experiment settings, we have the following results:

**PLF v.s. MPE_prefix v.s. MPE_adapter:** PLF outperforms MPE_prefix and MPE_adapter overall, with a gap larger for more available training samples. MPE_adapter outperforms MPE_prefix for almost every language, except a few languages with few samples. This conforms to the result in Sec.3.1 that fine-tuning has the advantage with large training set size, while prefix-tuning has the advantage with small training set size. One thing to notice is that adapter-tuning has comparable or even higher performance when the training set size is smaller than 10k, consistent with the result in Sec.3.1 and adapter's sensitivity to training set size is not as high as prefix-tuning. The latter is reflected in that the adapter's performance does not drop down dramatically as the training set size increases and keeps within -2 R2 of the baseline for almost all languages, which is even true for English with the highest training set size of 300,000. This means the parameter efficient adapter is a reasonable substitute of PLM fine-tuning regardless of the training set size.

**MPF v.s. PLF:** Multi-lingual model MPF significantly outperforms the baseline PLF in the low and medium resource languages with the gain decreases as the training set size becomes larger. This is expected because low and medium resource languages can benefit from joint training by positive transfer between sister languages (Lample and Conneau, 2019). A deterioration is also observed in the high resource languages. However, the multilingual model within -1 R2 drop down for 6 high resource languages and -3 R2 drop down for English, does not trail a lot. It is a good indication that by training a single multilingual model, the low and medium resource languages have been significantly improved without too much sacrifice of high resource languages. Similar to the low resource experiment of (Hasan et al., 2021), our result is stronger than theirs, which only selects 5 low resource languages to fine-tune individual LM.

**MPF v.s. MPS_adapter v.s. MPS_prefix:** By adding language-specific parameters under multilingual scenarios, compared to MPF, MPS_adapter and MPS_prefix have performance improvements R1:0.73, R2:0.46, RL:0.45 and R1:0.67, R2:0.47, RL:0.46 respectively for 45 languages on average. From each language performance, we can see that all high resource languages have performance improvements at the cost of jeopardizing the performance of a few low resource languages a little. This indicates that sharing LM as well as adding private language-specific parameters will maintain the jointly multi-lingual training's advantage of sharing information among languages, while reducing the harm of sharing all parameters to high resource languages due to the limit model capacity. Besides, the two ways to add additional parameters: private adapter, private prefix for each language have roughly the same overall performance on the whole dataset and the same trend lines depicted in Fig.4, despite their differences we have discussed in Sec.3.1. The possible explanation is that the disadvantage of prefix-tuning lacking the flexibility to modify freeze LM addressed in Sec.3.1, is alleviated or removed by tuning shared LM. Both prefix and adapter's advantage comes from adding private parameters, so they have similar behavior.

## 4 Related Work

### 4.1 Multilingual Tasks

With rapid development of pre-trained LMs, multilingual LMs have emerged to leverage the power of pre-training on a large number of languages, exemplified by **mBERT** (Devlin et al., 2019), **XLM-R** (Conneau et al., 2020), **XLM-R** (Conneau et al., 2020), which adopt masked language model paradigm, and **mBART** (Liu et al., 2020), **mT5** (Xue et al., 2021), which utilize a sequence-to-sequence framework. However, a few works have focused on multilingual summarization given the lack of benchmark datasets for other languages except English. (Giannakopoulos et al., 2015) benchmarked summarization systems over 40 languages, with limitation of dataset scale having 10k samples in total. (Scialom et al., 2020) released the multilingual summarization dataset spanning 5 languages with 1.5M article-summary pairs. (Cao et al., 2020) created a new dataset for two languages with 400k samples. (Hasan et al., 2021) introduced `XL-Sum` spanning 45 languages containing 1.1M article-summary pairs. More recently, (Varab and Schluter, 2021) released MassiveSumm containing 28.8 million articles across 92 languages.

### 4.2 Parameter Efficient Tuning

Parameter-efficient tuning methods only tune a small number parameters to achieve comparable results. It can be roughly divided into two categories, methods without additional parameters and methods with additional parameters. The former tune part of the pre-trained LM. (Lee et al., 2019) fine-tunes a few of the final layers, while (Min et al., 2021) only fine-tunes the bias terms of the LM. The latter introduces extra parameters while fixing the pre-trained LM. Popular methods include **adapter-tuning** (Houlsby et al., 2019; Bapna et al., 2019; Pfeiffer et al., 2020a) **prefix-tuning** (Li and Liang, 2021) **prompt-tuning** (Lester et al., 2021), and others (Mao et al., 2021; Hu et al., 2021; Guo et al., 2020). Among these works, a comprehensive discussion in the context of multilingual summarization is relatively missing.

## 5 Discussion

In this paper, we investigate the applicable scope of different families of tuning strategies for multilingual learning. We specifically ask three research questions, and by extensive experiments on summarization datasets with 45 languages we obtain diverse observations which, hopefully, would provide a useful instruction for future designing of multilingual tuning strategies. One limitation of our work is that we only conduct experiments on one summarization dataset, and more other NLP tasks could be explored as future work.

## 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, Naveen Arivazhagan, and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. *arXiv preprint arXiv:1909.08478*.

Yue Cao, Xiaojun Wan, Jinge Yao, and Dian Yu. 2020. Multisumm: Towards a unified model for multilingual abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11–18.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28:3079–3087.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. GSum: A general framework for guided neural abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4830–4842, Online. Association for Computational Linguistics.

George Giannakopoulos, Jeff Kubina, John Conroy, Josef Steinberger, Benoit Favre, Mijail Kabadjov, Udo Kruschwitz, and Massimo Poesio. 2015. MultiLing 2015: Multilingual summarization of single and multi-documents, on-line fora, and call-center conversations. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 270–274, Prague, Czech Republic. Association for Computational Linguistics.

Demi Guo, Alexander M Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, et al. 2021. Pre-trained models: Past, present and future. *AI Open*.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

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. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. 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 565–576, Online. Association for Computational Linguistics.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Zehui Lin, Liwei Wu, Mingxuan Wang, and Lei Li. 2021. Learning language specific sub-network for multilingual machine translation. 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 293–305, Online. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *arXiv preprint arXiv:2106.04647*.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen-tau Yih, and Madian Khabsa. 2021. Unipelt: A unified framework for parameter-efficient language model tuning. *arXiv preprint arXiv:2110.07577*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2021. Noisy channel language model prompting for few-shot text classification. *arXiv preprint arXiv:2108.04106*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Devendra Sachan and Graham Neubig. 2018. Parameter sharing methods for multilingual self-attentional translation models. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 261–271, Belgium, Brussels. Association for Computational Linguistics.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*.

Timo Schick and Hinrich Schütze. 2021. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. Mlsum: The multilingual summarization corpus. *arXiv preprint arXiv:2004.14900*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Daniel Varab and Natalie Schluter. 2021. Massivesumm: a very large-scale, very multilingual, news summarisation dataset. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10150–10161.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Mengjie Zhao, Yi Zhu, Ehsan Shareghi, Ivan Vulić, Roi Reichart, Anna Korhonen, and Hinrich Schütze. 2021. A closer look at few-shot crosslingual transfer: The choice of shots matters. 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 5751–5767.

# A Supplementary Material

## A.1 Hyper-parameters

In Table2, we report the hyper-parameters used in Sec.3.1.

## A.2 Additional Results

Table3 supplements Fig.4 in Sec.3.3.

| | learning rate | epoch | batch size | grad acc | prefix length | inner dim | beam size | length penalty |
|---|---|---|---|---|---|---|---|---|
| prefix-tuning | | | | | | | | |
| few shot setting | 3.00e-4 | 20 | 2 | 1 | 20 | 800 | 4 | 0.6 |
| not few shot setting, low resource | 2.00e-4 | 15 | 8 | 1 | 200 | 800 | 4 | 0.6 |
| not few shot setting, not low resource | 2.00e-4 | 15 | 16 | 4 | 200 | 800 | 4 | 0.6 |
| adapter-tuning | | | | | | | | |
| few shot setting | 1.00e-3 | 20 | 2 | 1 | - | 1200 | 4 | 0.6 |
| not few shot setting, low resource | 1.00e-3 | 15 | 8 | 1 | - | 1200 | 4 | 0.6 |
| not few shot setting, not low resource | 1.00e-3 | 15 | 16 | 4 | - | 1200 | 4 | 0.6 |
| PLM fine-tuning | | | | | | | | |
| few shot setting | 5.00e-4 | 20 | 2 | 1 | - | - | 4 | 0.6 |
| not few shot setting, low resource | 5.00e-4 | 15 | 8 | 1 | - | - | 4 | 0.6 |
| not few shot setting, not low resource | 5.00e-4 | 15 | 16 | 4 | - | - | 4 | 0.6 |

Table 2: Hyper-parameter settings for Sec.3.1. Grad acc is the abbreviation of gradient accumulation size, inner dim is the abbreviation of inner dimension.

| language | PLF | MPE_adapter | MPE_prefix | MPF | MPS_adapter | MPS_prefix |
|---|---|---|---|---|---|---|
| amharic | 17.46/6.64/16 | 16.58/5.88/14.88 | 15.33/5.42/13.8 | 20.08/7.41/18.05 | 20.6/7.61/18.5 | 20.33/7.48/18.34 |
| arabic | 34.82/15.13/29.22 | 32.29/12.85/26.43 | 29.02/10.94/24.14 | 34.89/14.76/29.15 | 35.16/14.96/29.27 | 35.36/15.11/29.49 |
| azerbaijani | 16.93/7.04/15.36 | 17.81/7.34/16.13 | 15.72/6.3/14.47 | 21.4/9.55/19.37 | 21.77/9.85/19.62 | 21.58/9.83/19.76 |
| bengali | 25.9/9.73/22.12 | 25.99/10.07/22.21 | 23.76/9.11/20.66 | 29.46/12.02/25.1 | 29.11/11.61/24.64 | 29.57/11.88/24.89 |
| burmese | 14.35/4.51/13.08 | 14.07/4.09/12.54 | 12.74/3.46/11.51 | 16.17/5.17/14.42 | 15.84/4.7/14.08 | 16.12/5.07/14.39 |
| chinese simplified | 40.87/25.97/34.09 | 39.81/24.98/33 | 36.3/22.34/30.22 | 43.8/28.76/36.9 | 44.95/29.66/37.79 | 44.37/29.1/37.18 |
| chinese traditional | 40.04/25.11/33.31 | 39.16/24.17/32.3 | 35.99/21.79/29.7 | 43.21/28.03/36.17 | 44.26/28.75/37.01 | 43.83/28.33/36.66 |
| english | 40.67/18.04/32.72 | 38.96/16.52/31.06 | 34.6/12.93/27.38 | 37.61/15.15/29.88 | 38.29/15.71/30.41 | 38.29/15.63/30.39 |
| french | 32.02/13.47/25.29 | 31.85/13.34/25.13 | 30.63/12.97/24.49 | 35.33/16.19/28.2 | 36.06/16.22/28.4 | 35.69/16.15/28.16 |
| gujarati | 19.38/6.49/17.6 | 19.31/6.23/17.62 | 18.03/5.65/16.45 | 21.96/7.72/19.9 | 22.45/7.97/20.21 | 22.38/8.14/20.31 |
| hausa | 36.16/15.43/29.13 | 35.75/14.84/28.21 | 34.08/13.82/27.07 | 39.41/17.72/31.64 | 39.75/17.64/31.85 | 39.63/17.8/31.96 |
| hindi | 38.64/17.33/32.38 | 37.57/16.09/31.1 | 34.32/13.23/28.23 | 38.57/16.87/32.03 | 39.18/17.37/32.49 | 39.02/17.3/32.35 |
| igbo | 27.16/8.76/21.37 | 25.22/7.08/19.38 | 23.27/5.25/17.36 | 31.64/10.2/24.51 | 30.17/9.2/22.98 | 30.11/9.5/23.1 |
| indonesian | 35.69/16.23/29.92 | 34.86/15.51/28.85 | 30.96/12.87/25.72 | 37.01/17.02/30.75 | 37.83/17.55/31.37 | 37.69/17.57/31.46 |
| japanese | 45.86/22.01/35.39 | 45.6/21.87/35.02 | 41.23/18.89/32.42 | 48.08/23.8/37.32 | 48.65/24.26/37.33 | 48.41/24.16/37.39 |
| kirundi | 28.97/12.84/23.6 | 28.94/12.23/23.06 | 26.18/10.18/20.51 | 32/14.41/25.82 | 32.65/14.98/26.22 | 32.65/14.91/26.26 |
| korean | 19.04/9.42/18.06 | 19.92/9.93/18.59 | 18.07/8.95/17.08 | 23.7/11.49/22.34 | 23.57/11.31/21.78 | 23.11/11.27/21.6 |
| kyrgyz | 13.66/5.58/12.43 | 13.89/5.62/12.69 | 11.77/4.88/10.87 | 18.34/8.01/16.5 | 18.4/7.72/16.0 | 18.3/7.8/16.11 |
| marathi | 20.21/8.94/18.31 | 20.03/8.49/18.19 | 18.3/7.55/16.83 | 22.06/9.57/20.01 | 23.13/10.32/20.8 | 22.82/10.02/20.54 |
| nepali | 23.16/9.06/21.14 | 23.63/8.69/21.51 | 20.91/7.37/19.25 | 26.57/10.2/24.22 | 26.5/10.05/24.06 | 26.59/10.2/24.21 |
| oromo | 16.55/5.35/14.61 | 16.2/5.14/14.13 | 13.7/4.14/12 | 18.74/6.21/16.19 | 19.68/6.45/16.91 | 19.49/6.71/16.88 |
| pashto | 37.69/15.38/31.19 | 36.74/14.02/29.85 | 33.85/12.14/27.88 | 38.28/15.49/31.77 | 38.85/15.85/32.05 | 38.92/16.05/32.16 |
| persian | 37.08/16.78/30.44 | 35.72/15.34/28.81 | 32.98/12.89/26.27 | 35.71/15.06/29.1 | 37.25/16.58/30.42 | 37.32/16.43/30.3 |
| pidgin | 34.55/12.67/27.03 | 35.11/13.14/27.41 | 32.85/11.57/25.79 | 37.97/15.13/29.86 | 38.56/15.6/30.14 | 38.89/15.88/30.47 |
| portuguese | 37.04/16.25/28.82 | 36.3/15.18/27.7 | 32.99/12.57/25.07 | 37.15/15.89/28.53 | 37.71/16.33/28.97 | 37.59/16.21/28.91 |
| punjabi | 26.18/9.61/21.95 | 25.75/8.58/20.97 | 25.94/8.49/21.43 | 30.77/12.15/25.57 | 31.14/12.3/25.33 | 30.77/12.27/25.29 |
| russian | 32.18/13.83/26.11 | 30.56/12.67/24.73 | 26.5/9.79/21.44 | 32.21/13.64/26.16 | 32.82/13.97/26.44 | 32.61/13.92/26.39 |
| scottish gaelic | 23.16/7.04/19.25 | 20.42/4.31/17.02 | 24.05/7.73/19.45 | 29.01/10.96/22.87 | 28.85/10.55/22.65 | 30.08/11.2/23.83 |
| serbian cyrillic | 18.64/4.83/15.51 | 17.83/4.26/14.49 | 18.52/4.89/15.58 | 23.79/7.99/20.13 | 24.5/8.42/20.73 | 24.4/8.35/20.62 |
| serbian latin | 17.13/4.19/14.24 | 18.09/4.31/14.82 | 18.04/4.21/14.9 | 21.64/6.68/18.23 | 22.91/7.18/19.26 | 22.57/7.11/19.02 |
| sinhala | 23.78/11.87/21.27 | 23.29/11.16/19.81 | 21.86/10.48/19.06 | 21.51/8.07/18.9 | 27.7/13.8/23.61 | 27.43/13.41/23.7 |
| somali | 29.08/10.21/22.45 | 29.06/9.97/22.13 | 28.09/8.88/21.45 | 31.54/11.55/24.22 | 32.1/11.38/24.42 | 32.48/11.72/24.55 |
| spanish | 30.05/10.98/23.13 | 29.28/10.3/22.23 | 28.09/9.05/21.14 | 31.51/11.87/24.07 | 31.63/11.9/24.11 | 31.59/11.92/24.07 |
| swahili | 33.82/14.82/27.78 | 34.79/15.52/28.24 | 33.43/14.91/27.19 | 37.68/17.86/30.92 | 38.24/17.82/31.16 | 37.74/17.65/30.8 |
| tamil | 22.47/10.33/20.58 | 19.81/8.83/18.12 | 16.45/6.58/15 | 24.31/11.04/22.07 | 24.58/11.11/22.3 | 24.48/11.08/22.11 |
| telugu | 16.66/5.84/15.1 | 17.02/5.91/15.24 | 15.14/4.99/13.77 | 17.73/5.73/15.84 | 20.09/7.1/17.78 | 20.1/7.26/17.85 |
| thai | 35.17/15.18/26.85 | 36.08/14.06/25.7 | 33.47/14.12/25.7 | 36.43/16.28/28.22 | 37.84/17.34/28.81 | 37.99/17.65/29.07 |
| tigrinya | 22.3/7.19/19.04 | 21.37/6.14/17.89 | 19.97/5.65/16.51 | 25.26/7.99/21.1 | 25.85/8.51/21.6 | 25.48/8.55/21.78 |
| turkish | 31.41/15.07/28 | 30.53/14.04/26.97 | 26.12/11/23.29 | 32.92/15.57/29.28 | 33.63/16.17/29.95 | 33.58/16.1/29.81 |
| ukrainian | 23.58/10.2/20.59 | 21.84/8.92/19.08 | 18.73/7.07/16.43 | 23.99/10.14/20.92 | 24.73/10.7/21.58 | 24.75/10.62/21.57 |
| urdu | 40.19/19.34/33.6 | 38.81/17.69/32.08 | 35.05/14.5/28.76 | 39.49/18.33/32.83 | 40.04/18.71/33.15 | 40.12/18.71/33.21 |
| uzbek | 13.48/4.84/12.32 | 14.44/5.21/13.1 | 11.77/3.98/10.88 | 16.82/6.35/15.38 | 17.45/6.74/15.73 | 17.63/6.71/15.87 |
| vietnamese | 32.53/16.45/25.94 | 30.78/14.57/23.78 | 27.33/12.34/21.27 | 30.24/14.39/24.13 | 33.62/16.43/26.46 | 33.49/16.57/26.38 |
| welsh | 31.58/11.46/25.6 | 30.19/9.97/24.17 | 27.92/8.06/22.24 | 32.64/11.59/26.12 | 33.09/12.03/26.41 | 33.13/11.88/26.35 |
| yoruba | 29.06/10.96/23.3 | 29.94/10.43/23.31 | 26.77/8.84/20.85 | 31.62/11.66/25.06 | 31.95/11.92/25.24 | 31.71/11.61/24.91 |
| average | 28.14/11.96/23.45 | 27.58/11.23/22.66 | 25.35/9.84/20.92 | 30.23/12.93/25.11 | 30.96/13.39/25.56 | 30.89/13.4/25.57 |

Table 3: R1/R2/RL of six models on 45 languages and the average score of all languages.