Orthogonal Language and Task Adapters in Zero-Shot Cross-Lingual Transfer

Anonymous ACL submission

Abstract

Adapter modules have recently been used for efficient fine-tuning and language specialization of massively multilingual Transformers (MMTs), improving downstream zero-shot cross-lingual transfer. In this work, we propose orthogonal language and task adapters (dubbed orthoadapters) for cross-lingual transfer. They are trained to encode language- and task-specific information that is complementary (i.e., orthogonal) to the knowledge already stored in the pretrained MMT parameters. Our zero-shot transfer experiments, involving three tasks and 10 diverse languages, indicate that the usefulness of orthoadapters in cross-lingual transfer, especially for the most complex NLI task, but also cross-lingual transfer. In this work, we advance the idea of augmenting MMT’s knowledge through specialized adapter modules. We aim to maximize the injection of novel information into both language- and task-specific adapter parameters, that is, we enforce the adapters to encode the information that complements the knowledge encoded in MMT’s pretrained parameters. To achieve this, we propose to learn orthogonal adapters (or orthoadapters for short). We augment the training objective with the orthogonality loss: it forces the representations produced by the adapters to be orthogonal to representations from the corresponding MMT layers, see Figure 1.

1 Introduction

Massively multilingual transformers (MMTs), pretrained on large multilingual corpora via language modeling (LM) objectives (Devlin et al., 2019; Conneau et al., 2020) have overthrown (static) cross-lingual word embeddings (Ruder et al., 2019; Glavaš et al., 2019) as the state-of-the-art paradigm for zero-shot (ZS) cross-lingual transfer. However, MMTs are constrained by the so-called curse of multilinguality: the quality of language-specific representations starts decreasing when the number of training languages exceeds the MMT’s parameter capacity (Arivazhagan et al., 2019; Conneau et al., 2020). Languages with smallest training corpora are most affected: the largest transfer performance drops occur with those target languages (Lauscher et al., 2020; Wu and Dredze, 2020).

Additional LM training of a full pretrained MMT on monolingual corpora of an underrepresented language is a partial remedy towards satisfactory downstream transfer (Wang et al., 2020; Ponti et al., 2020). However, this approach does not increase the MMT capacity and, consequently, might deteriorate representations for other languages. Adapters (Houlsby et al., 2019; Bapna and Firat, 2019), additional trainable parameters inserted into the MMT’s layers, have recently been used for their language and task specialization (Pfeiffer et al., 2020b), offering improved and more efficient ZS cross-lingual transfer. The current adapter-based approaches, however, do not provide any mechanism that would prevent language and task adapters from capturing redundant information, that is, from storing knowledge already encoded in the MMT’s parameters.

In this work, we advance the idea of augmenting MMT’s knowledge through specialized adapter modules. We aim to maximize the injection of novel information into both language- and task-specific adapter parameters, that is, we enforce the adapters to encode the information that complements the knowledge encoded in MMT’s pretrained parameters. To achieve this, we propose to learn orthogonal adapters (or orthoadapters for short). We augment the training objective with the orthogonality loss: it forces the representations produced by the adapters to be orthogonal to representations from the corresponding MMT layers, see Figure 1.

Our proof-of-concept ZS transfer experiments on POS-tagging, NER, and natural language inference (XNLI), spanning 10 typologically diverse languages, render language-specific and task-specific orthoadapters viable mechanisms for improving ZS transfer performance. However, we show that the optimal use of orthogonality is also largely task-dependent. We hope that our study will inspire a wider investigation of applicability and usefulness of orthogonality constraints for MMT fine-tuning.

---

1 For a language adapter, the training task is masked language modeling on the monolingual corpus of that language.
Orthogonality Loss. There is currently no mechanism in adapter-based approaches that would explicitly prevent adapter parameters from learning redundant information, already captured by the pretrained MMT. Inspired by the idea of orthogonal text representations from prior work on multi-task learning (Romera-Paredes et al., 2012; Liu et al., 2017), we introduce an auxiliary orthogonality loss to adapter-based language and task fine-tuning. It explicitly forces the adapters to dedicate their capacity to new knowledge, which should be complementary (i.e., non-redundant) to the knowledge already encoded in existing MMT parameters.

Let $x_h^{(i,j)}$ denote the hidden representation in the $i$-th layer of the MMT for the $j$-th token in the sequence, input to the adapter. Let $x_a^{(i,j)}$ be the corresponding output of the same adapter for the same token, as given in Eq. (1). The orthogonality loss of the $j$-th token in the $i$-th MMT layer is then simply the square of the cosine similarity between $x_h^{(i,j)}$ and $x_a^{(i,j)}$; we then derive the overall orthogonality loss by averaging token-level losses in each layer and then summing layer-level losses:

$$\mathcal{L}_{ORT} = \frac{1}{T} \sum_{j=1}^{T} \cos \left( x_h^{(i,j)}, x_a^{(i,j)} \right)^2$$

where $T$ is the maximal length of the input token sequence and $N$ is the number of MMT’s layers.

Two-Step Orthoadapter Training. We first train language orthoadapters (part a of Figure 1), independently for each language, aiming at extending the language knowledge in the pretrained MMT. We use MLM (cross-entropy loss) as the main training objective $\mathcal{L}_{MLM}$. We then alternately update the parameters of language orthoadapters, first by minimizing $\mathcal{L}_{MLM}$ and then $\mathcal{L}_{ORT}$.

In the second step (part b in Figure 1), the goal is to maximize the amount of novel information useful for a concrete downstream task: we train the task orthoadapters on the task-specific training data (POS, NER, XNLI) by alternately minimizing 1) the task-specific objective $\mathcal{L}_{TASK}$ and 2) the orthogonality loss $\mathcal{L}_{ORT}$. Note, however, that in this case $x_h$ is, in each transformer layer, first adapted by the source language adapter and then by the task adapter, and $x_a$ is the output of the task adapter.

$^2$We use two independent Adam optimizers (Kingma and Ba, 2015), one for each loss. We also experimented with minimizing the joint loss $\mathcal{L}_{MLM} + \lambda \mathcal{L}_{ORT}$ but this generally yielded poorer performance over a range of $\lambda$ values.

$^3$Cross-entropy loss for the whole sequence for NLI; sum of token-level cross-entropy losses for POS and NER.
Zero-Shot Cross-Lingual Transfer then proceeds in the same vein as in prior work (Pfeiffer et al., 2020b). It is conducted by simply replacing the source language orthoadapter with the target language orthoadapter while relying on exactly the same task adapter fine-tuned with the labeled source language data, stacked on top of the language adapters (see part c of Figure 1).

### 3 Experimental Setup

**Model Configurations.** The decomposition into two adapter types in the two-step procedure (Figure 1) allows us 1) to use language orthoadapters (L-ORT) instead of regular non-orthogonal language adapters (L-NOO); and/or 2) to replace non-orthogonal task adapters (T-NOO) with task orthoadapters (T-ORT). These choices give rise to four different model variants, where the L-NOO+T-NOO variant is the baseline MAD-X variant.

We also test the usefulness of task orthoadapters in a setup without dedicated language adapters: T-ORT variants are compared to T-NOO variants, and also to standard (computationally more intensive) full fine-tuning of the whole MMT (FULL-FT).

**Evaluation Tasks and Data.** We evaluate all model variants on standard cross-lingual transfer tasks, relying on established evaluation benchmarks: 1) sentence-pair classification on XNLI (Conneau et al., 2018); 2) cross-lingual named entity recognition (NER) on the WikiANN dataset (Pan et al., 2017); 3) part-of-speech tagging with universal POS tags from the Universal Dependencies (Nivre et al., 2018) (UD-POS).

In all experiments we rely on the pretrained multilingual XLM-R (Base) model (Conneau et al., 2020). English (EN) is our (resource-rich) source language. For completeness, we also report the results on the EN test data, i.e., without any transfer.

**10 Target Languages**, with their language codes available in the appendix, span 5 geographical macro-areas (Ponti et al., 2020) and 8 distinct language families. In NER evaluations we include three truly low-resource languages: Quechua, Ilo-

cano, and Meadow Mari.\(^7\)

### 4 Results and Discussion

The results of ZS transfer are summarized in Table 1 (XNLI), Table 2 (UD-POS), and Table 3 (NER). First, a comparison with the FULL-FT variant confirms findings from prior work (Pfeiffer et al., 2021), validating the use of the more efficient adapter-based approach: the ZS scores with adapter-based variants are on a par with or even higher than the scores reported with FULL-FT across the board.

**Regular vs Orthogonal Language Adapters.**

First, the usefulness of language orthoadapters (L-ORT variants) does depend on the task at hand and its complexity. As an encouraging finding, we observe consistent gains in cross-lingual NLI,\(^9\) at least +1 accuracy point on 4/5 target languages with the L-ORT+T-NOO variant. This variant also yields highest average ZS performance, and slight (but statistically insignificant) gains on EN NLI. The picture is less clear for UD-POS and NER: L-ORT+T-NOO does have a slight edge over the baseline L-NOO+T-NOO variant in UD-POS, but this seems to be due to large gains in Chinese. In a similar vein, while L-ORT+T-NOO is the best performing variant in NER on average, the gains over L-NOO+T-NOO are slight, and inconsistent across

---

\(^7\)The selection of target languages has been guided by several (sometimes clashing) criteria: C1 typological diversity; C2 availability in the standard evaluation benchmarks; C3 computational tractability; C4 evaluation also on truly low-resource languages. Given that the main computational bottleneck is MLM-ing for learning language adapters, we have started from the subset of languages represented in our evaluation datasets (C2) for which pretrained language adapters (regular, non-orthogonal) are already available online (C3) (Pfeiffer et al., 2020a), also respecting C1 and C4.

\(^8\)We select task (ortho)adapters solely based on the performance on the source language (i.e. English) dev set.

\(^9\)XNLI is arguably the most complex (reasoning) task in our evaluation and, unlike UD-POS and NER, requires successful high-level semantic modeling and ZS transfer.
languages (e.g., large gains on ILO, some on AR and SW, but some decrease on QU and MHR). We again speculate that this is mostly due to the nature and complexity of the task at hand.

Orthogonal Task Adapters display a different behavior, but we can again largely relate it to the properties of the evaluation tasks. First, task orthoadapters seem detrimental for XNLI (compare L-NOO+T+ORT vs. L-NOO+T-NOO as well as L-ORT+T+ORT vs. L-ORT+T-NOO in Table 1), and also yield no real benefits in the simpler setup (T-ORT vs. T-NOO). The two main objectives – (i) MLM for the original MMT pretrained and language adapter training, and (ii) cross-entropy loss for the whole sequence for NLI – might be structurally too different for the orthogonality loss to capture any additional task-related information. However, task orthoadapters seem to be useful UD-POS, with substantial gains reported on 3/5 languages – Arabic, Chinese, Hindi, all of which have non-Latin scripts. Combining them with language orthoadapters, however, does deteriorate the performance. The overall trend is even more complex with NER: while there are clear hints that using orthoadapters is useful for some languages and some model variants, there is still a substantial variance in the results.

5 Conclusion

We investigated how orthogonality constraints impact zero-shot (ZS) cross-lingual transfer via massively multilingual transformers (MMTs, e.g., XLM-R) for three standard tasks: NLI, POS, and NER. Relying on the standard adapter-based transfer techniques, we introduced the idea of orthogonal language and task adapters (or orthoadapters): we explicitly enforce the information stored in the parameters of the orthoadapters to be orthogonal to the information already stored in the pretrained MMT. In general, our results suggest that explicitly controlling for the information that gets captured in the orthoadapters can have a positive impact on ZS transfer via MMTs. The use of orthogonality, however, seems to be language- and task-dependent, warranting further investigations in future work.

The code will be available at: [URL].

10In order to perform cross-lingual transfer for NLI, the underlying MMT must capture and leverage more language-specific nuances than for sequence labeling tasks such as POS-tagging and NER. By enforcing the capture of non-redundant information in the additional language-specific adapters, we allow the model to store additional and, more importantly, novel target language information. While the same information is available also for NER and POS tagging, they require ‘shallower’ language-specific knowledge (Lauscher et al., 2020); this is why more complex target language-specific knowledge captured in orthoadapters (compared to regular non-orthogonal language adapters) does not make a difference.

11In fact, we speculate that the orthogonal loss might have emphasized this discrepancy between the objectives.

Table 1: Accuracy scores (×100%) of zero-shot transfer for the natural language inference task on the XNLI dataset. See §3 for the descriptions of different model variants. EN is the source language in all experiments. The scores in the AVGz column denote the average performance of zero-shot transfer (i.e., without English results).

<table>
<thead>
<tr>
<th>Variant</th>
<th>EN</th>
<th>AR</th>
<th>HI</th>
<th>SW</th>
<th>TR</th>
<th>ZH</th>
<th>AVGz</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL-FT</td>
<td>83.67</td>
<td>72.01</td>
<td>68.26</td>
<td>63.77</td>
<td>71.75</td>
<td>73.11</td>
<td>69.86</td>
</tr>
<tr>
<td>T-NOO</td>
<td>84.05</td>
<td>69.51</td>
<td>68.26</td>
<td>64.48</td>
<td>71.73</td>
<td>72.25</td>
<td>69.25</td>
</tr>
<tr>
<td>T-ORT</td>
<td>84.28</td>
<td>68.84</td>
<td>63.35</td>
<td>70.65</td>
<td>71.95</td>
<td>71.95</td>
<td>68.99</td>
</tr>
<tr>
<td>L-NOO+T+ORT</td>
<td>85.59</td>
<td>71.71</td>
<td>70.67</td>
<td>67.08</td>
<td>71.75</td>
<td>73.29</td>
<td>70.76</td>
</tr>
<tr>
<td>L-NOO+T-ORT</td>
<td>84.79</td>
<td>68.84</td>
<td>69.71</td>
<td>66.34</td>
<td>70.47</td>
<td>71.49</td>
<td>69.38</td>
</tr>
<tr>
<td>L-ORT+T+NOO</td>
<td>84.73</td>
<td>72.25</td>
<td>69.28</td>
<td>69.10</td>
<td>72.80</td>
<td>73.61</td>
<td>71.43</td>
</tr>
<tr>
<td>L-ORT+T-ORT</td>
<td>84.35</td>
<td>69.73</td>
<td>68.56</td>
<td>67.92</td>
<td>71.23</td>
<td>71.53</td>
<td>69.79</td>
</tr>
</tbody>
</table>

Table 2: F1 scores (×100%) of zero-shot transfer in the UD-POS task.

<table>
<thead>
<tr>
<th>Variant</th>
<th>EN</th>
<th>AR</th>
<th>ET</th>
<th>HI</th>
<th>ILO</th>
<th>MHR</th>
<th>QU</th>
<th>TR</th>
<th>ZH</th>
<th>AVGz</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL-FT</td>
<td>95.94</td>
<td>66.42</td>
<td>84.68</td>
<td>70.38</td>
<td>74.01</td>
<td>35.59</td>
<td>66.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T-NOO</td>
<td>95.59</td>
<td>64.35</td>
<td>84.43</td>
<td>71.06</td>
<td>72.68</td>
<td>31.47</td>
<td>64.80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T-ORT</td>
<td>95.68</td>
<td>65.28</td>
<td>85.17</td>
<td>70.42</td>
<td>72.93</td>
<td>40.03</td>
<td>67.77</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L-NOO+T+ORT</td>
<td>95.59</td>
<td>67.77</td>
<td>85.31</td>
<td>69.62</td>
<td>74.17</td>
<td>79.25</td>
<td>67.92</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L-NOO+T-ORT</td>
<td>95.66</td>
<td>67.15</td>
<td>85.67</td>
<td>71.57</td>
<td>74.08</td>
<td>31.68</td>
<td>66.03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L-ORT+T+NOO</td>
<td>95.63</td>
<td>66.62</td>
<td>84.26</td>
<td>68.93</td>
<td>73.02</td>
<td>29.50</td>
<td>64.27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L-ORT+T-ORT</td>
<td>95.63</td>
<td>67.40</td>
<td>84.22</td>
<td>67.26</td>
<td>71.21</td>
<td>31.79</td>
<td>64.38</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: F1 scores (×100%) of zero-shot transfer in the NER task on the WikiAnn dataset.
References


A Training Details

A.1 Task Training Details
We processed the data for all tasks using the pre-processing pipeline provided with the XTREME benchmark (Hu et al., 2020).\textsuperscript{13}

XNLI. In NLI training (i.e., for XNLI transfer) we were trained for 30 epochs with the batch size of 32. Maximum sequence length was 128 input tokens. Gradient norms were clipped to 1.0.

UD-POS. We trained for 50 epochs with batch size of 16. Maximum sequence length was 128 input tokens. Gradient norms were clipped to 1.0.

NER. We trained for 100 epochs with the batch size of 16. Maximum sequence length was 128 input tokens. Gradient norms were clipped to 1.0.

As prior work, we use the data splits of Rahimi et al. (2019).

A.2 Experimental Setup without Language Adapters
For the “non-MAD-X” experimental setup (i.e., the setup without language adapters, see §3), we relied on our own implementation of the adapter module. The bottleneck size for the task adapter was set to $d = 64$. For (X)NLI, we searched the following learning rate grid: $[2e^1, 5e^1, 7e^1, 5e^2]$: for UD-POS and WikiAnn the corresponding learning rate grid was $[5e^1, 1e^0, 1e^-1]$: For task orthoadapters, we searched the following additional learning rate grid for the orthogonal loss optimizer: $[1e^-1, 1e^-2, 1e^-3]$.  

A.3 Full Experimental Setup
For the more complex multi-adapter setup based on the MAD-X framework (i.e., with both language and task adapters, see Figure 1), we utilized the Adapter-Transformers library and the underlying AdapterHub service (Pfeiffer et al., 2020a).

Task Adapters and Orthoadapters. We followed the recommendation from the original paper (Pfeiffer et al., 2020b). We utilized the Pfeiffer configuration\textsuperscript{14} found in the Adapter-Transformers library with the adapter dimensionality of 48. Due to the computational constraints, the learning rate grid was $[1e^-1, 1e^-2, 1e^-3]$.  

Regular Language Adapters. We utilized the pretrained language adapters readily available via the AdapterHub service (Pfeiffer et al., 2020a). These language adapters have the dimensionality of 384. They were trained (while the rest of the model was frozen) by executing the MLM-ing for 250,000 iterations on the Wikipedia data in the target language. 

Orthogonal Language Adapters. We started from the MLM-ing training script for training language adapters provided by the Adapter-Transformers library and trained language orthoadapters on the Wikipedia data, relying on the setup of AdapterHub’s regular language adapters (dimensionality 384, 250,000 iterations). Due to computational constraints we reduced the maximum sequence length of the input to 128 tokens, while the batch size was 8. Finally, for the main optimizer and orthogonality loss optimizer we used the learning rates of $1e^-4$ and $1e^-7$, respectively.

A.4 Statistical Significance Testing
Statistical significance ($p < 0.05$) is reported following the recommended statistical significance tests for each task, see: https://arxiv.org/pdf/1809.01448.pdf. 

\textsuperscript{13}https://github.com/google-research/xtreme

\textsuperscript{14}Pfeiffer et al. (2021) found this configuration to perform on a par with the configuration proposed by Houlsby et al. (2019), who inject two adapter modules per transformer layer (the other one after the multi-head attention sublayer), while being more efficient to train.
<table>
<thead>
<tr>
<th>Language</th>
<th>Family</th>
<th>Type</th>
<th>ISO 639</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>IE: Germanic</td>
<td>fusional</td>
<td>EN</td>
<td>XNLI, UD-POS, NER</td>
</tr>
<tr>
<td>Arabic</td>
<td>Semitic</td>
<td>introflexitive</td>
<td>AR</td>
<td>XNLI, UD-POS, NER</td>
</tr>
<tr>
<td>Estonian</td>
<td>Uralic: Finnic</td>
<td>agglutinative</td>
<td>ET</td>
<td>UD-POS, NER</td>
</tr>
<tr>
<td>Hindi</td>
<td>IE: Indo-Aryan</td>
<td>fusalional</td>
<td>HI</td>
<td>XNLI, UD-POS, NER</td>
</tr>
<tr>
<td>Ilocano</td>
<td>Austronesian</td>
<td>agglutinative</td>
<td>ILO</td>
<td>NER</td>
</tr>
<tr>
<td>Meadow Mari</td>
<td>Uralic: Mari</td>
<td>agglutinative</td>
<td>MHR</td>
<td>NER</td>
</tr>
<tr>
<td>Quechua</td>
<td>Quechuan</td>
<td>agglutinative</td>
<td>QU</td>
<td>NER</td>
</tr>
<tr>
<td>Kiswahili</td>
<td>Niger-Congo: Bantu</td>
<td>agglutinative</td>
<td>SW</td>
<td>XNLI, NER</td>
</tr>
<tr>
<td>Turkish</td>
<td>Turkic</td>
<td>agglutinative</td>
<td>TR</td>
<td>XNLI, UD-POS, NER</td>
</tr>
<tr>
<td>Mandarin Chinese</td>
<td>Sino-Tibetian</td>
<td>isolating</td>
<td>ZH</td>
<td>XNLI, UD-POS, NER</td>
</tr>
</tbody>
</table>

Table 4: Target languages used in the main experiments along with their corresponding language family (IE=Indo-European), morphological type, and ISO 639-1 code (or ISO 639-2 for Ilocano; or ISO 639-3 for Meadow Mari). We use English (EN) as the source language in all experiments. EN is a fusional language (IE: Germanic).