Zero-shot Cross-lingual Transfer With Learned Projections Using Unlabeled Target-Language Data

Ujan Deb[*]
IIT Bhilai
ujand@iitbhilai.ac.in

Ridayesh Parab[*]
IIT Bombay
ridayesh@gmail.com

Preethi Jyothi
IIT Bombay
pjyothi@cse.iitb.ac.in

Abstract

Adapters have emerged as a parameter-efficient Transformer-based framework for cross-lingual transfer by inserting lightweight language-specific modules (language adapters) and task-specific modules (task adapters) within pretrained multilingual models. Zero-shot transfer is enabled by pairing the language adapter in the target language with an appropriate task adapter in a source language. If our target languages are known apriori, we explore how zero-shot transfer can be further improved within the adapter framework by utilizing unlabeled text during task-specific finetuning. We construct language-specific subspaces using standard linear algebra constructs and selectively project source-language representations into the target language subspace during task-specific finetuning using two schemes. Our experiments on three cross-lingual tasks, Named Entity Recognition (NER), Question Answering (QA) and Natural Language Inference (NLI) yield consistent benefits compared to adapter baselines over a wide variety of target languages with up to 11% relative improvement in NER, 2% relative improvement in QA and 5% relative improvement in NLI.

1 Introduction

Zero-shot cross-lingual transfer refers to the transfer of task-specific knowledge from a (high-resource) source language to a (zero-resource) target language that has no labeled task-specific data for training. A popular paradigm for cross-lingual transfer learning is to finetune pretrained multilingual models using labeled task-specific data in the source language and directly evaluate these finetuned models on target language test sets.

A parameter-efficient alternative to full finetuning for cross-lingual transfer is MAD-X (Pfeiffer et al., 2020b), an adapter-based framework that scaffolds on multilingual pretrained models to combine task-specific and language-specific modules in a plug-and-play manner. Adapters (Houlsby et al., 2019) are feedforward layer blocks inserted within each Transformer layer to selectively learn task-specific and language-specific capabilities via task adapters and language adapters, respectively. Language adapters are trained using self-supervised objectives like masked language modeling (MLM) and task adapters are trained using task-specific objectives. To enable task transfer to a target language, the relevant language and task adapters are combined at test-time.

In the zero-shot setting, we assume access to unlabeled text in the target languages. In MAD-X, this text is only used to train target language adapters and not further used during finetuning. Given knowledge of which languages we want to target, can we make effective use of unlabeled text in the target languages even during task-specific finetuning? This is the main question we tackle in this work.

We propose a general adapter-based technique to inject target language bias into task-specific finetuning. Using the unlabeled text in each target language, we construct an affine subspace from contextualized representations for every Transformer layer in the multilingual model. These subspaces are defined using singular value decomposition (SVD) and only need to be computed once per target language. During task-specific finetuning using labeled data in the source language, we project the source representations onto the target language subspaces. This projection can be invoked randomly using a projection probability defined as a hyperparameter. Projections can also be triggered depending on whether the current source representations are closer to the mean embedding of the source language subspace compared to the mean embedding of the target language subspace. We investigate both these projection policies and find

[*]Equal contribution

that they both improve performance across multiple tasks in multiple languages compared to state-of-the-art adapter baselines. We also release code[1] to reproduce our experiments.

2 Methodology

Adapters and MAD-X. Adapters for language models (Houlsby et al., 2019) are bottleneck feed-forward modules, typically inserted in each Transformer layer of a multilingual model before layer normalization. Instead of finetuning the entire model, only adapters are tuned for a specific task. Pfeiffer et al. (2020b) extended adapter-based fine tuning to support cross-lingual transfer. Their framework called MAD-X (Multiple Adapters for Cross-lingual transfer) comprises of language adapters and task adapters. Language adapters are pretrained using masked language modeling to learn language-specific features. Task adapters are stacked on top of language adapters during downstream task finetuning to learn task-specific information. To achieve zero-shot transfer, the model is trained with a frozen source-language language adapter and a task adapter. During test time, the source-language adapter is replaced with the target-language adapter and evaluated on test instances in the target language.

Overview of our technique. We are interested in the setting where we have apriori knowledge of which languages we want to target at test time. We aim to bias cross-lingual transfer towards known target languages during task-specific finetuning. We start with MAD-X as our underlying framework and adopt the following 3-step approach:

- We construct layer-specific subspaces for each of the target languages. This is done by computing SVD on contextualized token representations extracted from each layer. See §2.1 for more details.
- During task-specific training, we selectively project output representations from the language adapter of a chosen layer onto the target language subspace. These projections are triggered based on two policies: Random projection (§2.2) and Mean Cosine Distance (§2.3). The projected representations are further passed through the task adapter that is trained using labeled data in the source language.
- Similar to MAD-X, we evaluate on the target language by simply swapping the source language adapter with the target language adapter while keeping the task adapter fixed. No projection is done during inference.

[1] https://github.com/csalt-research/adapter-projections

2.1 Language Subspaces and Projections

Our objective is to bias the model towards the target language while fine-tuning for a task. For this, we need to extract language-specific information from model representations that jointly exhibit language-specific and language-independent properties. Language-specific subspaces have been typically used to analyze representations in multilingual language models. Choenni and Shutova (2020) showed that individual representations can be used to predict linguistic typological features after projecting onto language-sensitive subspaces. Chang et al. (2022) construct language subspaces with SVD using language-specific contextualized token embeddings. They analyze model performance and other properties after computing layer-wise projections of representations to various language subspaces.

We construct subspaces for each of the target languages using SVD and contextualized token representations for unlabeled text in the target language. Consider a pretrained model like XLMR (Conneau et al., 2020) that takes text sequences from the target language as its input. d-dimensional embeddings from a particular layer for a given language A can be grouped into a matrix $\mathbf{M}_A \in R^{n \times d}$. SVD of $\mathbf{M}_A$ (after subtracting the mean representation for A) can be written as: $\mathbf{M}_A = \mathbf{U}_A \Sigma \mathbf{V}_A^T$. The right singular matrix $\mathbf{V}_A$ is considered to be the subspace for language A. These subspaces only need to be computed once for each layer. Next, we look at when projections should be invoked.

2.2 Random Projection

For a given target language, during finetuning using task-specific data in the source language, we project the source representations onto the target language subspace with a predetermined probability p. This projection is invoked right before passing the representation through the task adapter, having already passed through the language adapter. To project onto a target subspace, we first shift the target language subspace so that it passes through the source language mean embedding and then take the projection onto the target subspace (Chang et al., 2022). Let S be the source language and Q be the target language. Let subspaces and means of representations from one of the Transformer layers

for the source language be $\mathbf{V}_S$ and $\boldsymbol{\mu}_S$, respectively. Projection of a representation $\mathbf{x}$ on S is given by:

$$\text{Project}_S(\mathbf{x}) = \mathbf{V}_S\mathbf{V}_S^T(\mathbf{x} - \boldsymbol{\mu}_S) + \boldsymbol{\mu}_S$$

The projection of $\mathbf{x}$ onto the target language subspace, that is shifted onto the source subspace, can be computed as:

$$\text{Project}_{Q,\boldsymbol{\mu}_S}(\mathbf{x}) = \mathbf{V}_Q\mathbf{V}_Q^T(\mathbf{x} - \boldsymbol{\mu}_S) + \boldsymbol{\mu}_S$$

The main intuition here is that by probabilistically projecting source representations onto the target language subspace during task-specific fine-tuning, the model can encode both source and target language information in its representations. The model cannot solely rely on source-language specific features during task-specific training.

2.3 Mean Cosine Distance (MCD)

We suggest another projection scheme, Mean Cosine Distance (MCD), that is more informed than randomly projecting source representations based on a projection probability p. Using MCD, we project those embeddings that are deemed as being further away from the target language subspace compared to the source language subspace. This is quantified using a cosine distance between an embedding from a layer and means of source and target language subspaces. If an embedding is closer to the source language mean compared to the target language mean, we project it onto the target language subspace so as to make it more similar to target language embeddings. However, if an embedding is closer to the target language mean, we can possibly omit projection since it already contains information relevant to the target language.

Consider a set of embeddings extracted from one of the Transformer layers. Let the means of all embeddings from this layer and the associated subspace be denoted by $\boldsymbol{\mu}$ and $\mathbf{V}$, respectively. $\boldsymbol{\mu}_S$ and $\boldsymbol{\mu}_Q$ denote the means for the source and target language, respectively. Similarly, $\mathbf{V}_S$ and $\mathbf{V}_Q$ refer to the respective subspaces. Let $\mathbf{x}$ denote a token embedding from the source language. The MCD policy can be written as:

$$\mathbf{x} = \begin{cases} \text{Project}_{Q,\boldsymbol{\mu}_S}(\mathbf{x}) & \text{if } \text{c}(\mathbf{x}, \boldsymbol{\mu}_Q) < \text{c}(\mathbf{x}, \boldsymbol{\mu}_S) \\ \mathbf{x} & \text{otherwise} \end{cases}$$

where $\text{Project}_{Q,\mu_S}(\mathbf{x})$ is defined in Section 2.2 as the projection of $\mathbf{x}$ onto the target subspace $\mathbf{V}_Q$ and $\text{c}(\mathbf{x}, \mathbf{y})$ refers to the cosine similarity between two embeddings $\mathbf{x}$ and $\mathbf{y}$.

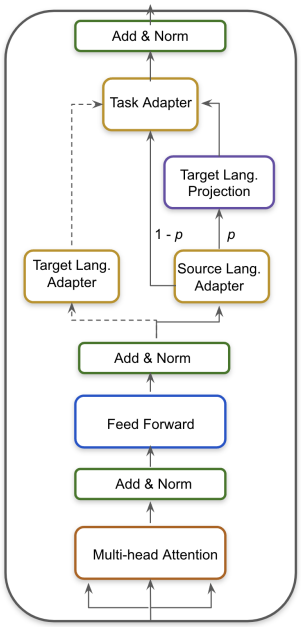

Figure 1: A single Transformer layer as modified by the MAD-X setup and our projection scheme. During training, the output from the source language adapter is projected onto the target language subspace with probability p for random projection (or, if deemed necessary, by the MCD scheme). Dotted arrows refer to the inference time pathway when representations pass through the target language adapter and no projection is applied.

Figure 1 provides an illustration of our proposed technique within a single Transformer layer that includes language and task adapters (as in the MAD-X framework).

3 Experimental Setup

Subspace construction. To construct language specific subspaces, we adopt the settings used by Chang et al. (2022). Text sequences of length 512 are taken from the OSCAR dataset (Ortiz Su'arez et al., 2019) and passed through XLMR (Conneau et al., 2020) to produce layer-wise contextualized embeddings. We pick 262K contextualized representations and subtract the representation mean before computing SVD. For a low-dimensional subspace, we select the greatest k singular values such that their sum of squares is greater than or equal to 90% of the total variance. (Total variance is given by the sum of the squared singular values produced.) Finally, in order to compute the language-specific subspaces, the corresponding right singular vectors are taken as the basis.

	NER									
	hi	vi	de	id	is	ilo	sw	my	jv	avg
MAD-X Adapters	68.3	66.8	75.9	49.4	76.2	74.0	74.8	52.7	57.3	66.1
Random Projection	**68.9**	**69.0**	**77.5**	53.8	**76.8**	**79.8**	**76.5**	**57.6**	**61.2**	**69.0**
MCD	68.5	68.1	77.1	**54.7**	76.1	76.9	75.4	53.6	59.3	67.7

Table 1: NER results (F1 scores) for nine languages.

	XQuAD			
	hi	vi	de	avg
MAD-X Adapters	68.1	71.4	71.8	70.4
Random Projection	68.2	72.2	72.2	70.9
MCD	**68.6**	**72.9**	**73.5**	**71.7**

Table 2: Results (F1) for QA for three languages

	NLI		
	qu	gn	avg
MAD-X Adapters	48.2	36.0	42.1
Random Projection	**49.3**	37.5	**43.4**
MCD	48.1	**37.8**	42.9

Table 3: Results (F1) for NLI for two languages

Datasets. We conduct cross-lingual transfer experiments on three tasks, Named Entity Recognition (NER), Question Answering (QA) and Natural Language Inference (NLI), where the source language is always English. For NER, we use the WikiANN dataset (Rahimi et al., 2019), and show results for nine languages Hindi, Vietnamese, German, Indonesian, Icelandic, Ilocano, Swahili, Burmese and Javanese with roughly 20K instances in the English train set and between 1K and 10K instances in the target dev and test sets. For QA, we use XQuAD (Artetxe et al., 2019), a multilingual extension of SQuAD (Rajpurkar et al., 2016) and we report results for Hindi, Vietnamese and German consisting of around 87K examples in the English SQuAD train set and 1190 instances in the three target dev sets. For NLI, we use the AmericasNLI dataset (Ebrahimi et al., 2021) which is an extension of the XNLI dataset (Conneau et al., 2018) with low-resource American languages. We report results on Quechua and Guarani, consisting of 392k instances in the English train set and 2490 and 5010 instances in the dev and test sets, respectively for each target language.

Training setup. We use transformer models from the adapter-transformers[2] fork of the HuggingFace transformers library (Wolf et al., 2020). We use pre-trained language adapters from AdapterHub (Pfeiffer et al., 2020a) for our transfer experiments. XQuAD and NLI fine-tuning experiments were conducted on a single NVIDIA A100 80 GB gpu for 15 epochs and 10 epochs, with learning rate 1e-4 and batch size 16. NER experiments were run for 30 epochs on Nvidia 1080 Ti with 12 GB ram. Further details can be found in Appendix A.

[2] https://github.com/adapter-hub/adapter-transformers

4 Results

NER, XQuAD and NLI results are shown in Table 1, Table 2 and Table 3 respectively. All values correspond to F1 scores averaged over 3 different seeds. We use the target language validation set to choose the best hyperparameter values for all experiments. Both MCD and random projections show consistent improvement over the MAD-X baseline numbers. With MCD, we explicitly instruct the model when to project. This removes a hyperparameter from the setup, compared to random projections, while maintaining consistent performance gains over the baseline. To further analyze MCD,

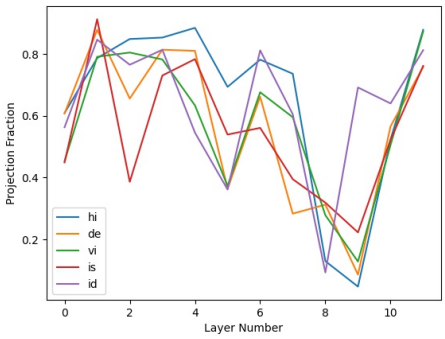

Figure 2: Projection fraction vs layer across epochs.

we consider the fraction of embeddings being projected onto the target language subspace for NER. Table 4 shows the fraction of embeddings projected during training (averaged across all layers) for each language. For languages dissimilar to en (such as hi and id), it makes sense that the projection fractions

Table 4: Projection percentages for NER.

	hi	vi	de	id	is
Proj. Frac.	0.65	0.57	0.57	0.63	0.55

are high since the language subspace means are closer to the source language mean (Chang et al., 2022), compared to languages more similar to en like de and is. Figure 2 shows how projection fractions vary across layers averaged across training epochs. We see high projection rates in early and final layers across languages. This correlates with these layers encoding a lot of English-specific information (Rogers et al., 2020) via training on the task-specific English data, thus triggering projections via MCD often.

5 Related Work

Multilingual language models like mBERT (Devlin, 2018), XLM-R (Conneau et al., 2020) possess some zero-shot cross-lingual capabilities, even without any explicit finetuning on the languages of interest (Wu and Dredze, 2019; Pires et al., 2019). Such transfer without any finetuning could lead to degradation in performance across certain language pairs (Hu et al., 2020). Nevertheless, multilingual models are a good foundation to bootstrap and further develop cross-lingual generalization. While there is a rapidly growing body of work on cross-lingual transfer, very few approaches utilize language-specific subspaces for this purpose. Both Choenni and Shutova (2020) and Chang et al. (2022) construct language-specific subspaces in multilingual models for an exploratory analysis of the model's representations. Yang et al. (2021) use projections on language specific subspaces to remove language specific information from the representations. We note such removal of language bias did not perform well on cross-lingual transfer in our experiments. Parović et al. (2022) train bilingual language adapters using both source and target language text before task adapter training. However, this requires training language adapters using both source and target language unlabelled text, for every language pair, in addition to training task adapters. In contrast, our setup is a simple architectural extension of MAD-X, requiring no additional training once the subspaces are computed for each language. To the best of our knowledge, ours is the first work to exploit language-specific subspaces for cross-lingual transfer.

6 Conclusions

In this work, we present a new adapter-based cross-lingual transfer technique for an apriori known set of target languages. We construct language subspaces using contextualized representations for source and target languages. Representations during task-specific training are projected onto the target subspace if they exceed a probability threshold or if they are closer to a mean source embedding. Both schemes consistently improve zero-shot transfer for three natural language understanding tasks across many languages.

Acknowledgements

The first author (Ujan) was supported by the Uplink Internship Program of the India Chapter of ACM SIGKDD. The authors are thankful to the anonymous reviewers for their constructive suggestions that helped improve this submission.

Limitations

While our proposed projection techniques often improve cross-lingual transfer, the choice of the projection layer and the projection probability in the case of random projection are hyperparameters that vary across tasks and languages. Our ongoing work involves identifying a mechanism via which we can parameterize these quantities, enabling the model to directly learn the optimal layer and probability values for projection.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2019. On the cross-lingual transferability of monolingual representations. *CoRR*, abs/1910.11856.

Tyler A. Chang, Zhuowen Tu, and Benjamin K. Bergen. 2022. The geometry of multilingual language model representations. arXiv:2205.10964.

Rochelle Choenni and Ekaterina Shutova. 2020. What does it mean to be language-agnostic? probing multilingual sentence encoders for typological properties. arXiv:2009.12862.

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. arXiv:1911.02116.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk,

and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Jacob Devlin. 2018. Multilingual bert readme document.

Abteen Ebrahimi, Manuel Mager, Arturo Oncevay, Vishrav Chaudhary, Luis Chiruzzo, Angela Fan, John Ortega, Ricardo Ramos, Annette Rios, Ivan Vladimir, Gustavo A. Giménez-Lugo, Elisabeth Mager, Graham Neubig, Alexis Palmer, Rolando A. Coto Solano, Ngoc Thang Vu, and Katharina Kann. 2021. Americasnli: Evaluating zero-shot natural language understanding of pretrained multilingual models in truly low-resource languages. *CoRR*, abs/2104.08726.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. arXiv:1902.00751.

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. arXiv:2003.11080.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, and Nicolas Patry. 2020. Datasets: A community library for natural language processing. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pedro Javier Ortiz Su'arez, Benoit Sagot, and Laurent Romary. 2019. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019, pages 9 – 16, Mannheim. Leibniz-Institut f"ur Deutsche Sprache.

Marinela Parović, Goran Glavaš, Ivan Vulić, and Anna Korhonen. 2022. Bad-x: Bilingual adapters improve zero-shot cross-lingual transfer. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1791–1799, Seattle, United States. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. arXiv:2005.00052.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, page arXiv:1606.05250.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

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

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).

Ziyi Yang, Yinfei Yang, Daniel Cer, and Eric Darve. 2021. A simple and effective method to eliminate the self language bias in multilingual representations. arXiv:2109.04727.

A Implementation Details

We use the xlm-roberta-base model from HuggingFace Transformers (Wolf et al., 2020) pretrained on 2.5 TB of CommonCrawl data[3], for all our experiments. NLI and XQuAD experiments were conducted on a single NVIDIA A100 GPU (80 GB

[3] https://commoncrawl.org/

Table 5: For random projection, best-performing projection layers for different languages obtained via a grid search on validation sets.

	NER									XQuAD			NLI	
	hi	vi	de	id	is	sw	ilo	jv	my	hi	vi	de	qu	gn
Random Projections	5	6	8	4	8	6	6	8	8	0	1	2	9	1
MCD	10	2	8	4	0	6	0	0	7	9	2	9	11	11

Table 6: Probability values (as determined by tuning on validation sets) for the layers in Table 5.

	NER									XQuAD			NLI	
	hi	vi	de	id	is	sw	ilo	jv	my	hi	vi	de	qu	gn
Random Projections	0.1	0.3	0.3	0.9	0.5	0.5	0.3	0.5	0.5	0.5	0.7	0.5	0.1	0.1

RAM) and the NER experiments ran on a single Nvidia 1080Ti GPU (12 GB RAM). We used a learning rate of 1e-4 with a batch size of 16. The hyperparameter choices for layers and probabilities for our experiments are given in Tables 5 and 6, respectively.

All datasets used are taken from HuggingFace Datasets (Lhoest et al., 2020). For evaluating models, we use the HuggingFace Evaluate library[4] as well as the seqeval python package[5]

[4] https://huggingface.co/docs/evaluate/index

[5] https://pypi.org/project/seqeval/