One Step Is Enough for Few-Shot Cross-Lingual Transfer: Co-Training with Gradient Optimization

Anonymous ACL submission

Abstract

The current state-of-the-art for few-shot cross-lingual transfer learning first trains on abundant labeled data in the source language and then fine-tunes with a few examples on the target language, termed *target-adapting*. Though this has been demonstrated to work on a variety of tasks, in this paper we show some deficiencies of this approach and propose a one-step co-training method that trains on both source and target data with *stochastic gradient surgery*, a novel gradient-level optimization. Unlike the previous studies that focus on one language at a time when target-adapting, we use one model to handle *all target languages simultaneously* to avoid excessively language-specific models. Moreover, we discuss the unreality of utilizing large target development sets for model selection in previous literature, and further show that our method is *development-free* for target languages and also able to escape from overfitting issues. We conduct a large-scale experiment on 4 diverse NLP tasks across up to 48 languages. Our proposed method achieves state-of-the-art performance on all tasks and outperforms target-adapting by a large margin[1], especially for languages that are linguistically distant from the source language, e.g., an average of 7.36% absolute F1 improvement on the NER task, up to a gain of 17.60% on Punjabi.

1 Introduction

The cost of linguistic data annotation and a plethora of differences across language resources and structures of natural language processing (NLP) tasks result in the problem that sufficient labeled data is only for a handful of high-resource languages (Bender, 2011). The lack of data for low-resource languages leads to the need for effective cross-lingual transfer learning, which aims to leverage abundant labeled high-resource languages to learn low-resource ones. The majority of methods for cross-lingual transfer are mainly based on multilingual language models (LMs) (Devlin et al., 2019; Conneau et al., 2020; Xue et al., 2021) which are pre-trained on massive multilingual data. *Zero-shot cross-lingual transfer* is widely explored where a multilingual LM is trained on a large amount of labeled data in the source language without any target data, and then is directly evaluated on the target test set, frequently achieving surprisingly good performance (Wu and Dredze, 2019; Pires et al., 2019; Conneau et al., 2020). Recently, Lauscher et al. (2020) emphasize the effective mechanism of *few-shot cross-lingual transfer* for improving target-language performance, where only a few (such as 10) extra target examples can obtain substantial improvements. The current state-of-the-art methods for few-shot cross-lingual transfer learning (Lauscher et al., 2020; Hedderich et al., 2020; Maurya et al., 2021; Zhao et al., 2021) utilize the source-trained model (the same model training on the source data in zero-shot learning) to fine-tune on small target examples, which is termed *target-adapting*.

In this paper, we dissect the potential weaknesses of the ubiquitous target-adapting method and propose a one-step co-training method that trains on both source and target data with a novel gradient-level optimization, **stochastic gradient surgery**. Specifically, we highlight 6 benefits (contributions) of our method in this paper as follows:

(1) State-of-The-Art Performance: Our proposed method achieves significant improvements compared to target-adapting on 4 diverse NLP tasks across up to 48 languages. For instance, averaged over all target languages, we demonstrate an absolute F1 improvement of 7.36% on NER using 5-shot learning, with our best performance gains on Punjabi where the gap is 17.60% (Section 4).

(2) One Model for All Languages: The target-adapting step generally focuses on only one target language. With the proposed method, we do not

[1]Code is available at: https://github/REDACTED.

need to fine-tune specialized models for every target language, which is of particular interest when scaling to dozens or even hundreds of languages. We discuss the benefits of co-training examples of all target languages on one model to handle all languages in our method, even though their number of shots is extremely small (Section 3.2).

(3) Efficient Gradient De-Conflicting and Information De-Dilution: Two issues arise when co-training using data from all target languages in addition to the source language — conflicting gradients among languages and target information dilution. Stochastic gradient surgery efficiently de-conflicts gradients and de-dilutes the target information (Section 3.4 and 3.5).

(4) Single Language Friendly: Though our proposed method normally uses information from multiple target languages, in the simplest setting, where we only have a single target language, stochastic gradient surgery co-trained on source and target still substantially outperforms standard target-adapting. The improvement is especially pronounced for languages linguistically distant from the source language (Section 5.3).

(5) The Same Script Helps: For a specific language, the model is able to use information learned from other languages. In Section 5.4, we show that this gain is most pronounced in languages that use the same script.

(6) Development-Free for Target Languages: Target development (dev) set used by previous studies (Hsu et al., 2019; Zhao et al., 2021) significantly outnumber training examples in few-shot cross-lingual learning, which is not realistic in the true low-resource settings. However, target-adapting is easily to overfit on small examples without target dev sets. In comparison, our proposed method is development-free for target languages and able to escape overfitting issues (Section 5.5).

2 Background and Related Works

2.1 Cross-Lingual Transfer Learning

Cross-lingual transfer learning enables us to co-learn the meaning of words across languages and facilitates model transfer between languages, particularly from high-resource to low-resource languages (Ruder et al., 2019). Language transfer is based on finding a shared cross-lingual space for source and target languages. One of the most common methods is to align the source and target embedding spaces, termed cross-lingual word embeddings (CLWEs) (Mikolov et al., 2013; Artetxe et al., 2016; Conneau et al., 2018a; Vulić et al., 2019). Recently, multilingual pre-trained encoders have shown stronger effectiveness over CLWEs of cross-lingual transfer in various tasks (Artetxe and Schwenk, 2019; Wu and Dredze, 2019), where some studies utilize static pre-trained encoders for transfer learning (Wang et al., 2019; Xu and Koehn, 2021), while more studies continuously train encoders for cross-lingual transfer (Conneau et al., 2020; Luo et al., 2021; Xue et al., 2021) based on the finding that source and target representations are still aligned after only fine-tuning on the source data (Hsu et al., 2019).

2.2 Few-Shot Learning

Few-shot learning was firstly investigated in computer vision (Fei-Fei et al., 2006). Currently, the majority of studies for NLP tasks are designed for one single language (usually English), e.g., model agnostic meta-learning (Finn et al., 2017) and prototypical networks (Snell et al., 2017). However, limited few-shot studies are explored in cross-lingual settings. Recent works mainly focus on zero-shot cross-lingual transfer to evaluate the cross-lingual generalization capabilities of multilingual representations, e.g., XTREME (Hu et al., 2020; Ruder et al., 2021) and XGLUE (Liang et al., 2020). Lauscher et al. (2020) further emphasize that additional fine-tuning on a few inexpensive labeled target-language instances is surprisingly effective across broad NLP tasks. Zhao et al. (2021) highlight the sensitivity to the selection of a few shots and suggest using the same shots for fair comparisons. State-of-the-art methods for few-shot cross-lingual learning follow the source-training + target-adapting paradigm. In this paper, we investigate deficiencies of this approach and propose more effective methods which significantly improve the transfer performance compared to target-adapting.

2.3 Gradient Surgery

Previous works on gradient optimization (Chen et al., 2018; Sener and Koltun, 2018; Yu et al., 2020) have successfully utilized gradient-level techniques to improve the performance of multi-task models. In fact, co-training multilingual data can be categorized into multi-task learning (Zhang and Yang, 2018) but in a monolithic manner by using a single language-agnostic objective on the concatenated data from all languages. Recently, multilingual machine translation utilizes gradient-level reg-

ularization to improve the translation performance (Wang et al., 2020; Yang et al., 2021; Wang et al., 2021b). In this paper, our experiments mainly focus on co-training multiple target languages, so we propose **stochastic gradient surgery** (Section 3.5) based on original gradient surgery (Yu et al., 2020) to improve the overall performance.

3 Methods

3.1 Ordinary Few-Shot Learning

The current state-of-the-art few-shot cross-lingual transfer learning method (Lauscher et al., 2020; Hedderich et al., 2020; Zhao et al., 2021) includes two stages, *source-training* and *target-adapting*. In the source-training stage, a pre-trained LM such as mBERT (Devlin et al., 2019) or XLM-R (Conneau et al., 2020) is fine-tuned with sufficient labeled data in the source language (which is usually English). In the target-adapting stage, the source-trained model is then fine-tuned only with a few examples in the target language. We abbreviate the name of this method to **ord-FS**.

3.2 Co-Fine-Tuning all Target Languages

The ord-FS method brings up a question: **is it necessary to fine-tune a language-specific model for each target language?** Can we use one model to handle all target languages to avoid excessively language-specific models? One straightforward method to have such a model is fine-tuning the source-trained model on concatenated examples of all target languages, instead of only one target. Here, we are interested in whether more few examples of other target languages will improve/degrade the overall performance. We abbreviate the name of this method to **co-FT**.

3.3 Co-Training Source and Target Languages

Ord-FS and co-FT follow the *transductive transfer*[2] learning method that first trains on the source domain and then fine-tunes on the target domain (Pan and Yang, 2009). However, recently, Xu et al. (2021) show that abruptly shifting the source domain to the target domain is not an optimized solution due to catastrophic forgetting (McCloskey and Cohen, 1989). Thus, we should be carefully about the language domain gaps between the source and target languages, especially for distant languages.

One naive but effective approach to preserve the source knowledge and escape catastrophic forgetting is simply co-training both the source and target data[3] (all target languages), where we simplify source-training and target-adapting into only one co-training step. We abbreviate the name of this method to **naive-co-train**.

3.4 Gradient Surgery in Co-Training

One issue of naive-co-train is conflicting gradients (Yu et al., 2020) among languages, which makes training more difficult because gradients point away from one another. We define that two gradients are conflicting if they have a negative cosine similarity. Another issue is that the information of the target domain will be diluted due to the overwhelming source data. Specifically, the gradient of source data is much larger in magnitude than the other languages in one batch training due to the small or even no target training instances in this batch. Hence, the source gradients will dominate the average gradient and result in information dilution of the target data and underestimation of the target language performance.

The main idea of using gradient surgery (Yu et al., 2020) to mitigate two issues above is, in each backpropagation step, projecting the dominant gradient to the normal plane of a target gradient to de-conflict their gradients and '*remind*' the model of target instances. Specifically, we denote g_s as the gradient for the source language and g_t as the gradient for the target language. We first compute the cosine similarity between g_s and g_t and judge g_s and g_t are conflicting gradients if their similarity is negative. Next, we project g_s into the normal plane of g_t only if they are conflicting:

$$g_s' = g_s - \frac{g_s \cdot g_t}{\| g_t \|^2} g_t \quad (1)$$

The modified g_s' replace the original dominant source gradient to update the model parameters.

3.5 Stochastic Gradient Surgery

However, target data is usually not guaranteed to exist in the batch due to the small training size. Even though we assume that we have target data for all target languages in each batch training, we should detect conflicting gradients not just between source and target languages, but also between every target language. However, this is extremely computationally expensive, especially when it comes

[2]The pre-training (source-training) and the fine-tuning (target-adapting) are the same task.

[3]Target data is randomly interpolated in the source data.

to large-scale languages for training. Based on this, we propose **stochastic gradient surgery** approach, composed of two parts, **oracle dataset creation** and **stochastic training**.

Oracle Dataset Creation In the case of K-shot learning, the oracle dataset comprises K training instances[4] for each target language. To not use any external information, the oracle datasets of target languages are the same as their training examples but only used for gradient surgery. Similar to Wang et al. (2020, 2021a); Yang et al. (2021), we create an oracle dataset to ensure that we can pair any one of the target languages with the source language to operate gradient surgery.

Stochastic Training In each batch training, we randomly pick oracle data of a random target language in a uniform distribution to conduct gradient surgery with the source batch data. Moreover, in order to avoid that small number of target examples constrain the source gradients into a sub-optimal place (especially for tasks which need higher-level semantic understanding), we also have a pre-set threshold α to control the probability of gradient surgery in each training step. The gradient surgery is conducted only if a sampled value $p \sim \text{uniform}[0, 1]$ is smaller than α.

The advantages of this method are that 1) we only focus on gradient de-conflicting between the source and one of the target languages, which only computes the gradient one additional time to avoid expensive computation, 2) and more importantly, the source language could be a pivot language which also helps gradients of target languages deconflict between each other (more discussion in Section 5.2). The detailed workflow is shown in Algorithm 1. We abbreviate the name of this method to **gradient-co-train**.

Algorithm 1: Stochastic Gradient Surgery

Input : Language Set $\mathcal{L}$; Pre-Trained Model θ; Co-Training Data $\mathcal{D}_{\text{train}}$; Oracle Data $\mathcal{D}^l_{\text{oracle}}, l \in \mathcal{L}$; Pre-Set Threshold α.

Initialize $\theta_0 = \theta$, step $t = 0$
while *not converged* **do**
  ▷ *Iterate batches $\mathcal{B}_{train}$ from data $\mathcal{D}_{train}$*
  for $\mathcal{B}_{train}$ *in* $\mathcal{D}_{train}$ **do**
    $g_{\text{train}} = \nabla_{\theta_t} L(\theta_t, \mathcal{B}_{\text{train}})$
    Sample a language l from set $\mathcal{L}$
    $g_{\text{oracle}} = \nabla_{\theta_t} L(\theta_t, \mathcal{D}^l_{\text{oracle}})$
    Sample a value $p \sim \text{uniform}[0, 1]$
    ▷ *Gradient surgery*
    if $g_{oracle} \cdot g_{train} < 0$ *and* $p < \alpha$ **then**
      $g_{\text{train}} = g_{\text{train}} - \frac{g_{\text{train}} \cdot g_{\text{oracle}}}{\|g_{\text{oracle}}\|^2} g_{\text{oracle}}$
    end
    Update $t \leftarrow t + 1$
    Update θ_t with gradient g_{train}
  end
end

4 Experiments

4.1 Development-Free Training

Importantly, Zhao et al. (2021) notice that few-shot learning easily tends to overfit quickly at a small number of shots, where the model performs best on the dev set at the beginning of training. One good solution to avoid overfitting is using target dev set for early stopping. Previous studies (Hedderich et al., 2020; Zhao et al., 2021) utilize a large amount of dev sets for each target language for model selection, e.g., even around 10K dev examples for Arabic in the NER task. However, it is unlikely that such a dev set would be available in reality, especially for the extreme low-resource training such as 1-shot and 5-shot learning, since it would be more effective to use it for training instead (Kann et al., 2019). The true standard setup of zero-shot cross-lingual learning only uses the source dev set (Zhao et al., 2021), and few-shot learning should also follow this setup, particularly at a small value of shots. Thus, we suggest **only using the source dev set for model selection**. However, target-adapting is appropriate to use the source dev for model selection due to different languages in the training and dev steps. Hence, the two-step methods, ord-FS and co-FT, use the last checkpoint for evaluation. Since naive-co-train and gradient-co-train train on both source and target data, they are suitable for using the source dev set for target model selection. We show that **our methods substantially outperform target-adapting whatever it uses unrealistic dev sets or not** in Section 4.4.

We consider all introduced methods in the experiment, including two-step methods — ord-FS, co-FT, and one-step methods — naive-co-train and gradient-co-train. Moreover, in order to investigate the difference between using and not using dev sets, we add another baseline, **ord-FS+dev**, ord-FS with

[4] XNLI use K examples from every class followed by the "N-way K-shot" discussion in Section 4.3.

unrealistically large dev sets[5] for model selection as Zhao et al. (2021) conduct.

4.2 Tasks and Datasets

We consider two lower-level (structured prediction) tasks, Wikiann Named-Entity Recognition (NER) task (Pan et al., 2017) and Part-of-Speech Tagging (POS) (Nivre et al., 2018) and two different types of higher-level tasks, Typologically Diverse Question Answering-Gold Passage[6] (TyDiQA-GoldP) (Clark et al., 2020) and Cross-lingual Natural Language Inference (XNLI) (Conneau et al., 2018b). We download datasets from the XTREME-R benchmark (Hu et al., 2020; Ruder et al., 2021). NER and POS cover 48 and 38 languages, respectively. Our experiments use 35 languages on POS because the remaining three languages, Thai(*th*), Tagalog(*tl*) and Yoruba(*yo*), do not have target training data in XTREME-R. TydiQA and XNLI cover 9 and 15 languages, respectively. We conduct aforementioned methods on all tasks for all languages. English is the source language and the others are targets. Statistics about languages are listed in Appendix B.

4.3 Settings

Two-step training methods, ord-FS(+dev) and co-FT, have two different settings for source-training and target-adapting. For one-step methods, naive-co-train and gradient-co-train, their settings are the same as source-training in the two-step methods. We run 10 epochs for NER and POS, 60 for TyDiQA, and 10 for XNLI in both source-training and target-adapting. The batch size of all tasks is 32 for source-training and K for target-adapting with a 2e-5 learning rate. Pre-set threshold α is 1 for NER and POS and 0.1 for TyDiQA and XNLI unless otherwise noted. The values of α are empirically selected, which might not be optimal but strongly effective. The model architecture of NER and POS is based on pre-trained XLM-$\text{R}_{\texttt{large}}$ attached with a feed-forward token-level classifier. For TydiQA, the representations of all subwords in XLM-$\text{R}_{\texttt{base}}$ are input to a span classification head —- a linear layer computing the start and the end of the answer. For XNLI, the model architecture is XLM-$\text{R}_{\texttt{base}}$ with a simple softmax classifier on the vector of the start token. The number of examples we consider is $K \in \{1, 5, 10\}$. The sampling method is simply extracting random K shots. The only exception is XNLI, where we adopt the sampling method of conventional few-shot classification learning — "N-way K-shot" (Fei-Fei et al., 2006) — we sample K examples for N classes. Here, N is the total number of classes in XNLI. We repeat every experiment 5 times with 5 different random seeds[7] suggested by Lauscher et al. (2020). All methods use the same K shots for a fair comparison. We finally report the average accuracy (XNLI) or F1 scores (other tasks) and their standard deviation.

4.4 Results

The main results on each task, conditioned on the number of examples K and **averaged across all languages**, are presented in Table 1. The full results with each target language are shown in Appendix C. For all values of K and all tasks, **gradient-co-train performs the best among all introduced few-shot learning methods**.

The zero-shot cross-lingual transfer results ($K = 0$) deliver similar results comparable to Ruder et al. (2021). Similar to the findings in Lauscher et al. (2020); Zhao et al. (2021), we notice substantial improvements with ord-FS(+dev) on lower-level tasks (NER and POS) and modest improvement on XNLI over zero-shot performance.

However, ord-FS significantly degrades the zero-shot performance on TyDiQA because it suffers from a tendency of overfitting on target training instances (more discussion in Section 5.5). On the other hand, with the help of dev sets in model selection, ord-FS+dev achieves higher performance than ord-FS on all tasks and particularly solve the overfitting issue.

Compared to ord-FS, NER and TyDiQA benefit most from co-FT, e.g., from 65.91% to 70.60% with $K = 5$ in NER. However, it still suffers from the overfitting issue but the impact decrease with more target examples. Co-training source sentences with target data (naive-co-train) seems a better solution. It consistently outperforms co-FT on all tasks with various K, and importantly, overcomes the serious overfitting on the TyDiQA task and highly boosts the performance (e.g., from 48.73% of co-FT to 57.03% of naive-co-train in 1-shot learning). Furthermore, applying stochastic gradient surgery on co-training (gradient-co-train) further achieves the best performance on all tasks

[5] Detail information of dev sets are shown in Appendix A

[6] We try to not use translated data such as XQuAD (Artetxe et al., 2020) to avoid unrealistic artifacts such as preserving source words (Clark et al., 2020).

[7] Shots are different with different seeds.

K	Methods	NER		POS		TyDiQA		XNLI	
		Avg. F1 (%)	sd.	Avg. F1 (%)	sd.	Avg. F1 (%)	sd.	Avg. Acc. (%)	sd.
$K = 0$	Zero-Shot	64.56	-	77.32	-	55.80	-	73.55	-
$K = 1$	ord-FS+dev	65.92	0.84	80.37	0.16	55.81	1.01	73.95	0.19
	ord-FS	64.11	0.98	80.24	0.19	47.44	1.47	73.70	0.17
	co-FT (Ours)	65.71	0.90	79.37	0.12	48.73	2.15	73.54	0.61
	naive-co-train (Ours)	67.31	0.58	80.04	0.23	57.03	0.56	73.29	0.43
	gradient-co-train (Ours)	**69.58**	0.99	**81.14**	0.27	**57.64**	1.02	**74.09**	0.54
$K = 5$	ord-FS+dev	68.22	0.69	83.15	0.23	55.60	1.07	74.08	0.36
	ord-FS	65.91	0.91	82.95	0.20	51.19	1.29	73.73	0.60
	co-FT (Ours)	70.60	0.85	81.95	0.16	54.49	1.76	73.13	0.74
	naive-co-train (Ours)	72.06	0.68	82.79	0.19	58.59	1.45	73.69	0.80
	gradient-co-train (Ours)	**73.27**	0.60	**83.48**	0.24	**59.34**	1.04	**74.41**	0.26
$K = 10$	ord-FS+dev	69.85	0.60	84.92	0.07	55.59	1.62	74.19	0.39
	ord-FS	68.75	0.67	84.66	0.08	53.17	1.56	74.03	0.38
	co-FT (Ours)	73.89	0.56	83.54	0.07	55.54	1.05	73.62	0.98
	naive-co-train (Ours)	74.13	0.45	84.52	0.17	58.88	1.37	74.23	0.37
	gradient-co-train (Ours)	**75.92**	0.61	**85.03**	0.16	**59.47**	1.73	**74.44**	0.38

Table 1: Main results of all methods with their standard deviation (sd.) of 5 repetitive experiments for all tasks with $K \in 1, 5, 10$. Scores are averaged by all target languages. Best scores are **bold**. Cells are colored by performance difference over zero-shot baseline: +3 or more, +0 to +3, -0 to -3, -3 or more. **ord-FS+dev**: ordinary few-shot learning that fine-tunes on one target language each time with development set; **ord-FS**: the **ord-FS+dev** method without development set; **co-FT**: co-fine-tuning concatenated target examples together; **naive-co-train**: naively co-training both source and all target examples together; **gradient-co-train**: utilizing stochastic gradient surgery during the naive-co-train.

NER		POS		TyDiQA		XNLI	
lang.	Δ F1 (%)	lang.	Δ F1 (%)	lang.	Δ F1 (%)	lang.	Δ Acc. (%)
pa	17.60	wo	3.82	bn	12.27	sw	2.36
zh	15.24	mr	3.51	te	11.14	ur	1.95
ar	14.14	hi	2.60	sw	10.58	ru	1.68
vi	13.22	tr	2.18	ar	9.45	fr	0.91
hi	12.68	fi	1.55	fi	9.05	zh	0.78

Table 2: Top-5 languages that achieve the highest improvement by using gradient-co-train methods compared to ord-FS on all tasks in 5-shot learning. Most languages are distant from English.

with all settings of K and outperforms ord-FS by a significant margin, such as up to 7.36% averaged absolute improvement on NER in 5-shot learning. On the other hand, the gap between our methods and ord-FS in POS is smaller than in NER (the same type of task). The reason could be that the POS task has already left less room for further improvement.

5 Analysis and Discussion

5.1 Which Language Benefits Most?

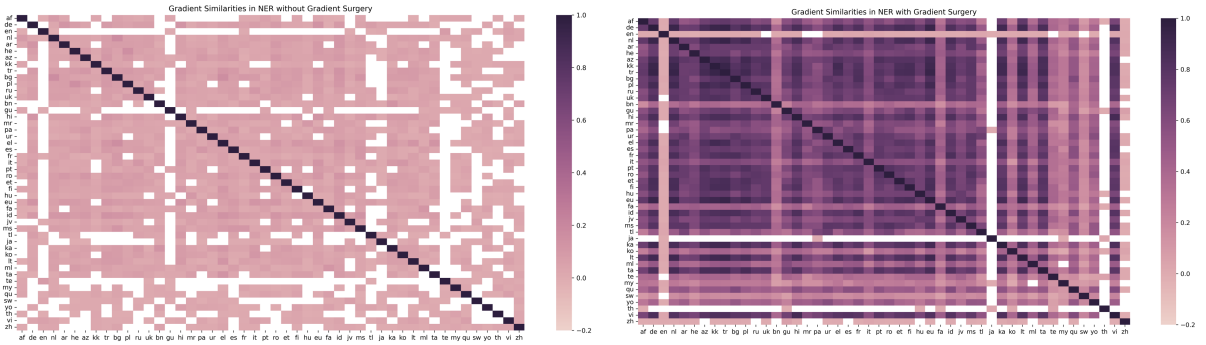
Table 1 shows the strong effectiveness of gradient-co-train in improving the overall performance of each task. Here, we are interested in taking a closer look at the results of specific languages and investigating which language benefits most. Take 5-shot learning as an example. Table 2 illustrates the top-5 languages which boost most by using gradient-co-train over ord-FS in all tasks[8], where the improvement is up to 17.60% absolute F1 scores for *pa* in the NER task. Most of the languages in the top-5 list are linguistically distant from English. We hypothesize that for such distant languages, the model has difficulty in learning the target training instances by abruptly shifting to the target domain. And for closely related languages, the model is able to extrapolate the target-specific knowledge whose priors are close to English so that the model is less sensitive to these few target training examples than distant languages. However, gradient-co-train is able to smoothly learn the distribution of source domain and extrapolate (distant) target domains by co-training and gradient-level optimization.

5.2 Visualization of Gradient De-Conflicting

We take the NER task as an example to analyze the gradient de-conflicting of stochastic gradient surgery since it covers the most languages among all tasks. In Figure 1, we use a symmetric heatmap to visualize pair-wise gradient similarities, averaged by all 5 checkpoints in 5-shot learning. Note that languages in the figure are adjacent to other languages in the same linguistic language family. The gradient of English is calculated by the randomly picked 100 batches on average, and gradients of the other target languages are calculated by their 5 training instances. To highlight the conflicting gradients across languages, we directly mark the cells with negative similarities as pure white color.

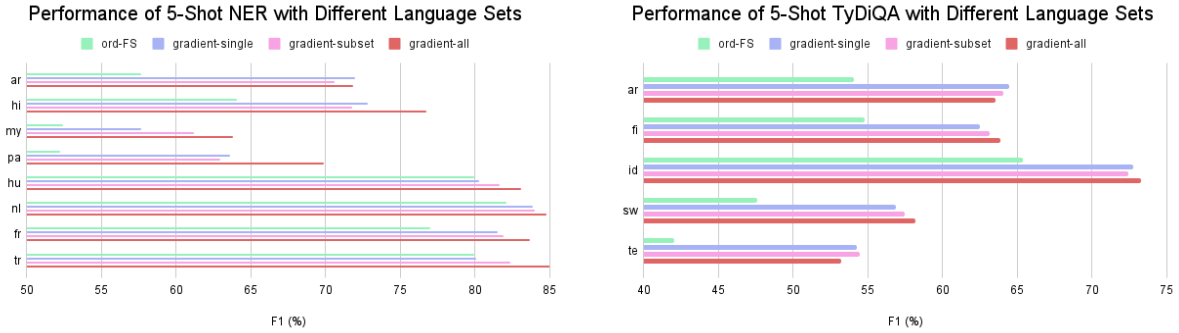
Figure 1a shows the gradient similarities of the naive-co-train model. As expected, gradient sim-

[8]For the languages that benefit the least, gradient-co-train still yields large gains over the baseline on NER and TyDiQA. We discuss this further in Appendix D.



(a) Gradient similarity across languages without gradient surgery (b) Gradient similarity across languages with gradient surgery

Figure 1: Gradient similarities across 48 languages in the NER task with 5 shots. Deeper colors represent higher cosine similarities. Conflicting gradients are directly marked as while cells in the heatmap. The similarities are highly improved after stochastic gradient surgery. Better view in color.



(a) Performance on various subsets of languages in NER (b) Performance on various subsets of languages in TyDiQA

Figure 2: Performance of gradient-co-train on different sets of languages compared to ord-FS for **(a)** NER and **(b)** TyDiQA. **gradient-{all,subset,single}** represents training on all/subset/single languages by using graident-co-train.

ilarities of many language pairs are conflicting (white color cells), and gradients of most languages are approximately orthogonal, where their similarities are close to 0. It is worth mentioning that gradients similarities between English and most languages are conflicting. In comparison, in Figure 1b, we illustrates the gradient similarities of gradient-co-train, and the gradient similarities between English and most of the target languages are positive. Moreover, gradients of most target language pairs have higher similarities (deeper colors), which also verifies the correctness of our statement that target languages utilize English as a pivot language to de-conflict and even improve their similarities. The only two exceptions are *th* and *ja*, the two hardest task in NER, whose F1 in zero-shot learning is only 1.02% and 18.31%. Their similarities with other languages are negative but positive between themselves. However, gradient-co-train still achieve impressive improvement on *th* ($\Delta = 3.13\%$) and *ja* ($\Delta = 5.40\%$) compared to naive-co-train (see the full results in Appendix C).

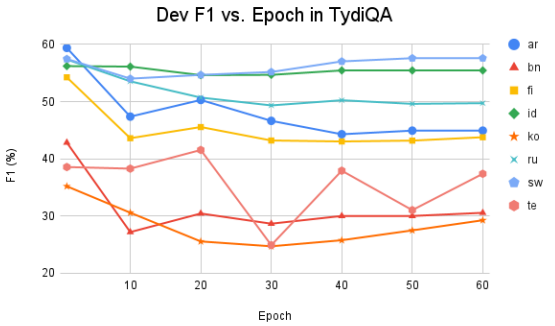


Figure 3: Dev F1 scores of ord-FS+dev in TyDiQA. 6 out of 8 target languages overfit quickly, where they achieve the best performance at the first epoch.

5.3 Co-Training with One Single Language

In some cases, people are only interested in one target language and do not have resources for other languages. Hence, we further explore the effectiveness of gradient-co-train in one target language

case. We conduct experiments on the NER and TyDiQA tasks that show larger gaps among different methods than other two tasks. Considering the high expense of training the source data from scratch for every target language, we run experiments on subsets of languages for each task. For the NER task, we test on 8 languages: *ar*, *hi*, *my*, *pa*, which are distant from English, *hu*, *nl*, *fr*, *tr*, which are similar to English. Figure 2a shows the results for NER. Gradient-co-train with only one single language is labeled as **gradient-single**[9] in the figure (blue, the second bar). We can focus on comparing ord-FS (green, the first bar). We notice that gradient-single still outperforms ord-FS by a large margin for 4 distant languages (e.g., 14.29% improvement for *ar*). In comparison, their gap becomes smaller when it comes to 4 languages related to English (e.g., 1.78% improvement for *nl*). Numeric results are shown in Appendix E. For the TyDiQA task, We pick 5 languages: *ar*, *fi*, *id*, *sw*, *te*. We still note that gradient-single highly boost the performance compared to ord-FS.

5.4 Do the Same Scripts Help?

Continuing the previous discussions in Section 5.3, we add a new baseline, **gradient-all** (red, the last bar in Figure 2), which uses gradient-co-train method with all languages (original settings). Interestingly, gradient-all outperforms gradient-single on all selected languages except for *ar* in NER, and a similar phenomenon also happens in TyDiQA. Note that *ar* is the only language that uses Arabic script in TyDiQA and only shares the same script with *yo* and *kk* among 48 languages in NER. It brings a question that **do small examples of other languages which use the same scripts help in few-shot learning?** Hence, we move our experiments further on using gradient-co-train with subsets of languages. We still consider the languages used in Section 5.3. Note that these languages are carefully selected. In NER, only *my* and *pa* share the same script (Brahmic) among 4 distant languages, and *hu*, *nl*, *fr*, *tr* share the Latin script from different language families. In TyDiQA, only *fi*,*id* and *sw* use the same script (Latin). We co-train 4 similar languages and 4 distant languages in NER, respectively. For TyDiQA, we co-train all 5 languages. The results of co-training subset of languages is denoted as **gradient-subset**[10] (pink, the third bar) in Figure 2. As expected, gradient-subset achieves better performance than gradient-single on all similar languages and on *my* among distant languages in the NER task. As for other languages using distinct scripts, their performance slightly degenerates compared to gradient-single. A similar discussion also holds for the high-level TyDiQA task, but gaps between gradient-single and gradient-subset are smaller. In conclusion, to pursue the best performance, we recommend using gradient-co-train with languages that share the same script or only a single language that uses a distinct script.

5.5 Escaping from Overfitting

The overfitting causes the significant degeneration of ord-FS performance in TydiQA. Figure 3 shows that 6 out of 8 target languages achieve the best dev score at the first epoch and decrease significantly afterwards. However, the phenomenon of degeneration is imperceptible in other tasks because only a few languages hit the same overfitting issue, e.g., 6.38% languages achieve the best score at the first epoch in 1-shot learning for NER, and even none of them has the issue in 10-shot learning. Different from two-step methods, one of the biggest benefits of gradient-co-train is the perfect fit for only using the source dev set to avoid overfitting (for model selection) because training and dev steps use the same (dominant) language. Thus, although gradient-co-train can also be further improved by using unrealistic target dev sets, the gaps are smaller compared to ord-FS (Appendix F).

6 Conclusion

We study the deficiencies of target-adapting in few-shot cross-lingual transfer and propose a co-training method with gradient-level optimization. Our best model achieves state-of-the-art on four diverse NLP tasks with all values of K. Moreover, we are the first to use a single model to co-train all target languages and find that languages can benefit from others that share the same scripts. We also show the effectiveness of our method compared to target-adapting in a single target language case, and the gaps are still significant. Finally, we propose only using source dev set in few-shot settings and show that our method is development-free for targets and also able to escape from overfitting issues.

[9] We reduce α for NER to 0.1 due to only one language considered.

[10] α is 0.4 for NER to ensure that each language has the same chance of explosion as gradient-single during training.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas. Association for Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the Cross-lingual Transferability of Monolingual Representations. In *Proceedings of ACL 2020*.

Mikel Artetxe and Holger Schwenk. 2019. Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Emily M Bender. 2011. On achieving and evaluating language-independence in nlp. *Linguistic Issues in Language Technology*, 6(3):1–26.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages. In *Transactions of the Association of 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.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018a. Word translation without parallel data. In *International Conference on Learning Representations*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018b. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of EMNLP 2018*, pages 2475–2485.

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.

Li Fei-Fei, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Michael A. Hedderich, David Adelani, Dawei Zhu, Jesujoba Alabi, Udia Markus, and Dietrich Klakow. 2020. Transfer learning and distant supervision for multilingual transformer models: A study on African languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2580–2591, Online. Association for Computational Linguistics.

Tsung-Yuan Hsu, Chi-Liang Liu, and Hung-yi Lee. 2019. Zero-shot reading comprehension by cross-lingual transfer learning with multi-lingual language representation model. 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 5933–5940, Hong Kong, China. Association for Computational Linguistics.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *CoRR*, abs/2003.11080.

Katharina Kann, Kyunghyun Cho, and Samuel R. Bowman. 2019. Towards realistic practices in low-resource natural language processing: The development set. 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 3342–3349, Hong Kong, China. Association for Computational Linguistics.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, Xiaodong Fan, Ruofei Zhang, Rahul Agrawal, Edward Cui, Sining Wei, Taroon Bharti, Ying Qiao, Jiun-Hung Chen, Winnie Wu, Shuguang Liu, Fan Yang, Daniel Campos, Rangan Majumder, and Ming Zhou. 2020. XGLUE: A new

benchmark datasetfor cross-lingual pre-training, understanding and generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6008–6018, Online. Association for Computational Linguistics.

Fuli Luo, Wei Wang, Jiahao Liu, Yijia Liu, Bin Bi, Songfang Huang, Fei Huang, and Luo Si. 2021. VECO: Variable and flexible cross-lingual pre-training for language understanding and 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 3980–3994, Online. Association for Computational Linguistics.

Kaushal Kumar Maurya, Maunendra Sankar Desarkar, Yoshinobu Kano, and Kumari Deepshikha. 2021. ZmBART: An unsupervised cross-lingual transfer framework for language generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2804–2818, Online. Association for Computational Linguistics.

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.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, et al. 2018. Universal dependencies 2.2.

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of ACL 2017*, pages 1946–1958.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.

Sebastian Ruder, Noah Constant, Jan Botha, Aditya Siddhant, Orhan Firat, Jinlan Fu, Pengfei Liu, Junjie Hu, Dan Garrette, Graham Neubig, and Melvin Johnson. 2021. XTREME-R: Towards more challenging and nuanced multilingual evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10215–10245, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. In *NeurIPS*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, 30:4077–4087.

Ivan Vulić, Goran Glavaš, Roi Reichart, and Anna Korhonen. 2019. Do we really need fully unsupervised cross-lingual embeddings? 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 4407–4418, Hong Kong, China. Association for Computational Linguistics.

Xinyi Wang, Ankur Bapna, Melvin Johnson, and Orhan Firat. 2021a. Gradient-guided loss masking for neural machine translation. *arXiv: Computation and Language*.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020. Balancing training for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8526–8537, Online. Association for Computational Linguistics.

Yuxuan Wang, Wanxiang Che, Jiang Guo, Yijia Liu, and Ting Liu. 2019. Cross-lingual BERT transformation for zero-shot dependency parsing. 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 5721–5727, Hong Kong, China. Association for Computational Linguistics.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. 2021b. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. 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 833–844, Hong Kong, China. Association for Computational Linguistics.

Haoran Xu, Seth Ebner, Mahsa Yarmohammadi, Aaron Steven White, Benjamin Van Durme, and Kenton Murray. 2021. Gradual fine-tuning for low-resource domain adaptation. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 214–221, Kyiv, Ukraine. Association for Computational Linguistics.

Haoran Xu and Philipp Koehn. 2021. Zero-shot cross-lingual dependency parsing through contextual embedding transformation. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 204–213, Kyiv, Ukraine. Association for Computational Linguistics.

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.

Yilin Yang, Akiko Eriguchi, Alexandre Muzio, Prasad Tadepalli, Stefan Lee, and Hany Hassan. 2021. Improving multilingual translation by representation and gradient regularization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7266–7279, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33.

Yu Zhang and Qiang Yang. 2018. An overview of multi-task learning. *National Science Review*, 5:30–43.

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, Online. Association for Computational Linguistics.

A Size of Dev Sets

In Figure 4, we show the size of dev sets that we used in our experiments, which are also the dev sets used by Zhao et al. (2021). Data of all tasks are downloaded from the XTREME-R benchmark (Hu et al., 2020; Ruder et al., 2021), where train/dev/test sets are already split. We can notice that the dev size of all languages in all tasks are tremendously higher than the largest number (10) of shots we pick in few-shot cross-lingual learning. However, in reality, if we only have access to a few training instance, we usually do not have a such large dev set. For tasks such as NER, POS and XNLI, we sample shots from the target training sets and directly use their supported dev sets. For TyDiQA which only supports train and dev sets in XTREME-R, we sample shots from the target training sets but use the remaining training data as dev sets, and we use dev sets for test.

B Language Statistics

In this paper, we cover a total of 49 languages in our whole experiments, including NER, POS, TyDiQA, and XNLI tasks. The list of full names of languages is shown in Table 3, with their ISO 639-1 code, script, and language families. We checkmark under the column of the task in the Table if the language is involved in the task.

C Full Results

The full results of NER, POS, TyDiQA and XNLI are shown in Table 4, Table 5, Table 6 and Table 7, respectively. In each task, we report F1 scores (or accuracy) of all covered languages in 1-,5-, or 10 shot learning by using all introduced methods. Best score among methods in each language is **bold**.

D Languages Benefits Least

In Table 8, we show the list of top-5 language which benefits least by using gradient-co-train in 5-shot learning. In NER and XNLI, we can notice a reverse phenomenon in the top-5 languages which benefit most — most of the languages are linguistically closer to English, at least using the same (Latin) script. In NER and TyDiQA tasks, although the gap left by gradient-co-train is much smaller than top-5 languages which benefits most, the improvements are still significant.

E Co-Training with Subsets of Languages

Here, we show the numeric results of Figure 2a and Figure 2b in Table 9 and Table 10, respectively.

F Our Methods with Dev Sets

We take *ar* in the NER task as an example to show that gradient-co-train can be further improved by utilizing large dev sets (around 10K). Figure 5 shows F1 scores of gradient-co-train and ord-FS both with and without dev sets with increasing epoch numbers. Methods with the help of the dev set start showing its effectiveness in model selection when it comes to large enough epoch numbers. Importantly, the gap led by the dev set in gradient-co-train is smaller than the one in ord-FS, which shows that gradient-co-train is able to select approximately optimal model even without target dev sets by using the source dev set. It is also worth mentioning that gradient-co-train even significantly outperforms the best performance of ord-FS with only 2 epoch of source (and target) data training. Still, ord-FS starts training based on 10-epoch source-trained model.

Langugae	ISO 639-1 code	Script	Language Family	NER	POS	TyDiQA	XNLI
Afrikaans	af	Latin	IE:Germanic	✓	✓		✓
Arabic	ar	Arabic	Afro-Asiatic	✓	✓	✓	
Azerbaijani	az	Latin	Turkic	✓			
Bulgarian	bg	Cyrillic	IE:Slavic	✓	✓		✓
Bengali	bn	Brahmic	IE:Indo-Aryan	✓		✓	
German	de	Latin	IE:Germanic	✓	✓		✓
Greek	el	Greek	IE:Greek	✓	✓		✓
English	en	Latin	IE:Germanic	✓	✓	✓	✓
Spanish	es	Latin	IE:Romance	✓	✓		✓
Estonian	et	Latin	Uralic	✓	✓		
Basque	eu	Latin	Basque	✓	✓		
Persian	fa	Perso-Arabic	IE:Iranian	✓	✓		
Finnish	fi	Latin	Uralic	✓	✓	✓	
French	fr	Latin	IE:Romance	✓	✓		✓
Gujarati	gu	Brahmic	IE:Indo-Aryan	✓			
Hebrew	he	Jewish	Afro-Asiatic	✓	✓		
Hindi	hi	Devanagari	IE:Indo-Aryan	✓	✓		✓
Hungarian	hu	Latin	Uralic	✓	✓		
Indonesian	id	Latin	Austronesian	✓	✓	✓	
Italian	it	Latin	IE:Romance	✓	✓		
Japanese	ja	Ideograms	Japonic	✓	✓		
Javanese	jv	Brahmic	Austronesian	✓			
Georgian	ka	Georgian	Kartvelian	✓			
Kazakh	kk	Arabic	Turkic	✓	✓		
Korean	ko	Hangul	Koreanic	✓	✓	✓	
Lithuanian	lt	Latin	IE:Baltic	✓	✓		
Malayalam	ml	Brahmic	Dravidian	✓			
Marathi	mr	Devanagari	IE:Indo-Aryan	✓	✓		
Malay	ms	Latin	Austronesian	✓			
Burmese	my	Brahmic	Sino-Tibetan	✓			
Dutch	nl	Latin	IE:Germanic	✓	✓		
Punjabi	pa	Brahmic	IE:Indo-Aryan	✓			
Polish	pl	Latin	IE:Slavic	✓	✓		
Portuguese	pt	Latin	IE:Romance	✓	✓		
CuscoQuechua	qu	Latin	Quechuan	✓			
Romanian	ro	Latin	IE:Romance	✓	✓		
Russian	ru	Cyrillic	IE:Slavic	✓	✓	✓	✓
Swahili	sw	Latin	Niger-Congo	✓		✓	✓
Tamil	ta	Brahmic	Dravidian	✓	✓		
Telugu	te	Brahmic	Dravidian	✓	✓	✓	
Thai	th	Brahmic	Kra-Dai	✓			✓
Tagalog	tl	Brahmic	Austronesian	✓			
Turkish	tr	Latin	Turkic	✓	✓		✓
Ukrainian	uk	Cyrillic	IE:Slavic	✓	✓		
Urdu	ur	Perso-Arabic	IE:Indo-Aryan	✓	✓		✓
Vietnamese	vi	Latin	Austro-Asiatic	✓	✓		✓
Wolof	wo	Latin	Niger-Congo		✓		
Yoruba	yo	Arabic	Niger-Congo	✓			
Mandarin	zh	Chinese	ideograms	✓	✓		✓

Table 3: Statistics about languages considered in this paper, including the scripts and language family of every language. A language used in a task is checkmarked under the column of the task.

K	Methods	ar	he	vi	id	jv	ms	tl	eu	ml	ta	te	af	nl	en	de	el	bn	hi	mr	ur	fa	fr	it	pt	es
$K=0$	Zero-Shot	45.75	55.35	78.67	52.47	61.35	69.65	71.95	56.37	65.79	55.82	52.85	78.34	83.76	84.50	78.78	78.38	74.39	69.71	61.87	54.85	56.82	79.78	81.39	81.91	76.64
	ord-FS+dev	51.62	55.86	78.10	55.51	63.32	69.18	72.27	59.12	65.27	57.92	53.39	78.53	83.43	**84.50**	78.72	78.91	74.15	70.84	63.66	61.52	65.64	79.17	81.63	81.81	76.82
	ord-FS	50.16	52.98	72.25	55.23	60.88	65.05	70.08	58.26	64.64	55.85	52.89	77.77	82.14	**84.50**	77.76	77.49	68.87	69.8	62.5	54.33	65.63	77.89	80.46	78.71	75.4
$K=1$	co-FT	51.24	57.53	77.55	51.46	61.81	64.44	70.2	62.64	67.17	59.58	57.46	**80.05**	83.33	**84.50**	79.21	78.46	73.12	72.0	64.64	56.95	63.56	80.23	80.80	81.61	77.51
	naive-co-train	54.59	58.21	77.06	58.38	63.38	69.92	73.94	64.31	66.27	61.48	57.47	78.27	83.78	84.40	78.84	79.25	77.01	72.43	66.41	67.16	72.43	80.90	**81.33**	82.37	80.14
	gradient-co-train	**61.75**	**60.39**	**79.41**	**60.49**	**65.86**	**71.00**	**74.98**	**67.32**	**69.45**	**63.69**	**61.08**	79.74	**84.12**	83.91	**79.75**	**80.69**	**78.82**	**74.32**	**67.9**	**72.77**	**77.33**	**82.39**	81.30	**83.37**	**82.20**
	ord-FS+dev	60.99	58.72	77.29	73.27	70.44	75.17	73.62	67.96	68.10	56.88	53.58	81.31	83.16	**84.50**	78.37	78.41	73.30	65.69	67.31	72.02	75.58	78.91	80.30	81.03	81.07
	ord-FS	57.69	58.18	68.01	72.43	68.12	73.99	68.57	67.54	65.51	56.05	52.08	79.26	82.11	**84.50**	75.87	74.11	68.42	64.08	66.31	69.54	75.45	76.99	72.38	77.95	78.36
$K=5$	co-FT	65.90	64.45	76.80	80.09	69.41	71.63	71.67	71.12	71.58	66.29	**63.55**	**82.30**	83.81	**84.50**	80.20	80.08	73.31	75.09	71.49	74.32	76.00	82.24	81.72	82.69	83.11
	naive-co-train	67.65	64.92	79.34	82.51	70.17	**75.84**	75.68	70.91	72.17	67.09	63.15	82.15	**84.95**	84.42	80.34	80.96	77.97	75.80	**74.13**	76.77	80.31	83.44	**82.30**	84.20	84.84
	gradient-co-train	**71.83**	**66.04**	**81.23**	**83.90**	**72.42**	75.51	**76.41**	**71.64**	**72.55**	**67.42**	63.42	81.99	84.77	83.98	**80.84**	**81.17**	**79.29**	**76.76**	73.76	**79.96**	**82.50**	**83.64**	82.04	**84.52**	**85.47**
	ord-FS+dev	64.33	61.77	76.13	78.96	71.28	77.80	72.29	71.66	69.27	57.31	58.84	81.54	82.60	84.50	79.39	79.08	74.60	70.97	66.89	78.05	80.47	79.32	81.07	81.40	80.76
	ord-FS	64.59	60.97	74.65	77.72	70.99	77.61	68.58	69.94	67.30	55.03	57.81	81.35	81.88	84.50	78.67	75.63	70.77	71.42	67.31	72.86	80.28	77.75	80.27	79.70	82.13
$K=10$	co-FT	71.84	66.90	79.75	85.57	73.82	**79.83**	74.90	73.73	74.59	70.69	65.88	**83.43**	85.02	84.50	81.38	81.30	78.22	77.29	76.38	79.25	82.25	82.76	82.97	84.77	85.68
	naive-co-train	74.96	67.46	81.15	85.21	73.61	76.39	76.41	74.74	74.42	69.22	65.55	82.71	84.82	**84.54**	80.70	81.81	79.61	77.71	75.17	80.18	84.23	83.86	82.75	84.75	85.30
	gradient-co-train	**75.48**	**69.17**	**82.01**	**86.89**	**77.93**	77.53	**77.87**	**77.35**	**76.58**	**72.33**	**66.69**	82.68	**85.42**	84.05	**81.86**	**82.72**	**80.90**	**78.93**	**77.55**	**83.87**	**84.32**	**83.91**	**83.51**	**85.35**	**86.40**

K	Methods	bg	ru	ja	ka	ko	th	sw	yo	my	zh	kk	tr	et	fi	hu	qu	pl	uk	az	lt	pa	gu	ro	Avg.
$K=0$	Zero-Shot	81.32	70.60	18.31	66.37	57.28	1.02	69.86	32.90	51.97	27.06	50.46	79.30	77.79	79.65	80.13	54.62	80.89	74.48	67.61	76.87	48.62	61.59	82.98	64.56
	ord-FS+dev	80.68	72.08	17.81	66.20	57.77	**3.46**	72.22	46.61	51.38	26.05	50.25	81.53	78.59	80.27	80.05	55.60	81.46	75.29	68.35	77.16	54.64	62.68	83.13	65.92
	ord-FS	79.22	68.02	14.92	64.82	54.94	2.13	72.07	45.24	49.72	20.68	49.60	81.51	76.79	79.48	79.08	56.06	81.15	71.14	67.59	76.28	53.96	60.03	81.16	64.11
$K=1$	co-FT	80.24	72.60	18.37	69.36	60.02	2.09	69.35	36.54	55.52	26.62	53.28	80.70	79.95	80.77	80.87	52.46	81.22	75.83	69.49	77.50	51.92	61.08	81.05	65.71
	naive-co-train	82.06	72.01	21.64	71.42	60.67	2.04	70.59	39.90	54.99	31.24	53.36	81.14	78.80	79.99	80.54	**56.13**	81.02	77.47	69.29	77.87	55.29	**61.95**	**81.89**	67.31
	gradient-co-train	**82.85**	**73.12**	**26.49**	**73.23**	**62.33**	2.81	**74.03**	**50.45**	**58.66**	**34.47**	**56.00**	**82.57**	**80.78**	**81.34**	**82.16**	54.53	**82.37**	**78.78**	**73.09**	**79.16**	**63.03**	61.79	81.62	**69.58**
	ord-FS+dev	80.49	72.83	19.34	69.44	58.08	3.59	75.25	56.54	58.57	25.20	58.83	81.56	80.05	80.82	80.75	52.71	82.53	77.52	68.75	76.87	54.93	59.56	83.37	68.22
	ord-FS	76.01	70.96	18.51	65.33	54.57	3.03	74.72	55.88	52.44	24.12	56.66	79.97	76.98	79.56	79.91	53.11	82.58	74.82	68.62	75.71	52.28	54.96	79.41	65.91
$K=5$	co-FT	82.69	73.77	21.58	73.08	65.44	3.83	74.02	53.59	59.82	30.89	**61.81**	83.57	80.85	82.15	82.42	58.27	82.34	78.17	72.04	79.55	60.23	62.71	82.81	70.60
	naive-co-train	84.34	74.12	25.05	73.91	64.09	4.64	74.68	57.02	59.00	35.55	61.40	84.09	80.86	81.95	82.62	61.85	82.63	**80.27**	72.37	80.09	65.26	**65.56**	**85.71**	72.06
	gradient-co-train	**84.36**	**74.91**	**30.45**	**73.99**	**65.82**	**7.77**	**77.48**	**60.97**	**63.78**	**39.37**	61.47	**85.21**	**82.16**	**82.88**	**83.06**	**62.66**	**83.48**	**80.27**	**73.78**	**81.43**	**69.89**	63.57	85.28	**73.27**
	ord-FS+dev	78.95	72.32	22.54	71.23	62.22	5.82	76.51	57.17	58.48	31.20	65.38	80.82	81.30	80.94	81.13	50.86	81.05	78.40	69.38	78.37	62.70	62.31	83.58	69.59
	ord-FS	77.27	68.68	22.70	70.97	60.73	4.89	78.16	59.85	57.31	30.01	64.89	78.57	80.14	80.49	79.04	51.5	80.25	76.07	69.59	77.89	60.36	59.63	81.46	68.75
$K=10$	co-FT	84.78	75.28	27.05	75.88	68.80	5.78	75.87	56.57	64.89	38.63	65.73	85.45	82.24	**84.23**	83.78	63.32	83.44	81.21	74.86	82.03	70.85	67.14	86.27	73.89
	naive-co-train	85.28	75.49	27.50	77.33	67.66	6.28	78.48	60.32	63.25	39.60	67.08	84.93	81.73	82.71	82.98	63.05	83.12	82.14	73.96	81.21	70.65	**69.77**	86.61	74.13
	gradient-co-train	**86.11**	**76.30**	**35.52**	**77.96**	**69.77**	**10.10**	**79.95**	**64.32**	**68.17**	**45.48**	**68.42**	**86.53**	**83.29**	84.15	**84.28**	**66.84**	**83.94**	**83.22**	**75.57**	**82.85**	**75.97**	67.29	**86.84**	**75.92**

Table 4: Full results (F1) of the NER task.

K	Methods	af	ar	bg	de	el	en	es	et	eu	fa	fi	fr	he	hi	hu	id	it	ja
$K=0$	Zero-Shot	89.39	69.52	88.65	88.50	86.40	96.12	89.18	86.74	73.20	74.49	86.22	87.79	68.91	75.50	83.75	83.32	89.63	27.82
	ord-FS+dev	89.76	73.91	89.62	**89.02**	86.55	**96.12**	90.04	87.02	76.72	**79.17**	86.51	88.60	77.14	80.28	**84.67**	83.45	90.73	64.19
	ord-FS	89.76	73.91	89.55	89.01	86.30	**96.12**	90.04	86.86	76.72	79.16	86.38	88.60	77.16	80.25	84.63	83.49	90.62	64.19
$K=1$	co-FT	**90.16**	71.58	89.33	88.81	86.46	**96.12**	89.85	86.90	75.72	76.14	86.57	88.63	72.06	77.97	83.81	82.88	89.99	52.06
	naive-co-train	90.14	72.17	89.16	88.73	86.79	96.10	89.65	87.69	76.50	76.55	86.86	88.16	72.93	79.56	83.70	83.55	90.08	58.95
	gradient-co-train	90.10	**75.40**	**90.58**	88.98	**86.83**	96.09	**90.37**	**88.34**	**77.76**	76.97	**87.41**	**89.41**	**74.24**	**82.05**	84.61	**84.22**	**90.98**	**64.58**
	ord-FS+dev	**91.15**	77.74	91.65	89.63	**90.24**	**96.12**	91.30	88.19	80.37	**81.47**	86.63	90.54	**83.02**	82.80	86.59	83.94	**92.54**	74.72
	ord-FS	91.11	77.60	91.75	**89.68**	90.15	**96.12**	91.31	88.07	80.02	81.37	86.54	90.61	82.72	82.75	**86.63**	83.96	92.48	**75.38**
$K=5$	co-FT	89.86	72.07	91.95	89.34	88.27	**96.12**	91.36	87.56	79.36	79.68	87.14	90.33	79.35	82.58	85.25	83.44	90.85	70.14
	naive-co-train	90.27	77.85	91.81	89.25	89,39	96,09	91.07	87.76	80.61	80.45	87.20	90.02	80.97	85.08	84.98	84.31	91.12	72.96
	gradient-co-train	90.65	**79.40**	**93.13**	89.58	89.97	96.07	**91.55**	**88.73**	**81.08**	81.05	**87.89**	**90.67**	81.36	**85.77**	85.61	**84.67**	91.98	73.53
	ord-FS+dev	92.72	80.16	93.03	**90.63**	**91.94**	**96.12**	91.95	89.12	81.84	**83.81**	87.21	91.77	**85.41**	84.62	**88.82**	85.20	**93.44**	**80.64**
	ord-FS	**92.85**	80.26	93.23	90.56	91.76	**96.12**	91.89	89.10	81.80	83.66	86.92	91.83	85.23	84.68	88.72	85.12	93.30	80.11
$K=10$	co-FT	90.67	79.62	93.12	89.55	90.23	**96.12**	91.90	88.48	82.35	83.17	87.91	91.10	84.15	87.05	86.57	84.73	92.39	78.85
	naive-co-train	91.28	80.67	93.24	89.58	90.96	96.07	91.90	88.48	82.35	83.17	87.91	91.10	84.15	87.05	86.57	84.73	92.39	78.85
	gradient-co-train	91.69	**81.36**	**93.94**	89.96	91.13	96.11	**92.03**	**89.27**	**82.76**	83.18	**88.47**	**91.72**	84.51	**87.29**	87.39	**85.42**	92.92	78.26

K	Methods	kk	ko	mr	nl	pt	ru	ta	te	tr	ur	vi	zh	lt	pl	uk	wo	ro	Avg.
$K=0$	Zero-Shot	78.97	54.11	83.51	89.67	89.69	89.71	77.81	86.78	75.45	67.45	58.83	39.67	84.40	85.01	85.69	28.89	85.58	77.32
	ord-FS+dev	78.81	54.16	82.99	**89.65**	90.19	89.93	77.72	86.54	75.86	73.50	**60.30**	60.76	84.73	85.18	86.14	36.85	86.16	80.37
	ord-FS	79.42	53.42	81.76	89.41	90.20	89.88	76.86	86.27	75.73	73.24	60.02	60.71	84.59	84.94	86.13	36.84	86.07	80.24
$K=1$	co-FT	79.78	55.04	83.97	89.56	89.85	90.10	77.37	86.08	76.13	70.46	58.50	59.48	84.72	84.85	86.37	34.76	85.89	79.37
	naive-co-train	79.72	54.68	83.90	89.62	90.29	90.10	77.17	85.56	76.18	74.15	59.17	61.71	85.07	85.61	86.27	38.06	86.36	80.04
	gradient-co-train	**80.43**	**55.26**	**83.74**	**89.65**	**90.59**	**78.70**	**86.85**	**76.62**	**77.67**	59.89	**63.52**	**85.49**	**86.75**	**87.32**	**40.92**	**87.15**	**81.14**	
	ord-FS+dev	**85.88**	55.33	83.94	89.81	90.85	91.12	78.44	85.12	75.97	82.00	68.23	**73.88**	85.49	85.98	87.04	44.40	87.97	83.15
	ord-FS	80.86	55.28	82.39	**89.90**	90.81	91.03	78.62	85.72	75.91	81.86	**68.24**	73.58	85.58	85.86	87.03	44.29	**88.07**	82.95
$K=5$	co-FT	81.27	55.92	85.46	89.43	90.75	90.71	77.66	85.76	77.04	76.94	60.85	69.91	85.31	86.98	87.45	40.07	87.22	81.95
	naive-co-train	81.81	55.95	84.65	89.75	90.98	90.89	78.45	85.62	77.40	81.83	62.91	72.95	85.85	87.06	87.27	45.76	87.57	82.79
	gradient-co-train	82.09	**56.34**	**85.12**	89.68	**91.25**	**91.32**	**80.07**	**86.26**	**77.50**	**83.93**	63.90	72.87	**86.47**	**87.99**	**88.21**	**48.09**	88.03	**83.48**
	ord-FS+dev	**91.13**	56.19	83.02	90.05	91.48	91.68	79.25	86.21	76.31	85.11	**72.46**	77.33	86.81	88.41	88.24	51.06	**89.02**	84.92
	ord-FS	82.21	56.46	83.39	**90.07**	91.40	91.48	79.49	86.25	76.03	85.22	72.38	**77.76**	86.83	88.44	88.10	51.44	**89.02**	84.66
$K=10$	co-FT	82.16	57.34	85.73	89.57	91.24	81.37	78.80	86.29	77.58	81.88	63.35	74.73	86.11	87.82	88.44	46.80	87.75	83.54
	naive-co-train	82.80	57.35	84.80	89.74	91.66	91.65	79.39	86.34	77.88	85.50	67.98	77.19	86.95	88.42	88.42	53.09	88.62	84.52
	gradient-co-train	83.23	**57.46**	**86.90**	89.83	**91.75**	**92.30**	**79.84**	**86.77**	**78.21**	**85.77**	69.94	77.05	**87.63**	**88.97**	**89.11**	**55.26**	88.70	**85.03**

Table 5: Full results (F1) of the POS task.

K	Methods	ar	bn	fi	id	ko	ru	sw	te	en	Avg.
$K=0$	Zero-Shot	62.53	42.24	61.82	70.62	42.99	57.75	56.40	43.23	65.51	55.80
$K=1$	ord-FS+dev	62.20	**45.92**	59.33	71.15	38.70	58.70	53.63	47.12	65.51	55.81
	ord-FS	48.53	33.91	54.56	63.36	40.43	49.59	47.56	23.49	65.51	47.44
	co-FT	50.93	36.83	54.04	61.72	38.46	51.35	46.26	33.48	65.51	48.73
	naive-co-train	62.46	42.52	62.32	72.32	43.44	58.28	54.87	**49.86**	67.17	57.03
	gradient-co-train	**62.60**	45.07	**62.88**	**72.43**	**46.05**	**58.82**	**55.47**	47.64	**67.81**	**57.64**
$K=5$	ord-FS+dev	58.59	46.68	59.23	69.87	41.19	**59.33**	54.47	45.50	65.51	55.60
	ord-FS	54.07	36.85	54.82	65.39	40.91	53.46	47.61	42.04	65.51	51.19
	co-FT	58.67	43.59	57.46	67.09	44.04	54.65	56.17	43.23	65.51	54.49
	naive-co-train	62.42	47.51	61.64	72.39	46.06	59.16	57.62	**53.96**	66.58	58.59
	gradient-co-train	**63.52**	**49.11**	**63.87**	**73.29**	**46.17**	59.09	**58.20**	53.19	**67.58**	**59.34**
$K=10$	ord-FS+dev	61.78	44.67	59.32	69.96	41.29	59.23	52.73	45.79	65.51	55.59
	ord-FS	59.46	43.21	56.21	65.88	40.67	52.64	53.45	41.61	65.51	53.17
	co-FT	60.51	44.64	58.42	67.23	44.99	56.39	58.12	44.09	65.51	55.54
	naive-co-train	**64.87**	**48.02**	62.12	72.63	**47.91**	60.43	**60.44**	46.18	**67.32**	58.88
	gradient-co-train	64.17	47.46	**63.37**	**72.77**	47.26	**60.48**	60.13	**52.73**	66.85	**59.47**

Table 6: Full results (F1) of the TyDiQA task.

K	Methods	ar	bg	de	el	es	fr	hi	ru	sw	th	tr	ur	vi	zh	en	Avg.
$K=0$	Zero-Shot	72.28	77.15	75.97	74.71	78.56	77.19	69.10	73.95	62.08	71.52	72.32	65.39	74.15	73.67	85.19	73.55
$K=1$	ord-FS+dev	**72.08**	77.49	**76.19**	**75.47**	**79.21**	**77.99**	69.16	74.69	62.29	72.31	**72.46**	66.00	74.60	74.20	**85.19**	73.95
	ord-FS	71.60	77.43	76.09	75.29	78.95	77.64	69.42	74.47	61.46	72.27	72.10	65.55	74.72	74.20	**85.19**	73.70
	co-FT	71.56	77.28	75.88	74.81	78.42	77.33	69.34	74.39	61.58	71.64	72.08	65.40	74.33	73.84	**85.19**	73.54
	naive-co-train	71.52	76.83	75.89	74.74	77.88	77.25	68.99	74.67	62.80	71.07	71.77	65.17	73.74	72.23	83.84	73.29
	gradient-co-train	71.97	**77.76**	76.12	75.27	78.47	**77.74**	**70.06**	**75.47**	**64.08**	**72.49**	**72.10**	**66.25**	**74.90**	**74.48**	84.20	**74.09**
$K=5$	ord-FS+dev	71.98	77.72	**76.55**	75.48	78.69	77.48	70.16	74.76	62.29	72.68	72.30	65.82	75.31	74.77	**85.19**	74.08
	ord-FS	71.57	77.25	76.18	75.39	78.64	77.03	69.94	74.44	61.59	72.33	71.92	65.21	74.93	74.37	**85.19**	73.73
	co-FT	70.85	76.37	75.23	74.20	77.41	76.79	69.09	74.19	61.89	71.27	71.13	65.28	74.00	74.05	**85.19**	73.13
	naive-co-train	72.02	77.43	76.12	74.74	78.19	77.41	69.63	74.67	62.95	72.02	72.22	65.86	74.33	73.86	83.90	73.69
	gradient-co-train	**72.05**	**77.89**	76.54	**75.48**	**78.83**	**77.94**	**70.64**	**76.12**	**63.94**	**72.83**	**72.40**	**67.15**	**75.31**	**75.15**	83.90	**74.41**
$K=10$	ord-FS+dev	71.74	77.51	**76.73**	75.33	79.03	77.69	70.11	75.09	62.46	72.92	**72.76**	66.00	75.28	75.03	**85.19**	74.19
	ord-FS	71.31	77.65	76.38	74.83	**79.20**	77.43	70.12	75.20	62.43	72.77	72.72	65.58	75.14	74.84	**85.19**	74.03
	co-FT	71.32	76.77	75.80	74.58	77.77	77.11	69.72	74.73	62.15	72.36	71.78	65.96	74.44	74.67	**85.19**	73.62
	naive-co-train	**72.22**	77.67	76.47	**75.39**	78.29	77.52	70.53	75.57	63.05	72.51	72.35	66.76	74.69	74.65	84.23	74.23
	gradient-co-train	71.74	**78.04**	76.61	75.29	78.89	**77.79**	**70.95**	**75.90**	**63.74**	**73.15**	72.41	**67.07**	**75.48**	**75.43**	84.10	**74.44**

Table 7: Full results (accuracy) of the XNLI task.

NER		POS		TyDiQA		XNLI	
lang.	Δ F1 (%)	lang.	Δ F1 (%)	lang.	Δ F1 (%)	lang.	Δ Acc. (%)
pl	0.90	vi	-4.35	ko	5.25	es	-0.32
ms	1.51	ja	-1.85	ru	5.62	tr	-0.31
nl	2.66	he	-1.36	id	7.90	de	0.23
af	2.73	hu	-1.02	fi	9.05	vi	0.34
sw	2.76	zh	-0.71	ar	9.45	fr	0.36

Table 8: Top-5 languages that achieve the least improvement by using gradient-co-train compared to ord-FS on all tasks in 5-shot learning.

	ar	hi	my	pa	hu	nl	fr	tr
ord-FS	57.69	64.08	52.44	52.28	79.91	82.11	76.99	79.97
gradient-single	71.98	72.82	57.67	63.60	80.28	83.89	81.51	80.10
gradient-subset	70.57	71.79	61.18	62.99	81.65	83.96	81.90	82.37
gradient-all	71.83	76.76	63.78	69.88	83.06	84.77	83.64	85.21

Table 9: Numeric results of Figure 2a.

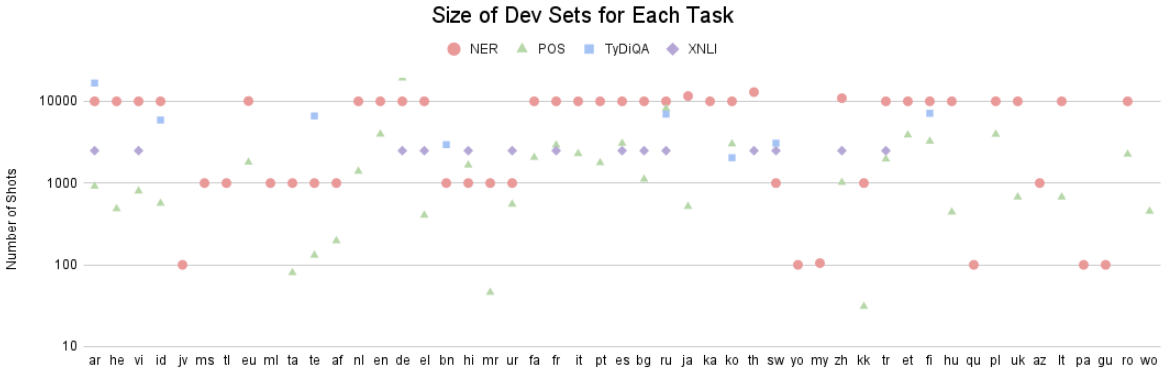


Figure 4: The size of dev sets that we use in the experiments for each language in each task.

	ar	fi	id	sw	te
ord-FS	54.07	54.82	65.39	47.61	42.04
gradient-single	64.43	62.52	72.78	56.91	54.28
gradient-subset	64.04	63.17	72.44	57.48	54.49
gradient-all	63.52	63.87	73.29	58.20	53.19

Table 10: Numeric results of Figure 2b.

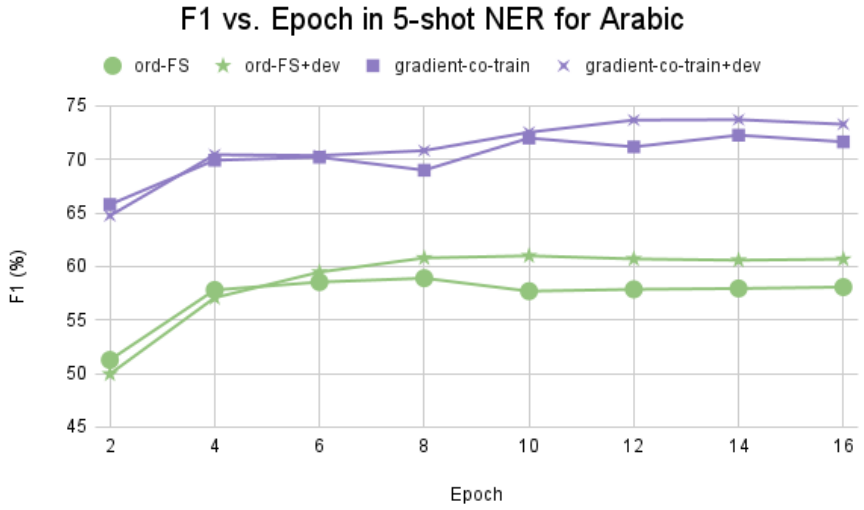


Figure 5: F1 scores of gradient-co-train(+dev) and ord-FS(+dev) with increasing number of epochs. The large dev set helps model selection after certain epochs. Gradient-co-train shows less gap led by the dev set than ord-FS and can select approximately optimal model by only using the source dev set.