Efficient Test Time Adapter Ensembling for Low-resource Language Varieties

Xinyi Wang\textsuperscript{1} Yulia Tsvetkov\textsuperscript{2} Sebastian Ruder\textsuperscript{3} Graham Neubig\textsuperscript{1}
\textsuperscript{1}Language Technology Institute, Carnegie Mellon University
\textsuperscript{2}Paul G. Allen School of Computer Science & Engineering, University of Washington
\textsuperscript{3}DeepMind
xinyiw1@cs.cmu.edu, yuliats@cs.washington.edu, ruder@google.com, gneubig@cs.cmu.edu

Abstract

Adapters are light-weight modules that allow parameter-efficient fine-tuning of pretrained models. Specialized language and task adapters have recently been proposed to facilitate cross-lingual transfer of multilingual pretrained models (Pfeiffer et al., 2020b). However, this approach requires training a separate language adapter for every language one wishes to support, which can be impractical for languages with limited data. An intuitive solution is to use a related language adapter for the new language variety, but we observe that this solution can lead to sub-optimal performance.

In this paper, we aim to improve the robustness of language adapters to uncovered languages without training new adapters. We find that ensembling multiple existing language adapters makes the fine-tuned model significantly more robust to other language varieties not included in these adapters. Building upon this observation, we propose Entropy Minimized Ensemble of Adapters (EMEA), a method that optimizes the ensemble weights of the pretrained language adapters for each test sentence by minimizing the entropy of its predictions. Experiments on three diverse groups of language varieties show that our method leads to significant improvements on both named entity recognition and part-of-speech tagging across all languages.

1 Introduction

Massively multilingual pretrained models (Devlin et al., 2019; Huang et al., 2019; Conneau and Lample, 2019; Conneau et al., 2020) combined with cross-lingual transfer now define the state of the art on a variety of NLP tasks (Hu et al., 2020). Within this paradigm, multilingual pretrained models are fine-tuned on annotated data of a task in a high-resource language, and transferred to other languages. Several recent works propose parameter-efficient fine-tuning methods that insert small adapter modules between the layers of pretrained models (Rebuffi et al., 2017; Houlsby et al., 2019). In this line of work, the pretrained model is usually frozen while only the adapters are fine-tuned for a downstream task, which is conducive to both improving the model’s learning ability and compactness with respect to storage on disk or in memory. The adapters can be applied to the cross-lingual transfer setting by training separate language and task adapters (Pfeiffer et al., 2020b; Üstün et al., 2020). Specifically, Pfeiffer et al. (2020b) propose to perform zero-shot transfer by first training language-level adapters on monolingual data in different languages and then a task adapter on annotated data in the source language.

One drawback of this framework is that a separate language adapter is required for each target language, which is problematic in cases where the data to train these adapters cannot be easily obtained, such as for languages with diverse regional or demograhic variations. In fact, certain language varieties are not included in the standard language identification tools, which makes it challenging to reliably obtain even unlabeled data (Salameh et al., 2018; Caswell et al., 2020; Demszyk et al., 2021). To give just one example, the Nordic languages and dialects form a dialect continuum where the total number of language varieties is difficult to estimate, and language varieties constantly emerge in culturally and linguistically diverse areas (Svendsen and Røyneland, 2008; Røyneland and Jensen, 2020). Although highly related, these language...
varieties have many systematic differences, which need to be addressed by NLP systems that equitably serve all speakers (Kumar et al., 2021). One potential mitigation strategy is directly using an adapter trained on another similar language variety, but we find this sub-optimal in experiments (§ 4).

Instead, we propose two methods to combine existing language adapters to adapt the model to new language varieties at test time without any training data. First, we find that simply ensembling multiple related language adapters can significantly improve the fine-tuned model, compared with using individual language adapters. Second, we propose Entropy Minimized Ensemble of Adapters (EMEA; Fig. 1), which adapts the ensemble weight of the language adapters for each test instance by minimizing the ensemble model’s prediction uncertainty. Our experiments show that EMEA further improves over vanilla ensembling for three groups of uncovered language varieties with many systematic differences, which need to be addressed by NLP systems that equitably serve all speakers (Kumar et al., 2021). One potential mitigation strategy is directly using an adapter trained on another similar language variety, but we find this sub-optimal in experiments (§ 4).

Second, for each task $T$, its adapter $T_i$ is inserted on top of a $src$ language adapter $L_{src}$. The resulting model $T_i \circ L_{src} \circ M$ is trained on the downstream task $T_i$ in language $L_{src}$. After these two steps, $T_i \circ L_j \circ M$ can be used to perform zero-shot cross-lingual transfer for any task $T_i$ and language $L_j$.

### Shortcomings
This approach requires a separate adapter for each language one wishes to support. The online database AdapterHub\(^1\) aims to improve the efficiency and reuse of trained language and task adapters (Pfeiffer et al., 2020a) but currently supports only about 50 languages, and hence most languages are not covered. More importantly, as mentioned in the introduction, certain languages have diverse regional varieties and difficulty of reliably obtaining data for them makes adapter-based approaches especially brittle in these cases. In the following § 3, we propose strategies to improve the robustness of language adapters to uncovered languages without training new adapters.

### 3 Generalizing Language Adapters to Related Languages

We consider the setting where we have a multilingual pretrained model $M$ as well as the pretrained task adapters $T_1, T_2, ..., T_m$ and language adapters $L_1, L_2, ..., L_n$. We want to use $M$ and the existing adapters to support a new language $L_{new}$, which is not in $\{L_1, L_2, ..., L_n\}$ on a given task $T$ without training a new adapter for $L_{new}$.

#### Related Language Adapters
One potential solution is to find the most related language $L_{rel} \in \{L_1, L_2, ..., L_n\}$ and then use $T \circ L_{rel} \circ M$ to do inference in $L_{new}$. However, this has two disadvantages. First, the task adapter $T$ is only trained in the setting of $T \circ L_{src} \circ M$, so it might not generalize well to the test time setting of $T \circ L_{rel} \circ M$ (as shown in § 4.1). Second, while the pretrained model $M$ may be relatively robust against distribution shifts (Hendrycks et al., 2020), the specialized language adapters might make the model brittle to language variations because they are trained for specific languages. Our experiments in § 4.1 show that this solution indeed leads to poor performance.

#### Adapter Ensembling
As a first solution to this problem, we propose an extremely simple strategy of averaging the transformed outputs of multiple language adapters. Specifically, we use both the source language adapter $L_{src}$ and adapters from related languages with similar linguistic properties to the new language. Let $R$ be the set of the source and related language adapters. To do inference on a task $T$ for the new language $L_{new}$, we transform $L_{rel} \in R$ into $L_{src}$ and use the resulting model $L_{src} \circ M$ for prediction. This approach achieves strong performance across a variety of languages and tasks, especially when the language adapter $L_{rel}$ is not trained for the related language.

\(^1\)https://adapterhub.ml/
the output $h$ of each layer in $\mathcal{M}$ with the language adapters as $L_{\text{avg}}(h) = \frac{1}{T} \sum_{i=1}^{T} L_i(h)$.

**Entropy Minimized Ensemble of Adapters**

While ensembling is a simple and effective strategy to combine multiple potentially beneficial language adapters, the equal weighing of all language adapters could be sub-optimal for $L_{\text{new}}$; different language varieties, or even sentences, could benefit from a different weighting of the pretrained language adapters. To further improve adapter ensembling, we generalize $L_{\text{avg}}(h)$ into a learnable weighted average:

$$L_{\text{avg}}(h) = \sum_{i=1}^{R} \alpha_i L_i(h)$$

where $\alpha_1, \alpha_2, ..., \alpha_R$ are learnable weights satisfying $\alpha_i \geq 0$ and $\sum_{i=1}^{R} \alpha_i = 1$. Next, we propose Entropy Minimized Ensemble of Adapters (EMEA) method, which learns the adapter weightings for each sentence without additional training.

The intuition behind our method is that a good adapter weight $\alpha$ for a test input $x$ should make the model more confident in its prediction for $x$, that is, it should lead to lower model entropy over the input (Shannon, 1948; Wang et al., 2021). Specifically for structured prediction tasks, we want to classify each word $x_w$ in a test input $x$ with $W$ words into one of the possible $C$ classes. We consider the entropy: $H(x; \alpha) = -\sum_{w=1}^{W} \sum_{c=1}^{C} P(c|x_w; \alpha) \log P(c|x_w; \alpha)$, where $P(c|x_w; \alpha)$ is the prediction of the model $T \circ L_{\text{avg}}(h) \circ \mathcal{M}$. Since $P(c|x_w; \alpha)$ is a function of the ensemble weights $\alpha$, we can calculate the gradient of $\alpha$ as $g_i = \nabla_{\alpha_i} H(x; \alpha)$.

To minimize the entropy loss, we can simply do gradient descent steps on each $\alpha_i$ using the corresponding gradient $g_i$ by $\alpha_i = \alpha_i - \gamma g_i$, where $\gamma$ is the learning rate. We can then use the updated $\alpha$ to calculate the final prediction for $x$. In § 4, we find that a single step of gradient update already leads to better performance than simple ensembling. We can additionally perform multiple steps of gradient descent to obtain a better $\alpha$ at the cost of lower inference speed. Alg. 1 shows the pseudo code of our method.

**4 Experiments**

**Data** We focus on zero-shot cross-lingual transfer with English as the source language. We conduct experiments on named entity recognition (NER) and part-of-speech tagging (POS). We use the WikiAnn dataset (Pan et al., 2017) for NER and Universal Treebank 2.0 for POS tagging (Nivre et al., 2018).

**Model** We use the mBERT (Devlin et al., 2019) model, which shows good performance for low-resource languages on the structured prediction tasks (Pfeiffer et al., 2020b; Hu et al., 2020). We use the English annotated data to train the task adapter. Each experiment is run with 3 different random seeds and we report the average performance. More details can be found in Appendix A.

**Languages** Due to the scarcity of datasets for dialects, we focus on three groups of closely related languages to simulate the setup of language varieties. Each group has a language with a pretrained adapter available on the AdapterHub (Pfeiffer et al., 2020a), and we test on the languages without adapters. The language with adapter and the target languages for each group are: 1. Hindi (hi): Marathi (mr), Bengali (bn), Tamil (ta), Bhojpuri (bho); 2. Icelandic (is): Faroese (fo), Norwegian (no), Danish (da); 3. Russian (ru): Bulgarian (bg), Ukrainian (uk), Belorussian (be). For our methods, we additionally use the adapter for English (the src language), and optionally for another highly related language if there is one available on the AdapterHub. The adapters used are listed in Tab. 1.

**Algorithm 1: Training with EMEA**

Input: Uniform weights $\alpha^0$, weighted adapter output; $L_{\text{avg}}(h, \alpha^0)$; test data $x$; number of update steps $T$

Output: Prediction $\hat{y}$

1. for $t$ in 0, 1, ..., $T$ do
   1.1. Calculate entropy
   1.2. $H(x, \alpha) \leftarrow \text{Entropy}(T \circ L_{\text{avg}}(h, \alpha^t) \circ \mathcal{M})$
   1.3. Calculate gradient
   1.4. $g^t = \nabla_{\alpha} H(x, \alpha^t)$
   1.5. Update weighting
   1.6. $\alpha^{t+1} \leftarrow \text{Update}(\alpha^t, g^t)$
2. end

3. Calculate final prediction
4. $\hat{y} \leftarrow \text{Predict}(T \circ L_{\text{avg}}(h, \alpha^T) \circ \mathcal{M})$

---

<table>
<thead>
<tr>
<th>Related</th>
<th>Additional</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>hi</td>
<td>en,ar</td>
<td>mr, bn, ta, bho</td>
</tr>
<tr>
<td>is</td>
<td>en, de</td>
<td>fo, no, da</td>
</tr>
<tr>
<td>ru</td>
<td>en</td>
<td>be, uk, bg</td>
</tr>
</tbody>
</table>

Table 1: Test language groups and their corresponding language adapters. Adapters from languages in the first two columns are applied to the test languages in the third column.
Table 2: F1 of the baselines and our methods for each language group. EMEA-s1 updates the adapter weights with a single gradient step while EMEA-s10 updates for 10 steps.

<table>
<thead>
<tr>
<th>Method</th>
<th>mr</th>
<th>bn</th>
<th>ta</th>
<th>avg.</th>
<th>fo</th>
<th>no</th>
<th>da</th>
<th>avg.</th>
<th>be</th>
<th>uk</th>
<th>bg</th>
<th>avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>En</td>
<td>48.0</td>
<td>54.4</td>
<td>29.6</td>
<td>44.0</td>
<td>57.5</td>
<td>73.3</td>
<td>80.5</td>
<td>70.4</td>
<td>67.1</td>
<td>67.6</td>
<td>71.1</td>
<td>68.6</td>
</tr>
<tr>
<td>Related</td>
<td>51.7</td>
<td>47.0</td>
<td>30.8</td>
<td>43.1</td>
<td>54.3</td>
<td>72.7</td>
<td>79.3</td>
<td>68.7</td>
<td>66.2</td>
<td>65.8</td>
<td>69.8</td>
<td>67.3</td>
</tr>
<tr>
<td>CL</td>
<td>48.1</td>
<td>55.2</td>
<td>28.9</td>
<td>44.1</td>
<td>57.5</td>
<td>73.6</td>
<td>80.6</td>
<td>70.6</td>
<td>67.0</td>
<td>67.8</td>
<td>71.0</td>
<td>68.6</td>
</tr>
<tr>
<td>Fusion</td>
<td>49.8</td>
<td>58.3</td>
<td>33.7</td>
<td>47.2</td>
<td>56.0</td>
<td>69.3</td>
<td>77.8</td>
<td>67.7</td>
<td>70.1</td>
<td>69.1</td>
<td>72.3</td>
<td>70.5</td>
</tr>
<tr>
<td>Ensemble</td>
<td>55.5</td>
<td>53.5</td>
<td>35.8</td>
<td>48.8</td>
<td>57.4</td>
<td>74.0</td>
<td>80.8</td>
<td>70.7</td>
<td>70.5</td>
<td>72.2</td>
<td>74.2</td>
<td>72.3</td>
</tr>
<tr>
<td>EMEA-s1</td>
<td>57.2</td>
<td>61.2</td>
<td>37.4</td>
<td>51.9</td>
<td>59.2</td>
<td>74.3</td>
<td>81.3</td>
<td>71.6</td>
<td>71.5</td>
<td>72.9</td>
<td>74.9</td>
<td>73.1</td>
</tr>
<tr>
<td>EMEA-s10</td>
<td>57.5</td>
<td>63.2</td>
<td>38.3</td>
<td>53.0</td>
<td>61.6</td>
<td>74.9</td>
<td>82.0</td>
<td>72.8</td>
<td>72.9</td>
<td>72.9</td>
<td>75.1</td>
<td>73.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>mr</th>
<th>bho</th>
<th>ta</th>
<th>avg.</th>
<th>fo</th>
<th>no</th>
<th>da</th>
<th>avg.</th>
<th>be</th>
<th>uk</th>
<th>bg</th>
<th>avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>En</td>
<td>62.6</td>
<td>39.5</td>
<td>53.4</td>
<td>51.8</td>
<td>71.6</td>
<td>84.6</td>
<td>87.6</td>
<td>81.1</td>
<td>85.3</td>
<td>81.4</td>
<td>84.6</td>
<td>83.7</td>
</tr>
<tr>
<td>Related</td>
<td>53.2</td>
<td>46.9</td>
<td>47.0</td>
<td>49.0</td>
<td>72.8</td>
<td>82.4</td>
<td>86.9</td>
<td>80.7</td>
<td>84.0</td>
<td>79.5</td>
<td>82.9</td>
<td>82.1</td>
</tr>
<tr>
<td>CL</td>
<td>62.6</td>
<td>39.6</td>
<td>53.6</td>
<td>51.9</td>
<td>71.7</td>
<td>84.2</td>
<td>87.7</td>
<td>81.2</td>
<td>85.6</td>
<td>81.5</td>
<td>84.7</td>
<td>83.9</td>
</tr>
<tr>
<td>Fusion</td>
<td>59.8</td>
<td>42.3</td>
<td>53.5</td>
<td>51.8</td>
<td>72.9</td>
<td>81.3</td>
<td>86.0</td>
<td>80.0</td>
<td>85.8</td>
<td>80.0</td>
<td>83.3</td>
<td>83.0</td>
</tr>
<tr>
<td>Ensemble</td>
<td>62.2</td>
<td>45.5</td>
<td>53.7</td>
<td>53.8</td>
<td>73.9</td>
<td>83.6</td>
<td>87.9</td>
<td>81.8</td>
<td>85.9</td>
<td>81.6</td>
<td>84.6</td>
<td>84.0</td>
</tr>
<tr>
<td>EMEA-s1</td>
<td>62.1</td>
<td>45.1</td>
<td>54.3</td>
<td>53.8</td>
<td>74.0</td>
<td>83.5</td>
<td>87.8</td>
<td>81.7</td>
<td>86.2</td>
<td>81.4</td>
<td>84.6</td>
<td>84.0</td>
</tr>
<tr>
<td>EMEA-s10</td>
<td>62.5</td>
<td>44.9</td>
<td>55.6</td>
<td>54.3</td>
<td>73.8</td>
<td>83.7</td>
<td>88.0</td>
<td>81.8</td>
<td>86.0</td>
<td>81.6</td>
<td>84.9</td>
<td>84.2</td>
</tr>
</tbody>
</table>

Figure 2: Improvements over ensemble with different batch size.

Figure 3: Improvements by adding English adapter for different src language adapters.

Figure 4: Comparison to training adapter on different amount of monolingual data.

Baselines We compare with several baselines: 1) En: the English adapter; 2) Related: the best performing related language adapter; 3) Continual learning (CL): we use the English language adapter and update its parameters using the entropy loss for each test input; 4) Fusion: learn another set of key, value and query parameters in each layer that uses the layer output as a query to mix together the output of each adapter (Pfeiffer et al., 2021). Since we do not use labeled data in the new language, we train the fusion parameters on English labeled data.

4.1 Results

The results can be found in Tab. 2. For most languages using the English adapter is better than the best individual related language adapter. This confirms our hypothesis that specialized language adapters are not robust to language variations. CL leads to slight improvements for some languages but is generally comparable to En. Fusion improves over En for the NER task but it requires training and storing extra parameters. Its performance is also not consistent across languages and tasks, likely because it is only trained on English labeled data.

Using multiple language adapters brings significant gains Ensembling leads to significant gains for the non-Latin language group. It also brings improvements or is comparable to the best baseline on other languages. EMEA delivers further improvements across almost all languages, demonstrating the effectiveness of adapting language adapter weights to each test sentence. With only a single gradient update step on the ensemble weights, EMEA-s1 already leads to significant improvements over ensembling for NER. EMEA-s10 brings additional improvements on both tasks because it learns more optimal ensembling weights with 10 gradient update steps (we list the inference cost for each method in Appendix B). We hypothesize that the proposed methods improve non-Latin languages more because these are low-performing languages that the model is more uncertain about.

Effect of test batch size In Fig. 2 we plot the result of using different test batch sizes with EMEA on the NER task. A smaller batch size leads to more fine-grained test time adaptation with a higher computational cost. Fig. 2 shows that a smaller batch size indeed leads to better performance while using a larger batch size still outperforms the baseline.

Significance of source language adapter We investigate whether the benefit of adding the src language adapter comes from the discrepancy between training and testing of the task adapter. We train
Comparison to training new adapters  In order to better understand how much data is required to train new language adapters that are competitive with EMEA, we trained new adapters using a small amount of monolingual data in the target language. We focus on two languages, mr and no, on the NER task, and show the results in Fig. 4. Note that this setting puts EMEA at a disadvantage because EMEA does not require any training. It takes about 100k monolingual data for no to reach comparable performance with our method, while mr still lags behind EMEA. As large amounts of monolingual data are difficult to obtain for many language varieties and under-represented languages, EMEA can serve as a useful baseline for applying NLP models to such low-resource settings.

Analysis of weights  We plot the mean and standard deviation of ensembling weights from EMEA in Fig. 5. The En adapter gets the highest weight for both language groups, in line with the results in Tab. 2 showing en as the best individual adapter. For the hi language group, the ar adapter tends to have the least benefit, probably because it has a different script from the languages we test on.

5  Related Work

Our work is related to parameter efficient fine-tuning of pretrained models (Bapna et al., 2019; Pfeiffer et al., 2020b; Li and Liang, 2021; Guo et al., 2021). Specifically, (Üstün et al., 2020; Karimi Mahabadi et al., 2021) make adapters more generalizable by learning a parameter generator, while our work aims to utilize existing pretrained adapters without further training. Pfeiffer et al. (2021) propose to learn extra parameters using labeled data to combine pretrained multitask adapters whereas our method does not require any training or labeled data. While we focus on language adapters in this work, our method is also applicable to ensembling domain or task adapters. Finally, our method is inspired by the test time adaptation framework proposed for image classification (Sun et al., 2020; Wang et al., 2021; Kedia and Chinthakindi, 2021). Instead of adapting a single model, we focus on efficient utilization of many pre-trained language adapters to improve the model’s robustness to language variations.

6  Discussion and Conclusion

Language and dialect cannot be simply categorized into monolithic entities. Thus a truly intelligent NLP system should be able to recognize and adapt to personalized language varieties after it is trained and deployed. However, the standard system evaluation is built on the assumption that an NLP model is fixed once it is trained. In this paper, we focus on a specific case of this general problem—we find that specialized language adapters might not be robust to unseen language variations, and that utilization of multiple existing pretrained language adapters alleviates this issue. We hope our findings can inspire future work on models that are robust and adaptive to language variations.

We identify two limitations of this paper, which we leave to future work. First, there are limited datasets and benchmarks that evaluate NLP models’ ability to generalize to unseen dialect variations. Therefore, we only test our method on NER and POS tagging tasks because they have the best language coverage. It is an important future direction to construct high-quality datasets that consider language and dialect variations. Second, our method has slower inference speed due to test time computation. Future work can aim to reduce the cost by algorithmic or hardware innovations.

Acknowledgement

This material is based on work supported by the National Science Foundation under grants 2040926 and 2152501. XW is supported by the Apple PhD fellowship. The authors would like to thank Laura Rimell, Sachin Kumar and Hieu Pham for helpful discussions on the drafts of the paper.
References

Isaac Caswell, Theresa Breiner, Daan van Esch, and Ankur Bapna. 2020. Language ID in the wild: Unexpected challenges on the path to a thousand-language web text corpus. In COLING.


Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhoun. 2019. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. In EMNLP.


Table 3: Decoding speed for different methods used in the paper.

<table>
<thead>
<tr>
<th>Method</th>
<th>Example/Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Adapter</td>
<td>250</td>
</tr>
<tr>
<td>CL</td>
<td>77</td>
</tr>
<tr>
<td>Fusion</td>
<td>200</td>
</tr>
<tr>
<td>Ensemble</td>
<td>200</td>
</tr>
<tr>
<td>EMEA-s1</td>
<td>62</td>
</tr>
<tr>
<td>EMEA-s10</td>
<td>9</td>
</tr>
</tbody>
</table>

A Implementation Details

We preprocess the data using scripts in XTREME (Hu et al., 2020). We use the best performing adapter configuration in Pfeiffer et al. (2020b). For NER, we train the task adapter for 100 epochs using learning rate of 1e-4. For POS tagging, we train the task adapter for 50 epochs with the learning rate of 1e-4. For EMEA, we search over the learning rate $\gamma$ of 0.1, 1, 10 on the English validation set and pick $\gamma = 10$ for all experiments.

For Fusion, we use learning rate of 5e-5 which is recommended by (Pfeiffer et al., 2021). We search over the best learning rate for CL on the performance of English labeled data. We use the learning rate of 2e-5 and do 1 step of gradient update for each batch.

For our experiment on training new adapters, we find that training from scratch on no and mr is not competitive when using very small amount of data. Therefore, we continue training from their related language adapters.

B Decoding Speed

We list the inference time for various methods in the paper in Tab. 3. EMEA leads to better performance at a cost of lower inference speed. We leave it to future work to explore strategies that speed up the test time optimization.

C Examples of outputs

We compare the outputs of EMEA with the best baseline on the POS tagging task for Norwegian (no). Although both methods struggle with verb and adjective predictions, EMEA is often better at predicting the correct adjectives compared to the baseline.