FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

UNIVERSITÄT
DES
SAARLANDES

MASTER THESIS

Jacobo Del Valle

Analyzing Modular Cross-Lingual Transfer Learning

Institute of Formal and Applied Linguistics, Charles University
Department of Computational Linguistics and Phonetics, Saarland University

Supervisors of the master thesis: Jindřich Libovický
Marius Mosbach
Mareike Hartmann
Dietrich Klakow

Study programme: Language Technologies and Computational Linguistics, Charles University,
Language Science and Technology, Saarland University

Prague 2023

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

This work would not have been possible without the financial and otherwise help of the LCT program and its scholarship program. Thank you to the administrators and local coordinators, Anna Felsing, Markéta Lopatková, Jürgen Trouvain, Ivana Krujiff-Korbayová, and Tanja Bäumel.

I want to thank my supervisors and advisors Dietrich Klakow (UdS), Jindřich Libovický (CUNI), Mareike Hartmann (UdS), Marius Mosbach (UdS), Julius Steuer (UdS), and everyone who helped me review this work. Thank you for the insightful feedback and endless conversations.

I would also like to thank my internship mentors Yannick Versley, Shailza Jolly, Andy Rosenbaum, and Swetlana Fischl-Schuster. I learned a great deal from all of you.

A very special thanks to my wife Katherine, who kept me going and supported me through this exciting, and sometimes stressful, journey in more ways than I could have hoped for. I could not have done it without her.

I also thank my family who has cheered me on from afar.

Thanks to the friends and colleagues I have made along the way and who have helped me in all sorts of ways: Kirill, Zhenya, Kate, Kuzya, Chris, Adele, Aditya, Vilém, and everyone who supported me.

Title: Analyzing Modular Cross-Lingual Transfer Learning

Author: Jacobo Del Valle[†‡]

Supervisors:
Mgr. Jindřich Libovický, Ph.D.[†]
Mgr. Marius Mosbach[‡]
Dr. Mareike Hartmann[‡]
Prof. Dr. Dietrich Klakow[‡]

Institutes:
[†]Faculty of Mathematics and Physics: Institute of Formal and Applied Linguistics, Charles University,
[‡]Department of Computational Linguistics and Phonetics, Saarland University

Abstract: Cross-lingual abilities have been evident in large multilingual language models over the past few years. However, understanding why and under what circumstances they work is not entirely clear. In this work, we work towards a better understanding of these aspects in a specific subset of multilingual models, namely modular multilingual models with cross-lingual transfer learning abilities. We try to quantify claims in Pfeiffer et al. [2022] regarding their proposed model, X-MOD, as it was tested in a very specific setting which may not align with common low-resource settings. Specifically, we evaluate how the following factors may affect downstream performance: the amount of available pre-training data; hyperparameters such as number of training steps, checkpoint selection criteria, available overlapping lexicon.

With the help of our findings, we also aim to provide guidelines on how to best use X-MOD, especially from a low-resource perspective.

Keywords: transfer learning cross-lingual learning low-resource language models

Contents

Introduction

Pre-training language models [Howard and Ruder, 2018, Peters et al., 2018, Devlin et al., 2019] has resulted in a change of paradigm in the world of NLP. The prevalent strategy to achieve good performance on downstream tasks is now to use a pre-trained model and fine-tune it on the downstream task data [Devlin et al., 2019, Howard and Ruder, 2018]. Furthermore, if the pre-training data is modified to have data from various languages, the models are capable of achieving surprising cross-lingual learning abilities [Conneau et al., 2020, Conneau and Lample, 2019]. This can happen without having to explicitly provide cross-lingual learning signals, i.e., there is no need for parallel data. However, there are certain conditions that need to be met in order for these cross-lingual learning abilities to arise [Wang et al., Dufter and Schütze, 2020]. Among them, a non-overparameterized model, overlap in vocabulary and/or syntax similarities in the multiple languages. Finally, performance of the model in downstream tasks can degrade as more languages are added to the model [Conneau et al., 2020], this phenomenon was coined the *curse of multilinguality*.

Considering the above, Pfeiffer et al. [2022] propose an architecture, which relies on modularity of languages making use of adapters [Houlsby et al., 2019], as a solution to the curse of multilinguality; they call this approach X-MOD. This architecture promotes the arising of shared representations as it adds language-dependent adapters at each layer, which should transform the output of each layer to be specific to that of the relevant language. The authors propose a method to seamlessly add languages to an original pre-training of X-MOD. One of the main claims in the work is that it makes little difference in downstream performance whether a language was part of the original pre-training or whether it was added post-hoc. Moreover, the authors also claim that the approach has the potential to cover all languages in the world.

In this work, we aim to analyze these claims and try to identify crucial conditions under which they hold. We do this by, firstly, replicating the experiments presented in the paper. We also simulate low-resource language scenarios and try to ascertain whether/where claims break down. Specifically, we:

- Limit the number of training steps in the language adaptation phase
- Artificially limit the overlapping vocabulary in languages
- Restrict available monolingual data for languages to be *added*

We find that the number of steps and available data are important conditions that, if not met, could break down the claims in the original work.

Finally, we test the model on a real so-called low-resource languages and test their performance when compared to available baselines for zero-shot cross-lingual transfer of knowledge on these low resource languages.

Our main objective is to provide guidelines for making the best use out of X-MOD when confined to a low-resource scenario, which tends to be the case for the majority of languages spoken in the world, since research is only carried out in a few languages for which resources are readily available [Joshi et al., 2020]. A secondary objective is to move away from square one research Ruder et al. [2022], i.e., focusing only on one aspect of performance, fairness, and computational efficiency. This work aims to focus on all aspects and tries to pave the way for future research to do the same.

Finally, we also reanalyze criteria for determining whether a language is related to another, as we feel the original publication could be improved on this analysis. We leave the results of this analysis as preliminary, and they will be refined in future work.

Chapter 1

Background and Related Work

In this chapter, we provide the necessary background in order to understand and motivate the contents of this thesis. In section 1.1, we cover language models, their evolution to neural approaches, and transfer learning. In section 1.4, we discuss multilingual language models and their development over recent time. In section 1.5, we discuss the phenomenon of cross-lingual transfer learning. In section 1.6, we cover strategies on how to efficiently, but still effectively, train modern language models. Finally, in section 1.7, we elaborate on modular multilingual models, and we motivate why they are the focus of this work.

1.1 Language Modeling

Language models (LMs) have, in recent years, become the base model from which a lot of natural language processing (NLP) solutions depart from. Language modeling as a task has evolved over time in its approaches and solutions. In this section, we discuss such developments and what consequences they have had on the field.

1.1.1 Language Models

The language modeling task, in principle, assigning a probability to a sequence of words [Manning and Schutze, 1999]. The task is classically presented as the prediction of the next word given the previous words, the *history*, i.e., estimate the probability distribution:

$$P(w_t|w_{t-1}, \ldots, w_1)$$

Actual approaches of estimating these probability functions rely on estimates derived from the maximum likelihood estimate from an often large dataset using counts:

$$P(w_t, \ldots, w_1) = \frac{C(w_t, \ldots, w_1)}{N} \tag{1.1}$$

$$P(w_n|w_{n-1}, \ldots, w_1) = \frac{C(w_1, \ldots, w_n)}{C(w_1, \ldots, w_{n-1})} \tag{1.2}$$

Where N represents the number of training examples, C is a function that represents the frequency of the word sequence that is passed as an argument. This frequency corresponds to the frequency in the dataset used for estimating our probability functions.

However, there is no straight-forward way of capturing the information for all possible histories. In fact, it is impossible in practice, as language constantly evolves. Classically, a *Markov assumption* was made, which allows models to only take into account a local history as opposed to a complete one. This results in *n-gram* language modeling. Where n is usually in the order of units, i.e., a very local history. This means that the above equations would change to reflect this local history:

$$P(w_t, \ldots, w_1) \approx P(w_t, \ldots, w_{t-n+1}) = \frac{C(w_{t-1}, \ldots, w_{t-n+1})}{\#\text{ of n-grams}} \tag{1.3}$$

$$P(w_t|w_{t-1}, \ldots, w_1) \approx P(w_t|w_{t-1}, \ldots, w_{t-n+1}) = \frac{C(w_t, \ldots, w_{t-n+1})}{C(w_{t-1}, \ldots, w_{t-n+1})} \tag{1.4}$$

Other approaches of limiting the space that needs to be modeled, is by attributing every word a class. This results in the space being reduced, assuming of course that the number of classes selected is lower than the vocabulary size of the language being modeled.

These models, although powerful, have a serious limitation in the form of what is called **the curse of dimensionality**: in order to be able to generalize, the model needs to grow in the number of sequences it has seen. This in turn will increase the model size and training time significantly.

Further considerations need to be made for this approach to work in practice, such as smoothing, setting a vocabulary, limiting the corpus to said vocabulary, etc. Furthermore, its limitations are obvious in terms of capabilities, as data sparsity is a real issue Manning and Schutze [1999].

1.1.2 Evaluating Language Models

When working with language models, it is often necessary to be able to evaluate them on their intrinsic ability to solve the task, without evaluating a downstream task Manning and Schutze [1999].

In most scenarios, cross-entropy or perplexity is used to evaluate language models. Perplexity is usually defined as follows:

$$\text{perplexity}(X, m) = b^{H(X,m)} \tag{1.5}$$

Where X is a distribution of tokens, b is the base we are working on (usually 2 or e), m is the model we are evaluating and H is the cross-entropy between the empirical distribution and the distribution learned by the model.

1.1.3 Neural Language Modeling

Bengio et al. [2003] propose to learn a distributed representation of words, such that knowledge gained from each training also extends to semantically similar sequences. This addresses one of the main drawbacks of classical language models

based on n-grams; the curse of dimensionality, which was discussed in the previous section. This new shared representation approach is the basis for modern language models. The neural language model learns a dense representation for each word while also learning the probability function for sequences Bengio et al. [2003].

Though there are many ways to formulate the language modeling problem, the formulation does not change from what we've described in the previous section. We will call this specific formulation of the task causal language modeling (CLM).

The dense representations are learned in a completely different way than classical language models. Equation 1.6 describes how the prediction for a word w_t is obtained from the network using a softmax operation. Equation 1.7 describes the operations performed by the network to arrive at each y_i in equation 1.6.

$$\hat{P}(w_t|w_{t-1},\ldots,w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}} \tag{1.6}$$

$$y = b + Wx + U\tanh(d + Hx) \tag{1.7}$$

The neural network is trained by maximizing the log-likelihood of the data using a gradient descent[1] approach by implementing a forward and a backward pass.

$$\theta \leftarrow \theta + \epsilon \frac{\partial \log \hat{P}(w_t|w_{t-1},\ldots,w_{t-n+1})}{\partial \theta} \tag{1.8}$$

$$L = \frac{1}{T}\sum_t \log f(w_t, w_{t-1},\ldots,w_{t-n+1};\theta) + R(\theta) \tag{1.9}$$

Where ϵ is the learning rate, or step size, in the gradient descent algorithm; L is our log-likelihood; θ is the set of parameters of the network; f is the network itself; and $R(\theta)$ is a regularization term over the parameters.

This fresh approach is not without issues. As with any modern neural approach, the model needs large amounts of data. Processing times were therefore intractable when this approach was proposed. Furthermore, the training process, iterative in nature, requires significant amounts of computational power and time. Part of the main focus in the original publication is on training efficiency due to these factors.

1.2 The Transformer Architecture

was first proposed in Vaswani et al. [2017]. Previous to it, recurrent neural networks (RNNs). In particular, LSTMs [Hochreiter and Schmidhuber, 1997] were used to model examples with a temporal aspect to them, i.e., sequence modeling. The main drawback of RNNs is their need of sequentially processing inputs in order to go through the training phases. This disallows parallelization at training time. The transformer architecture, as seen in Figure 1.1, addresses this by being able to process the input sequence at once using only an attention

[1] More recent formulations minimize the negative log-likelihood which has a direct correspondence.

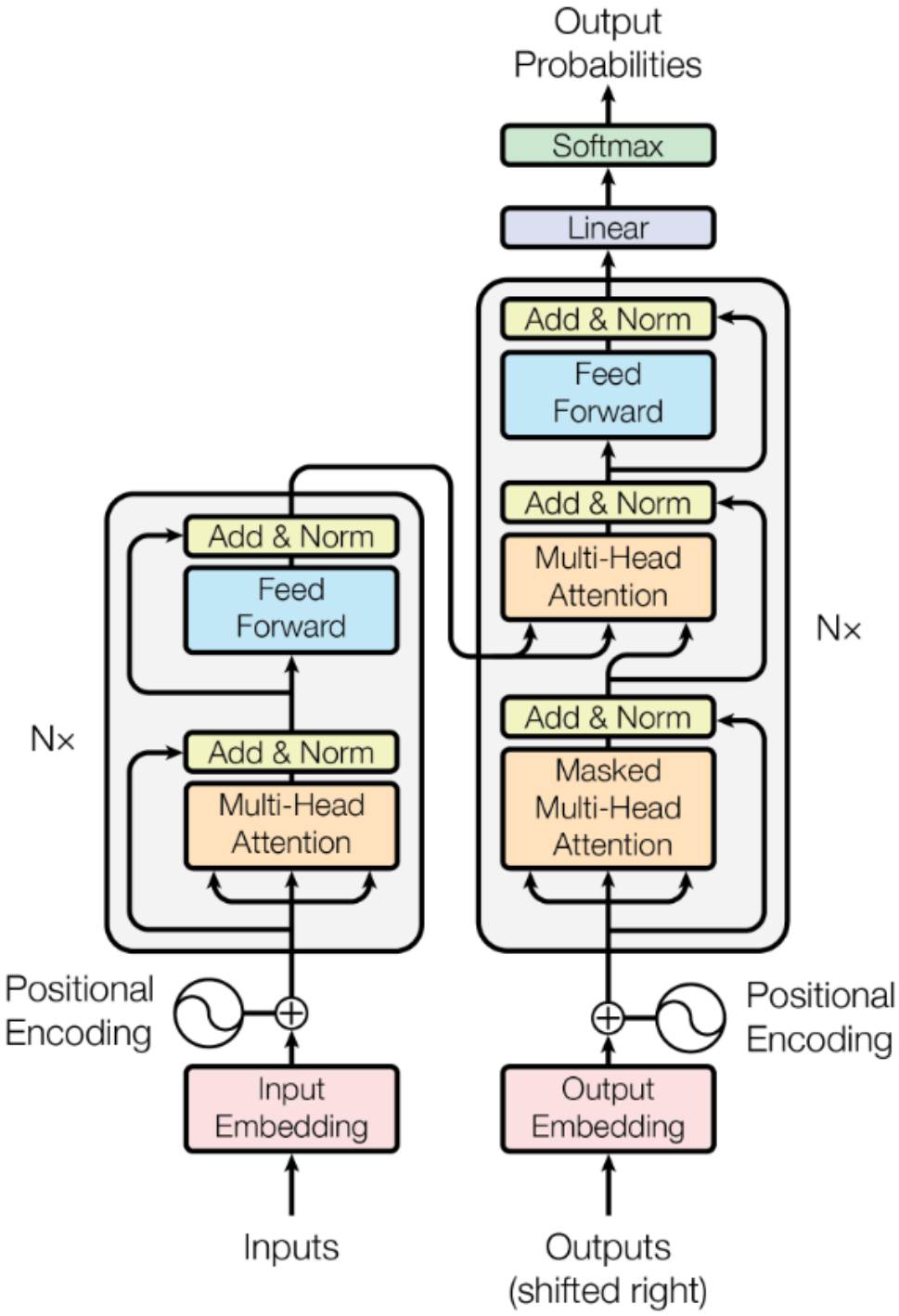

Figure 1.1: The transformer architecture as depicted in the original publication, reprinted from Vaswani et al. [2017].

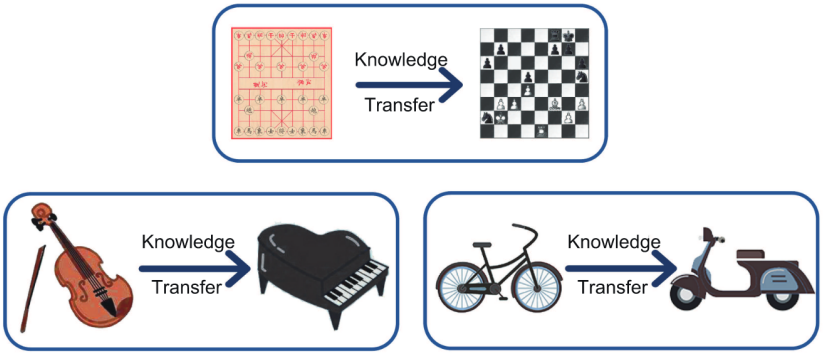

Figure 1.2: Examples of intuitive transfer learning, reprinted from Zhuang et al. [2020]. Note that domains are related or similar in nature.

[Bahdanau et al., 2014] mechanism. The authors call this approach self-attention, as any word can attend to any other word in the input in just one time-step. The dot product attention mechanism the author's use for their self-attention mechanism is formally represented as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.10}$$

where Q represents a set of queries grouped in a matrix, K and V represent keys and values, and $\sqrt{d_k}$ is a normalization constant that depends on the dimensionality of the model.

Originally, the transformer was a proposed solution for a machine translation problem, i.e., a sequence to sequence problem. Modern language models however may consist of only encoder layer blocks, only decoder layer blocks or both.

1.3 Pre-training and Transfer Learning

As stated in the previous section, compute power and time constraints were a major hurdle for dense neural approaches in the language modeling task. In recent years, inspired by the success of transfer learning in the computer vision field, work arose that attempted to pre-train a model in order to be used as a "base" model for future downstream tasks Howard and Ruder [2018], Peters et al. [2018], Devlin et al. [2019]. In this section, we will elaborate on this technique and how it has evolved.

Transfer learning is the process in which knowledge from a model in a, usually, general domain is transferred to another learner in a different, often, more specific domain. Figure 1.2 shows some examples of intuitive knowledge transfer learning scenarios. Formally, Zhuang et al. [2020] define transfer learning as leveraging information from a source domain, given $m^S \in \mathbb{N}^+$ examples from the source domain, $\mathcal{D}^S$, to improve the performance in a target domain decision-making, $f^{T_j}(j = 1, \ldots, m^T)$, given some $m^T \in \mathbb{N}^+$ examples from the target domain. This formalization allows us to clarify some important points: Firstly, it is not necessary for tasks to be related, this is merely a desire as it would most likely lead to better leveraging of the knowledge from the source task. Secondly,

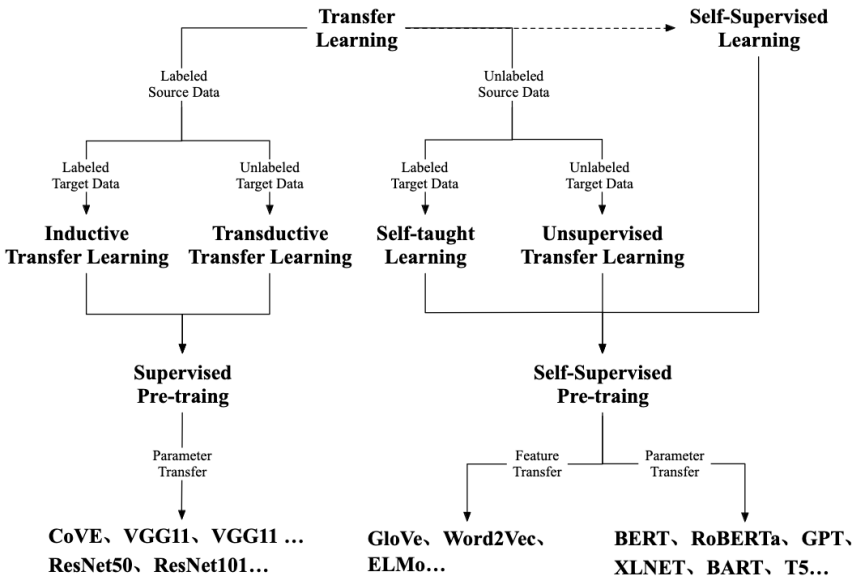

Figure 1.3: The various ways in which transfer learning can happen across different domains and training techniques. Reprinted from Han et al. [2021].

there are no explicit restrictions on the number of examples for each task, which allows us to be able to leverage a large m^S, i.e., number of examples from the source task, while potentially only having access to a low m^T number.

Pre-training is the process of training a model, from a random state, on a general task with the aim of being able to transfer the knowledge learned from the source task to other tasks in potentially other domains, i.e., downstream tasks. The source task and training process can vary depending on the domain, e.g., training an image classifier on thousands of classes for a computer vision domain, or training a language model on large amounts of text for a natural language processing domain. These examples are the most common choices for the domains mentioned. The main goal of pre-training is to obtain a model that has acquired knowledge that is not task specific. For instance, in computer vision, early layers contain information on linear filters or color blobs Yosinski et al. [2014]. In NLP, earlier layers have information on syntactical structure Rogers et al. [2021]. These features are not task specific, but rather general to the domain. After obtaining the pre-trained model, transfer learning is usually accomplished by fine-tuning the model. The general knowledge should ideally be preserved. Figure 1.3 shows the many paradigms under which transfer learning is possible, naming specific examples for each of the approaches. It is important to note that the pre-training phase can differ in its type. In the NLP field, a self-supervised training objective is the common choice, as depicted in the figure.

BERT The transformer architecture, along with increasing computation capabilities, have contributed to a dramatic trend in NLP. Modern language models are trained with a transformer architecture [Devlin et al., 2019, Liu et al., 2019, 2020b, Brown et al., 2020, inter alia]. Among them, BERT [Devlin et al., 2019] has inspired a lot of analysis and has been consistently built upon.

[Devlin et al., 2019] is an encoder only transformer model trained on a masked language modeling task. Masked language modeling differs from causal language

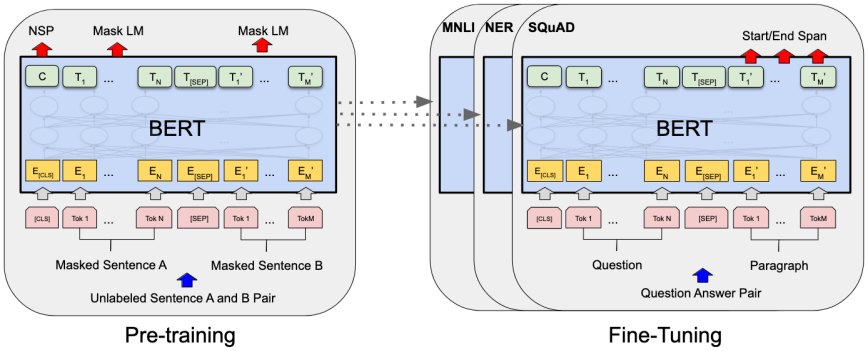

Figure 1.4: BERT model in its pre-training and fine-tuning stages. Reprinted from Devlin et al. [2019]. In the pre-training phase (left) a subset of tokens is masked, which the model must learn to predict. In the fine-tuning phase, the model must learn useful representations of the sequences, depending on the downstream task.

modeling in how probabilities are attributed to a sequence. In causal language modeling, the model assigns a probability distribution over the vocabulary depending on how each item is likely to appear as the following token given a specific history. In masked language modeling, a portion of the tokens in a sequence are masked, i.e., become unknown to the model. The model learns to predict which words should appear in these masked portions of the sequence. This allows for taking advantage of the transformer architecture and its parallelization capabilities. Furthermore, much like CLM, the MLM objective is a self-supervised task. This means that no labeling efforts are needed in order to gather the data necessary to train the model. Finally, the model was also trained on a next sentence prediction (NSP) objective. The setup for this objective was to aggregate a representation from two sequences being fed into a single vector and predicting whether the first sequence preceded the second one originally.

In Figure 1.4, the different phases of the usage of BERT are depicted. In the pre-training phase, the model learns to predict masked tokens and to encode the prediction of the NSP objective into the output of the first token. At fine-tuning time, depending on the downstream task, the model learns to encode its prediction at specific locations of the sequence. For sequence classification, the prediction is encoded in the `CLS` token, i.e., the first in the sequence. For sequence labeling, the prediction is encoded for each token.

Later work has improved upon the training process of the original proposed model [Liu et al., 2019] by changing the training objective, namely removing the NSP objective from the training; by trying a different architecture [Raffel et al., 2020], which includes both encoder and decoder blocks in order to form an encoder-decoder model; or by simply utilizing decoder blocks only and expanding in size [Brown et al., 2020]. These efforts have continued to push the limits of what can be achieved using language models.

Additional considerations. PLMs are usually fine-tuned by updating all parameters on downstream data, and are able to achieve better performance than training from scratch by making use of this transfer learning [Peters et al., 2018, Howard and Ruder, 2018]. However, it has been shown that their training might be unstable due to different factors such as vanishing gradients [Mosbach et al.,

2020]. Furthermore, Popel and Bojar [2018], Liu et al. [2020a] show that the warm-up stage is crucial when fine-tuning transformers, as removing it results in divergence and instability. Moreover, measures are taken in order to motivate the model to perform the knowledge transfer successfully, i.e., avoiding **catastrophic forgetting**, such as gradually unfreezing layers as the training progresses Howard and Ruder [2018].

1.4 Multilingual Language Models

There are over 7,000 languages spoken on Earth. Despite this fact, Ruder et al. [2022] found that 70% of papers at a major conference only evaluated their approach on the English language, other studies have found similar results [Joshi et al., 2020]. This is evidence for there being a clear need for language technologies and research to cover more languages.

Shortly after pre-trained language models were proposed, [Devlin et al., 2019] tried training their language model using an MLM training objective except they included data from multiple languages, namely 100 languages using Wikipedia[2] data. The resulting model was coined **mBERT**. The goal of this type of multilingual model is to have a *base* model that can be readily fine-tuned on different language models.

Shortly after, more multilingual models arose [Conneau et al., 2020, Liu et al., 2019, 2020b, Xue et al., 2020]. All of these languages displayed a phenomenon called **cross-lingual transfer learning** (discussed in detail in section 1.5 which allows the model to leverage knowledge acquired from one language and applying it to another. These models were trained using similar strategies and all performed well across the different languages. **XLM-R** [Conneau et al., 2020], in particular, proves to outperform mBERT significantly in various downstream tasks across most languages. The authors achieved this by:

1. Gathering significantly more diverse data in larger quantities than previous approaches
2. Scaling the model according to the number of languages
3. Scaling the vocabulary size according to the amount of data
4. Simplifying tokenization by using a Sentence Piece[3] [Kudo and Richardson, 2018] tokenizer

The XLM-R proponents focused on scaling multilingual language models, both in terms of quantity and diversity. The authors were also the first to fully document **the curse of multilinguality**, which states that there is a trade-off between downstream performance of cross-lingual learning for low-resource languages and the number of languages that the model supports. This phenomenon is discussed in detail in section 1.5.

[2] `https://en.wikipedia.org/wiki/List_of_Wikipedias`

[3] A sub-token tokenization technique that is language agnostic and allows training on raw text data.

1.5 Cross-lingual Transfer Learning

The ability of leveraging of information gained from one language and applying it to another language is called cross-lingual transfer learning. It follows the same definition of transfer learning we present in section 1.3, with a small caveat. In cross-lingual transfer learning, the learner usually remains the same and can run inference on multiple languages. When inference is run in a domain and/or language that has not seen a training example, we deem it **zero-shot** inference.

Curse of multilinguality. Conneau et al. [2020] propose the concept as the following trade-off: "[support for] more languages leads to better cross-lingual performance on low-resource languages up until a point, after which the overall performance on monolingual and cross-lingual benchmarks degrades". In their experiments, the authors experience a significant degrading in results across all-languages after adding enough languages to a model with fixed capacity. They find similar effects on a high versus low-resource language level, where the sampling rate of languages affects performance in a trade-off manner between high and low-resource languages.

1.5.1 Understanding Cross-lingual Transfer Learning

Wu and Dredze [2019] first document the zero-shot cross-lingual capabilities of mBERT [Devlin et al., 2019]. Since then, further work has gone into investigating this phenomenon [Wang et al., Dufter and Schütze, 2020, Conneau et al., 2020, Pires et al., 2019]. The main aim of these models is, after all, to support various languages and thus allow for a single model to be a foundation model for various tasks and various languages and cross-lingual learning provides additional benefits to them.

Collectively, cross-lingual learning research has determined crucial elements for the phenomenon to occur.

Model Capacity, Structure, and Architecture

Dufter and Schütze [2020] conclude that there needs to be a ceiling and to the model size such that there is enough capacity available to model all relevant languages and not too much such that over-parameterization occurs, and thus no shared representation is encouraged. Similarly, Conneau et al. [2020] conclude that there needs to be a floor on the model capacity, as this will act as a mitigating effect against the curse of multilinguality.

In terms of architecture, transformer models dominate the current LM space. Recently, work has gone into adding adapters to models creating modular approaches [Pfeiffer et al., 2022, 2020, 2023]. These approaches will be discussed in a future section (section 1.7.

Training Objectives

MLM as a training objective is ubiquitous in NLP. However, it is important to study how it might affect cross-lingual abilities. Wang et al. [2019] first studied the effect of the NSP objective originally used by BERT [Devlin et al., 2019]. They find that not only NSP affects monolingual performance, but also cross-lingual

abilities even more drastically. heir experiment setup consisted of eliminating conflating factors by building artificial languages. They test on language pairs that consist of natural languages and artificial languages. They found the effect occurred for all language pairs.

Dufter and Schütze [2020] investigate how auxiliary cross-lingual signals can be provided in a slight modification of the MLM training objective. Instead of masking with random words, masking with related words in another language may boost cross-lingual transfer learning performance.

Maronikolakis et al. [2021] show that care needs to be taken when picking the size of the vocabulary for each language that will be supported by the multilingual model. The authors posture that these compatible vocabulary sizes may encourage multilingual representation spaces in the model. They also propose a metric that may aid future work to select the size of the vocabulary for each language.

Finally, Deshpande et al. [2022] find that zero-shot downstream performance is directly correlated to embedding alignment. Thus, encouraging this alignment through an auxiliary training objective or an objective that seeks anchoring lexical overlap may boost cross-lingual capabilities.

Data and languages

In Dufter and Schütze [2020], the authors argue that lexical overlap is not necessary for cross-lingual learning to arise. However, they note that syntax similarity is necessary if there is no lexical overlap. Deshpande et al. [2022] study how domain similarity may encourage cross-linguality. Their results show that comparable corpora, even without parallel sequences, encourages a shared representation in the model. Parallel sequences perform the best, even without any pairing. These results are compatible with the findings of Dufter and Schütze [2020] who ran a similar experiment using the bible as its example corpus.

A number of studies has been done on determining whether language similarity influences cross-lingual abilities [Dufter and Schütze, 2020, Deshpande et al., 2022, Xu et al., 2022, inter alia]. The experimental setups are similar across the studies: a set of artificial languages based on natural languages are created. These artificial languages include modifications to their structure, e.g., inversion, transliteration, syntax modification. As more modifications are applied, the worse the model performs on the language pair.

Finally, work has gone into investigating how much lexical overlap may influence performance. Though preliminary research suggested that sub-word overlap was paramount for cross-linguality to occur, it has been shown that it is not absolutely necessary and might not contribute to performance as previously thought [Dufter and Schütze, 2020, Wang et al., 2019].

1.6 Efficient Fine-tuning

While PLMs show a lot of promise, the fine-tuning process usually results in updating the weights for the whole model. As these models are trained on massive amounts of data, their size, and thus number of parameters, have increased significantly over the past years. This means that fine-tuning on downstream data is

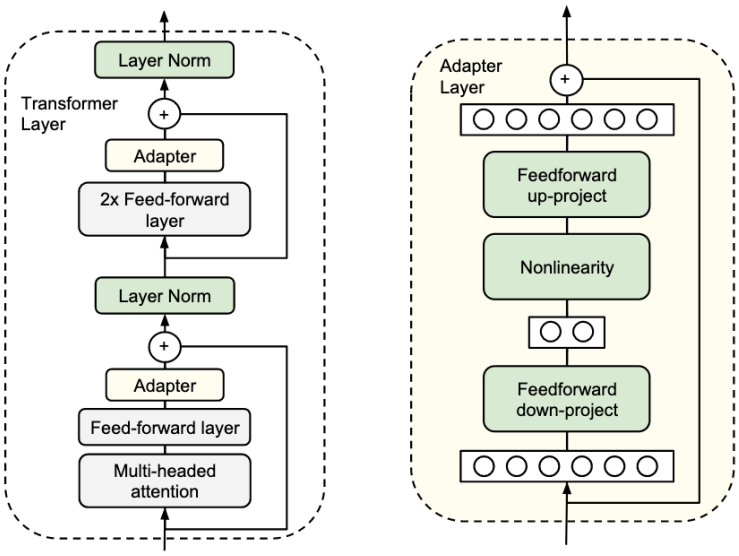

Figure 1.5: Adapters as originally presented. Reprinted from Houlsby et al. [2019]. At fine-tuning time, attention layers and feed-forward layers are frozen; only the bottleneck adapters are updated.

relatively inefficient and computationally expensive. This is particularly relevant for communities of languages which do not have access to a lot of resources, which often occurs on both the data and the hardware front [Ahia et al., 2021]. It is, thus, of uppermost importance for these communities, that proposed approaches are efficient in their implementation.

Houlsby et al. [2019] propose a method to fine-tune a model by adding small module *adapters* to the model. These adapters are added after pre-training has happened, they are not present during pre-training. These modules only contain few parameters, as illustrated in Figure 1.5. Their structure relies on a bottleneck feed-forward subnetwork. The rate of this bottleneck is a hyperparameter that must be found, though the original authors provide general recommendations. Their proposed approach to fine-tuning involves only updating the weights of the adapters, as opposed to updating 100% of the parameters in the model. This results in a significant reduction of parameters that must be updated. Furthermore, the impact on performance is not drastic and, thus, this approach can be very valuable for resource-constrained communities.

More recent approaches allow for models to perform downstream tasks without updating parameters [Brown et al., 2020]. These approaches however will not be covered further in this work and are mentioned for completion's sake.

1.7 Modular Multilingual Language Models

The problem of inefficient fine-tuning of models also affects multilingual language models, with the added challenge that one needs to fine-tune models without disrupting the shared representations and thus avoiding catastrophic forgetting. Pfeiffer et al. [2022] propose using individual bottleneck adapters, each exclusive to each language supported by the model, at every layer of the model. These modules are added even at pre-training time, unlike what is originally proposed

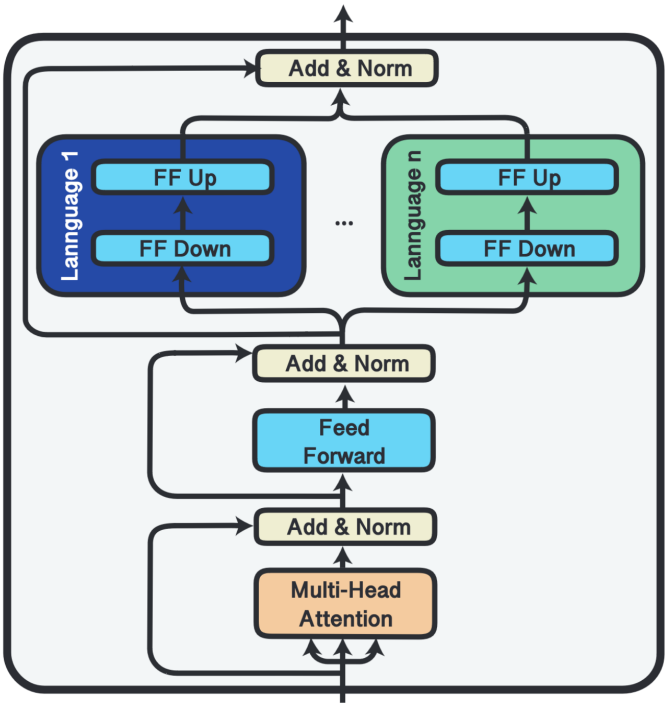

Figure 1.6: X-MOD as presented originally. Reprinted from Pfeiffer et al. [2022]. The adapters are added at, and therefore are part of, the pre-training phase.

in Houlsby et al. [2019].

Figure 1.6 shows where adapters are added. These adapters are added at each layer of the encoder block. From the figure, we can also see that each adapter corresponds exclusively to one language, i.e., these adapters are language-dependent.

Figure 1.7 illustrates how X-MOD is trained and adapted depending on which stage is occurring. At pre-training time, these adapters are also trained. When adding a new language to the model post-hoc, i.e., after pre-training is done, a new embedding matrix is trained and a new set of adapters is trained. The rest of the layer parameters are frozen and thus not trained, we shall call these parameters the *shared parameters*. Finally, at training time, adapters are frozen, and the shared parameters are updated. This implies that modules will remain compatible across each other after fine-tuning. When running inference, the data examples are routed through the relevant language-dependent modules.

Similar proposals have also been released, such as MAD-X [Pfeiffer et al., 2020]. A crucial difference with the MAD-X approach is that not only are there language-dependent adapters, there are also task-dependent adapters. Pfeiffer et al. [2022] show that in some contexts, stacking these adapters might not be optimal, as the increase in performance is marginal and the number of parameters rises significantly.

Finally, Pfeiffer et al. [2023] propose a similar modular multilingual model: modular multilingual T5 (mmT5). In their work, the authors propose a modular alternative to the mT5 [Xue et al., 2021] model. The original mT5 model is a sequence-to-sequence model that performs cross-linguality really well. However, at generation time, it can generate the correct semantic meaning, but it has issues

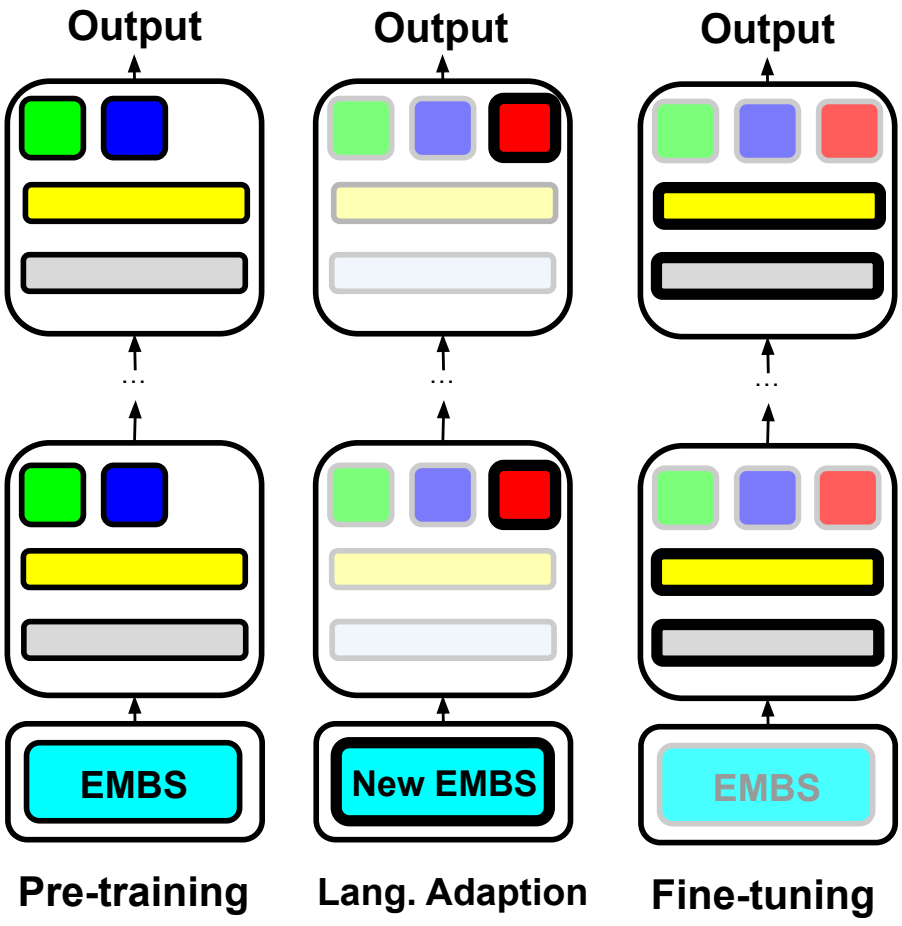

Figure 1.7: X-MOD in its different phases. Grayed-out components are frozen. Bolded edges represent the focus at each stage.

generating in the correct language. This problem is called the **language source hallucination problem**. The authors of Pfeiffer et al. [2023] add language-dependent adapters at pre-training time. The authors claim that they raise the rate of generating in the right language from 7% to 99% when compared against the original mT5.

Chapter 2

Methodology and Hypotheses

In this chapter, we look into the experimental setup and claims in Pfeiffer et al. [2022]'s work. We cover what was tested in their work and what was not tested. Our work focuses on analyzing the claims proposed in Pfeiffer et al. [2022] in more robust settings by trying to introduce controls. We also attempt to qualify in more precise conditions when the claims in the original work hold. This chapter is structured as follows:

- **Training setup in Pfeiffer et al. [2022]**: here we address how X-MOD is pre-trained, adapted for more languages and fine-tuned for downstream tasks.
- **Experimental setup in Pfeiffer et al. [2022]**: we address the experiment setup used by the authors in order to support their claims. These are not meant to be exhaustive sections explaining all of their experiments, only those that are relevant to the claims this work attempts to quantify.
- **Claims in Pfeiffer et al. [2022]**: in this section, we discuss the claims made in the original publication and select a subset of them to be investigated by this work further.
- **Hypotheses**: we present and detail the hypotheses to be tested in our work.

2.1 X-MOD Training Setup

Adapters are commonly added only at fine-tuning time, [Houlsby et al., 2019, Pfeiffer et al., 2020]. The resulting model of Pfeiffer et al. [2022], henceforth **X-MOD**, however adds them at pre-training time. In this section, we discuss this and other decisions made when establishing the training setup for X-MOD. This section covers the variations of X-MOD extensively, but not comprehensively. All information has been extracted from Pfeiffer et al. [2022], unless stated otherwise.

2.1.1 Model Details

X-MOD is an extension of mBERT [Devlin et al., 2019] and XLM-R [Conneau et al., 2020]. Both are multilingual models pre-trained on an MLM objective

Model	Size	# train steps	# langs
X-MOD.base.13.125k	BERT-base	125k	13
X-MOD.base.30.125k	BERT-base	125k	30
X-MOD.base.30.195k	BERT-base	195k	30
X-MOD.base.60.125k	BERT-base	125k	60
X-MOD.base.60.265k	BERT-base	265k	60
X-MOD.base.75.125k	BERT-base	125k	75
X-MOD.base.75.269k	BERT-base	269k	75
X-MOD.base	BERT-base	1M	81
X-MOD.large.prenorm	BERT-large	500k	81

Table 2.1: List of models released by Pfeiffer et al. [2022]. Most models use the BERT-base architecture, which features 110 million parameters, whereas the BERT-large architecture features 340 million parameters.

and have shown great cross-lingual abilities. These models are discussed in more detail in chapter 1. The extension made to these base models is the addition of the bottleneck feed-forward adapters, as described in section 1.7. One adapter per language is added. This means that with every language that is added to the model, the number of parameters increases linearly. It is important to note that, since data examples are routed through their specific modules depending on the language, the computational cost, measured in FLOPs, remains constant. Furthermore, the authors decided to place modules after the layer normalization stage in each transformer block. Finally, a residual connection is placed to allow for adapters to learn a modification to the representation obtained from the transformer block. This modification to be learned can be the equivalent of a `NOP`, i.e., no change is applied. This allows for knowledge to traverse the network directly if no language-dependent alterations are needed.

Table 2.1 details the models that were released by the original authors. The varying values for training steps and number of languages is discussed in later sections.

2.1.2 Data

The pre-training dataset used for the experiments is CC-100 [Conneau et al., 2020, Wenzek et al., 2020]. Conneau et al. [2020] followed the filtering process of Wenzek et al. [2020], which relies on filtering according to the perplexity of a language model contrasted with a high-quality domain such as Wikipedia, to obtain monolingual corpora of what they call "high quality" by filtering CommonCrawl[1]. This filtering of CommonCrawl results in larger amounts when compared to using only data from Wikipedia while maintaining similar quality. This approach of gathering data for the pre-training task differs from that of mBERT [Devlin et al., 2019] which used only Wikipedia data.

Table 2.2 depicts the amount of data available for a subset of languages used in the original experiment setup. We can see from the table that even the lower resourced languages used to test the relevant claims have millions if not hundreds

[1] `https://commoncrawl.org/about/`

Language	**Size** (GiB)	**Tokens** (M)
English	301	55,608
German	67	10,297
Spanish	54	9,374
Bulgarian	58	5,487
Greek	47	4,285
Turkish	21	2,736
Urdu	6	730
Korean	54	5,644
Japanese	69	530
Hindi	20	1,715
Russian	278	23,408
Ukrainian	85	6.5

Table 2.2: Information on CC100 for some relevant languages in the experimental setup of Pfeiffer et al. [2022]. Size is rounded up to the nearest gigabyte. For the full table, see Wenzek et al. [2020].

Stat	**Size** (GiB)	**Tokens** (M)
Average	23.94	2950.92
Median	4.75	337.50
Stdev	48.03	7132.01
Max	300.80	55608.00
Min	0.10	5.00

Table 2.3: Token and size statistics for CC100 languages. Tokens are show in millions and dataset size.

or thousands of millions of tokens available for pre-training. It is very likely that so-called low-resource languages may not have as much data available [Ahia et al., 2021]. Furthermore, although the table does not show quantitative examples, we can infer from the source of the data that it is diverse, containing different styles and domains as it is the result of crawling the web.

As mentioned in section 1.5.1, the method of gathering data used is of great benefit as it features two key aspects: the amount of data is large, and data is diverse in domain.

2.1.3 Languages

The CC100 corpus contains data for 100 languages. Table 2.2 shows a small subset of these languages. The magnitude of the amount of data can be appreciated in this table.

Table 2.3 elucidates how much data is available per language in the CC100 corpus. The smallest language counts on roughly 5 million tokens for its (pre-) training, which in its encoding equates to 100 megabytes worth of data. The media language features a lot more data: around 337 million tokens, which, depending on encoding, roughly equates 5 gigabytes of space.

Language Family	Languages
IE:Germanic	Afrikaans, Dutch, English, Frisian, German, Icelandic, Norwegian, Scottish Gaelic, Yiddish
IE:Albanian	Albanian
Afro-Asiatic	Amharic, Arabic, Hausa, Hebrew, Oromo, Pashto, Sindhi, Somali
IE:Armenian	Armenian
IE:Iranian	Assamese, Bengali, Gujarati, Hindi, Kurdish, Marathi, Nepali, Oriya, Punjabi, Sanskrit, Sindhi, Sinhala, Urdu
Isolate	Basque
IE:Slavic	Belarusian, Bosnian, Bulgarian, Croatian, Czech, Macedonian, Polish, Russian, Serbian, Slovak, Slovenian, Ukrainian
IE:Celtic	Breton, Irish, Welsh
IE:Romance	Catalan, French, Galician, Italian, Latin, Portuguese, Romanian, Spanish
Sino-Tibetan	Chinese
Uralic	Estonian, Finnish, Hungarian
Constructed	Esperanto
Kartvelian	Georgian
IE:Hellenic	Greek
Dravidian	Kannada, Malayalam, Tamil, Telugu
Japonic	Japanese
Austronesian	Indonesian, Javanese, Malay, Sundanese, Tagalog
Koreanic	Korean
Mongolian	Mongolian
Mongolic	Pashto
Kra-Dai	Thai
Turkic	Turkish
Austroasiatic	Vietnamese
Niger-Congo	Swahili, Xhosa

Table 2.4: Languages and their respective language families

In terms of language diversity, the set is typologically diverse. 23 different families are featured in the set of 100 languages. Of these 23 families, 8 are Indo-European. Although only 8 families are Indo-European, most of the languages in the dataset belong to these families. Table 2.4 depicts the exact distribution of the languages and their families.

One of the main ways in which Pfeiffer et al. [2022] support one of their claims relies on an experiment which separates languages based on their language typology. Further, they list a set of languages which have no family in common with the rest of the languages. Those languages are: Vietnamese, Thai, Korean, Japanese, Greek, and Turkish. Previous work [Dufter and Schütze, 2020] has shown that despite there being no lexical overlap, similar syntactical structure in languages may allow for cross-lingual transfer learning. We theorize that this conclusion may be extended to say that the typology of the language is not as relevant as syntax similarity or lexical overlap. A preliminary analysis may conclude that the syntax of the languages with a unique language family may be similar to other languages.

2.1.4 Training Details

The authors of Pfeiffer et al. [2022] share a non-exhaustive description of how their training is carried out. All the training is performed using the `fairseq` library [Ott et al., 2019]. Most models extend the *base* transformer architecture, which consists of 12 layers and a dimensionality of 768. Furthermore, the language adapters have a bottleneck factor of 0.5. This results in modules representing only around 2.5% of the total number of parameters. Some details change depending on the phase the model is being trained on.

At pre-training time. The tokenizer used when pre-training is the original XLM-R tokenizer [Conneau et al., 2020]. The learning rate has a linear schedule and peaks at `7e-4`. At this stage, all parameters are trained and updated.

Adapting new languages. The tokenizer is trained from scratch for the new language. The vocabulary size is set at $30,000$. The learning rate peaks at `1e-4`. Finally, only the new embedding matrix and the new language modules are updated at this stage. This means that all transformer blocks, i.e., the shared parameters, are frozen. It is important to note that tokens that overlap with the original vocabulary are not initialized randomly, but with the values of the pre-trained representation. It is theorized that this shared vocabulary will act as an anchor between the two arising representations.

Fine-tuning to downstream task. The tokenizer and embedding matrix used depends on the language that is being fine-tuned on. These parameters are however frozen at this stage. The learning rate is found through a hyperparameter search and is dependent on the task itself. At this stage, only the shared parameters are updated. The embedding matrix and the language modules are frozen and thus not updated.

The number of training steps at pre-training time is something Pfeiffer et al. [2022] experimented with. They found that guaranteeing the same number of examples per language worked best. This invariably resulted in training for longer. It is, however, unclear from the publication, for how long at each of the other stages the model is trained for. For the fine-tuning stage, only the number of

epochs for each task is provided.

2.2 Tasks

2.2.1 Natural Language Inference (NLI)

The NLI task consists on evaluating two sequences (the premise and the hypothesis) and determining whether a relation exists, namely an entailment as defined in formal semantics, and of which type it is.

It is usually framed as a classification problem, where the possible predictions/labels are: entailment, neutral, contradiction.

Several datasets have been proposed to evaluate this task. Most prevalent is MultiNLI [Williams et al., 2018], which includes examples from multiple domains. XNLI [Conneau et al., 2018] is an extension of MultiNLI, where MultiNLI examples were manually translated to other languages. How these translations are done may impact the quality of the data as they might introduce artifacts, such as leaking information in the premise or hypothesis by using the same vocabulary choices, etc. [Artetxe et al., 2020a].

AmericasNLI [Ebrahimi et al., 2022] is yet another extension. This time, the authors gather translations of the Spanish translations in XNLI to obtain training examples for the NLI task for language indigenous to the Americas.

2.3 X-MOD Experimental Setup

In this section, we explain the setup for three of the main experiments in Pfeiffer et al. [2022]. These are the experiments that were used to back up the main claims of the paper.

2.3.1 Language Adaptation and Zero-Shot Performance

In this experiment, the authors want to show how well languages added post-hoc, i.e., after pre-training is done, perform on the downstream task in a zero-shot scenario.

Languages are added to a version of X-MOD pre-trained on 60 languages for 265k steps. The set of languages being added is relatively typologically diverse: Bulgarian, German, Greek, Spanish, Turkish, Urdu, Mandarin. The authors contrast X-MOD against a version of the model which only features a single adapter for all languages. This single adapter is scaled such that each language retains the same capacity when compared against X-MOD. They test zero-shot performance on three tasks: NER [Rahimi et al., 2019, Pan et al., 2017], NLI [Williams et al., 2018, Conneau et al., 2018] and SQuAD [Rajpurkar et al., 2016, Artetxe et al., 2020b, Lewis et al., 2020]. The authors find that X-MOD outperforms the alternative model on all tasks and languages.

2.3.2 Language Adaptation vs. Pre-training

For this experiment, the authors divide a set of languages into two groups. They then pre-train two models, one for each set of languages, along with the rest of

the 60 languages that are part of the 60 language version of X-MOD. After this, they extend the model to support languages from the group, on which the model was not pre-trained on. The splitting of languages takes into account the family of languages. When evaluating each language, they test zero-shot performance for languages added post-hoc and those that were part of the pre-training set. Their results show that there is no significant difference between pre-training and adding post-hoc.

2.3.3 Language Relatedness

For this experiment, the setup remains the same as the above experiment. They, however, include languages of unique families in the two sets of languages. They test performance on these particularly diverse languages in a zero-shot scenario. The results show that these languages can also perform at the same level regardless of whether they were part of the pre-training or were added post-hoc.

2.4 Claims in Previous Work

We believe that some claims in Pfeiffer et al. [2022] need further exploring, as they are supported by experiments in a restricted scenario: the languages on which they tested are fairly high-resourced and mostly Indo-European. In particular, the claim we focus on is the following:

- **Coverage Claim**: Because of the above claim, the authors further claim that "**X-MOD has the potential to cover all languages of the world, as the model has the capability to be adapted to new languages post-hoc**". Part of the reasoning used to support this claim is a secondary claim, which follows.

- **Pre-training vs. Adding Claim**: Pfeiffer et al. [2022] further claim that "the per-language performance is on par when pre-training vs. when adding the language post-hoc." We will briefly address this supporting claim in Section 4.6.

This work will mainly focus on the *Coverage Claim*.

2.5 Analysis of Previous Experimental Setup

Pfeiffer et al. [2022] back the claims from the previous section by running a series of experiments in very specific scenarios. In this section, we go over what those settings are. In order to analyze properly what limitations the experiments have, we have to consider two perspectives: the data used and the languages used. These were covered in previous sections. We find the following issues with the claims and experiments that support them:

Coverage Claim Most languages that were added or pre-trained on to X-MOD, had a great amount of data of decent quality. In a real life scenario, this may not always be a reality. Furthermore, it is possible that confounding factors were introduced as the similarity between languages was only analyzed at

a typological level. Finally, as the authors point out, testing X-MOD on real-life low-resource languages is something they have left for future work.

Pre-training vs Adding Claim The language setup for testing this claim might involve a lot of confounding factors. Since the set of languages to pre-train on in this experiment is quite large, it was difficult to ascertain whether there are no *related* languages to the languages of *unique* origin. Relatedness and family isolation are concepts that need to be well-defined in order for this claim to be exploitable.

2.6 Hypotheses

The hypothesis of this work is that the main claims from Pfeiffer et al. [2022] are under-specified and need to be quantified. That is, the claims do not hold in all settings and thus need to be better defined. We hypothesize that:

- The effectiveness of the model relies on the quantity and quality of the pre-training data. We hypothesize that there is a lower limit to X-MOD's usability. We test our hypothesis in Chapters 3 and 4.
- The initialization of the new embeddings with anchoring overlapping vocabulary could be paramount in X-MOD's performance. As an overlap may not always take place, we propose to evaluate how much this overlap influences performance. We test this hypothesis in Chapter 3
- Further, we hypothesize that family genealogy is not as important as similar syntactical features and lexical overlap in languages when trying to predict performance of an added language. We briefly address this in Section 4.6. We will leave its thorough analysis to future work.
- Finally, we hypothesize that restricting the claims to a more constrained scenario might aid in better guidelines emerging of when to make use of this model. We address this in Chapter 4.

With this work, we also aim to motivate further research in modular multilingual models analysis.

Chapter 3

Analysis Experiments

In this chapter, we present and discuss experiments that attempt to elucidate which concepts of traditional cross-lingual learning apply to modular cross-lingual learning. Furthermore, we aim to replicate the main results of the X-MOD publication [Pfeiffer et al., 2022]. The chapter discusses the following experiments and results:

1. A reproduction of original main language adaptation for X-MOD. Carrying out this first experiment allows for an experiment pipeline to be established and allows for easier experimenting in later stages of the thesis work.

2. Experiment on the number of training steps necessary for cross-lingual transfer learning to kick in and be successful. With this experiment, we evaluate how sensitive performance is to the number of training steps.

3. We evaluate using perplexity as a parameter for picking a checkpoint for the language adaptation step.

4. We evaluate whether and how much the vocabulary overlap between the new language and the pre-training vocabulary affects final downstream performance.

5. We simulate a low-resource setting by limiting how much data is available in the language adaptation step.

We will discuss further details about the training data that were not included in the previous section, as they are findings separate from previous work. Finally, we also outline the main conclusions and findings for each experiment accordingly.

3.1 Hardware and Computation Time

We run our experiments on single nodes of 8 A100 GPUs. We fix a batch size of 2048 on average. Training time every 10,000 steps ranges from 8 to 12 hours.

Experiments were conducted using a university infrastructure, which is located in a country which has an average carbon efficiency of 0.19 $kgCO_2eq/kWh$.[1]

We estimate the carbon footprint using the *ML CO_2 IMPACT* [Lacoste et al., 2019] tool. Total emissions are estimated to be 0.57 $kgCO_2eq$. per experiment.

[1] `https://ourworldindata.org/co2/country/germany`

3.2 Main Experiment Reproduction

One of the main results presented by Pfeiffer et al. [2022] are their results on performance for languages added post-hoc. In this section, we present the pipeline defined when attempting to replicate their results.

3.2.1 Implementation and Setup

The overall language adaptation pipeline can be described as follows:

Obtain monolingual data for the relevant languages. In the case of this experiment replication pipeline, we simply rely on using the CC100 dataset [Conneau et al., 2020]. Documents are pre-separated and come from diverse sources. Section 2.1.2 includes more information on this dataset. It is important to note that the amount of data available for each language varies.

Train a tokenizer using the gathered monolingual data. We follow the setup of Pfeiffer et al. [2022] to define the vocabulary size: a fixed $30,000$ tokens for every new language added. We call this new vocabulary V_{add}. It is unclear from the paper how this number was picked. For our training, we used the official SentencePiece implementation[2]. We limited the number of sentences to be used to 10 million. These sentences are drawn at random from the monolingual corpus. This is done as it is intractable to train the tokenizer on the amount of monolingual data that some languages have available. Further, we use 40 threads and use a model type of `bpe`. These options all should preserve the quality of the tokenizer as used by the original authors of X-MOD and proponents of XLMR and SentencePiece.

Tokenize the data with the trained tokenizer and **binarize** it so that fairseq [Ott et al., 2019] can process it. At this stage, we also shard[3] the data. For this pipeline, we split the data into 40 shards. This allows for seamless loading of the data in scenarios with constrained resources.

Load pre-trained model and freeze shared weights, i.e., any weight that is not part of the language modules and the embedding layer. This model will contain an embedding matrix $\mathcal{E}_{\text{pre}} \in V_{\text{XLM-R}}$. It's worth noting that positional embeddings are also left unfrozen, meaning they will be updated. At this stage we also initialize the new embedding layer to be trained by doing the following:

- Create a new embedding matrix of the appropriate size: $\mathbb{R}^{|V_{\text{add}}| \times d}$
- Iterate over the elements of V_{add}
- For each $w \in V_{\text{add}}$ that is present in the original vocabulary $V_{\text{XLM-R}}$, we initialize the embedding value for w with the value present in $\mathcal{E}_{\text{pre}}$

It is important to note that the dimensions of $\mathcal{E}_{\text{pre}}$ and $\mathcal{E}_{\text{add}}$ differ. However, this is not a problem since they only differ in the size of vocabulary and both use d as their dimensionality to represent each sub-token. For this reason, the integrity of the traversal of the data remains intact.

[2] `https://github.com/google/sentencepiece`

[3] Split the data into several files on disk so that fairseq loads each individually and the GPU memory does not get filled trying to load the data.

Train language adapters and embeddings. For the training of these parameters, we train on the MLM task. We mask at the originally recommended masking rate of 15% [Devlin et al., 2019]. The learning rate is linearly scheduled with a peak of `1e-4`. We experiment with the number of training steps and warm up steps, as they are not explicitly reported in Pfeiffer et al. [2022]. Table 3.1 shows our results for varying numbers of training steps. We decide on a small range of steps due to hardware and time limitations. Our range goes from 1,000 to 10,000, both for final number of steps and for checkpoint evaluation. We also run a single experiment up to 50,000 steps, which took over 50 hours to finish. As the number of experiments and languages on which we test is large, we decide to limit future experiments to 10,000 steps. Further, due to the nature of the low-resource double bind [Ahia et al., 2021], we decide that 10,000 steps is a good threshold, as resources might not be readily available for low-resource language communities.

Fine-tuning on MultiNLI [Williams et al., 2018] data. This step is completely asynchronous with the rest of the pipeline as long as it happens before the final evaluation. We fine-tuned the pre-trained version of X-MOD we are using on data of the downstream task, in this particular case, MultiNLI. We fine-tune on English data only in this pipeline. This fine-tuning is done with a fixed learning rate of `1e-5`. This hyperparameter differs from the originally recommended learning rate in Pfeiffer et al. [2022], which was an order of magnitude larger than the value we used. This change reflects the different batch sizes we used due to constraints in hardware available. It is unclear what batch size was used in the original fine-tuning, however it can be inferred that this number was 2048, as that was the size used for the other phases of training. We repeat this step for 3 different seeds.

Zero-Shot Evaluation on the downstream task. This phase involves:

- Loading both the fine-tuned model and the model with the additional language adapters and new embedding matrix
- Replacing in the fine-tuned model the embedding matrix and adding the new language adapters
- Route every data point to be evaluated zero-shot to the right language adapters. This mainly involves passing an extra argument per example that signals to which language the example belongs.

After getting the predictions, we evaluate using accuracy. We repeat this process for the three difference seeds of the fine-tuned models. This is the metric used by Pfeiffer et al. [2022] and will allow us to directly compare our results. We average our results and report both the average and the standard deviation.

3.2.2 Results

Table 3.1 shows the results obtained by our pipeline. We run our pipeline by adapting for 4,000 steps and for 10,000 steps. The best final average accuracy value reflects similar performance to the results obtained by Pfeiffer et al. [2022]. There are however some crucial differences. We hypothesize that these differences are due to the differences in hyperparameters.

Language	Ours (4k)	Ours (10k)	Pfeiffer et al.
German	68.1 ± 1.58	71.8 ± 1.18	75.4
Spanish	76.5 ± 0.21	76.5 ± 0.06	78.5
Bulgarian	70.8 ± 0.51	73.7 ± 0.15	77.4
Greek	73.1 ± 0.19	73.6 ± 0.14	76.2
Turkish	57.9 ± 0.91	62.7 ± 1.15	72.4
Urdu	58.8 ± 1.13	60.9 ± 1.11	64.9

Table 3.1: Zero-shot accuracy obtained on the XNLI task when reproducing experiments from Pfeiffer et al. [2020]. Our results represent an average over three seeds. Pfeiffer et al. [2022] report their results for 5 seeds and do not report standard deviations.

Parameter	Pfeiffer et al. [2022]	Ours
Learning rate	1.00E-04	1.00E-04
Batch size	2048*	∼3000
Training steps	Not reported	2,000 - 10,000
Warm-up steps	Not reported	10%

Table 3.2: Experimental hyperparameters in both works. (*) means we have inferred the value. The precise number in batch size differs from language to language in our setup, hence the range for this parameter in the table.

Table 3.2 presents the hyperparameters that differ from the original publication. We can infer from our results that adapting for a larger number of steps leads to better performance in the downstream task. As it is unclear for how many steps the original authors trained, this could be the only factor separating our obtained performance with their reported performance. We investigate this effect further in a future section.

Previous work Dufter and Schütze [2020] has shown that the number of pre-training steps must be large for cross-lingual abilities to arise. Pfeiffer et al. [2022] posture that this may also apply to X-MOD in its pre-training phase, but do not specify if this reasoning could also apply to the language adaptation stage.

3.2.3 Conclusion

From our results, we can safely conclude that our pipeline has been set up satisfyingly, as the results obtained heavily reflect the performance obtained by the original work, bar some differences due to hyperparameter variety. This pipeline will be the base for all experiments presented in this work.

3.3 Effect of Number of Training Steps

In this section, we investigate the sensitivity of X-MOD's language adaptation step to the number of training steps. In the previous section, we presented results for two different training steps: 4,000 and 10,000. In this section, we, again, train our language adapters and a new embedding layer for the German language. We

do this for up to 10,000 steps, with a single experiment of up to ∼50,000 steps. This step count marks the end of a single epoch over the available German data from the CC100 dataset.

3.3.1 Setup

The setup for this experiment is simple: train for N steps in the language adaptation step, save checkpoints for every 1,000 training steps, and evaluate on them.

However, despite the simple setup, some considerations must be made in order to understand the limitations of such a setup:

- As we are using a linear scheduler for the learning rate, evaluating at n steps and training for n steps might yield different results
- The larger N is set to, the more checkpoints that will be evaluated during the warm-up stage

Due to resource constraints, we choose to report results with this basic setup. In order to address the concerns, we also train for a limited number of steps and evaluate only on the final checkpoint. This means that each evaluation point will be at the end of the learning rate schedule and not in different phases of it.

We set $N = 10,000$ for our initial evaluation (every 1,000 steps).

Additionally, we define a set of N_k where $k \in \{4000, 6000, 8000, 10000\}$ for the second part of this analysis. Where the sub-index indicates the final number of steps used for training, i.e., we train for k steps for each N_k. For all runs, the number of warm-up steps corresponds to 10% of N_k.

Furthermore, we train for 50,000 steps without storing checkpoints every 1,000 steps as this results in lower storage demands and faster overall training.

For this training, we use the same hyperparameters as the reproduction experiment, i.e., those recommended by the original publication of X-MOD.

Finally, we evaluate on three seeds of the fine-tuned version of X-MOD on the NLI task. This fine-tuning is done on English data. This is the same setup as in the previous section.

3.3.2 Results

Figure 3.1 shows the results of this experiment. The results show that performance increases with the number of steps. The increase in performance seems to be directly correlated with the rate of change of the parameters in the model are changing, i.e., the learning rate. We use Spearman's correlation coefficient to measure this correlation and obtain an $r_s = 0.92857$. While this is completely expected, it may impact our checkpoint evaluation setup.

Additionally, one can see that the effect continues past the 10,000-step mark as evidenced by the 50,000 training step experiment. After training for 50,000, the model achieved the highest accuracy. When contrasted to the accuracy obtained by Pfeiffer et al. [2022]: 75.4, it is entirely possible that the discrepancy between our results in Section 3.2 lies only in the number of training steps. However, an effect of diminishing returns can be seen in the results. Though there are,

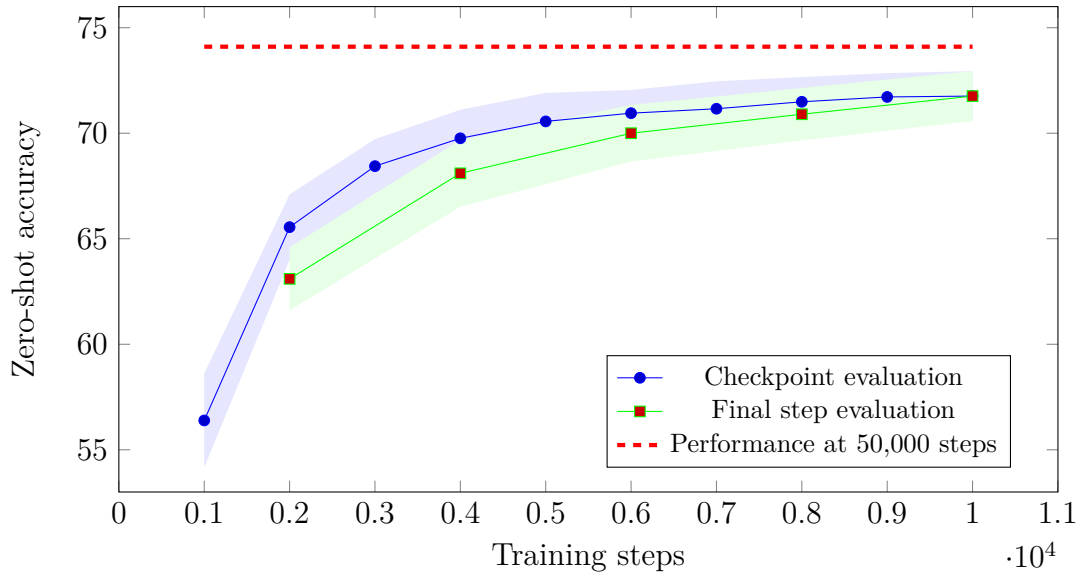

Figure 3.1: Performance for German zero-shot accuracy behavior as number of training steps increases. This is done in the same training with linear learning rate. Evaluation happens using checkpoints every 1,000 steps (checkpoint evaluation). We also evaluate, separately, runs with different total number of training steps and evaluate the last checkpoint (final step evaluation).

potentially, 5 accuracy points to be gained when comparing 10,000 steps and 100,000+, most of those gains are made during the first 50,000 steps if not fewer.

To investigate the relationship between learning rate, total number of updates, and final performance, we perform an additional sub-experiment.

The results of this experiment are also shown in Figure 3.1. We can see how the performance for each N_k, i.e., each training run with k total number of training steps performs slightly worse than evaluating the model corresponding to the longer training at the checkpoint which corresponds with k.

3.3.3 Conclusion

As we had hypothesized, the number of training steps does influence performance significantly. Some additional insights were gained in this experiment:

- Training for longer and obtaining intermediate checkpoints might be more efficient than training for a smaller number of steps
- There are diminishing returns after a certain number of steps. We leave determining this threshold for future work.
- It is unclear how repetitions over the monolingual data may affect these results, as our experiment setup only covered 1 epoch.

From these insights, we may conclude that X-MOD is sensitive to the number of training steps hyperparameter. More work needs to go into determining what number would be a good baseline for different amounts of available pre-training data. A bias towards larger numbers of training steps would be recommended according to our results.

Language	Absolute overlap	Overlap (percentage)
Bulgarian	10,787	35.95%
German	11,479	38.26%
Greek	6,770	22.57%
Spanish	12,154	40.51%
Turkish	10,677	35.59%
Urdu	7,419	24.73%

Table 3.3: Language overlap in vocabulary presented as absolute numbers and as percentages. All the languages have a new vocabulary of 30,000 elements.

k	r_k	Effective overlap
0	0	0%
1	0.1	4.5%
2	0.25	10.1%
3	0.5	20.25%
4	0.75	30.35%
5	1.00	40.51%

Table 3.4: Effective keep rates for each r_k tested in this experiment. It is important to realize that the keep rate does not equate to the final overlap with the original vocabulary, but what portion of the original overlap is kept.

3.4 Effect of Vocabulary Overlap

Previous work [Pires et al., 2019, Wu and Dredze, 2019] has shown that an overlap in lexicon is crucial to cross-lingual transfer learning, as explained in more detail in Section 1.5.1. Conversely, related work [Dufter and Schütze, 2020, Wang et al., 2019] has shown that this may not necessarily be the case.

In this experiment, we aim to further understand modular cross-lingual transfer learning by experimenting with different overlapping rates of vocabulary.

3.4.1 Original Overlap

We first analyze the overlap in vocabulary of the languages added in the previous experiment (Section 3.2). The overlaps in lexicon for these languages are presented in Table 3.3. Overlap percentages range from 22% to 40%. A Pearson correlation analysis indicates that there is only some non-significant positive correlation between the downstream performance and the vocabulary overlap ($r(4) = 0.405$).

3.4.2 Artificially Reducing the Overlap

We now take Spanish, the language with the highest overlap and highest NLI performance, and artificially reduce the overlap between its vocabulary and the original XLM-R vocabulary. We do this by setting a *keep rate* r_k. We randomly draw a value and decide based on it, if it is smaller r_k, whether an overlapping

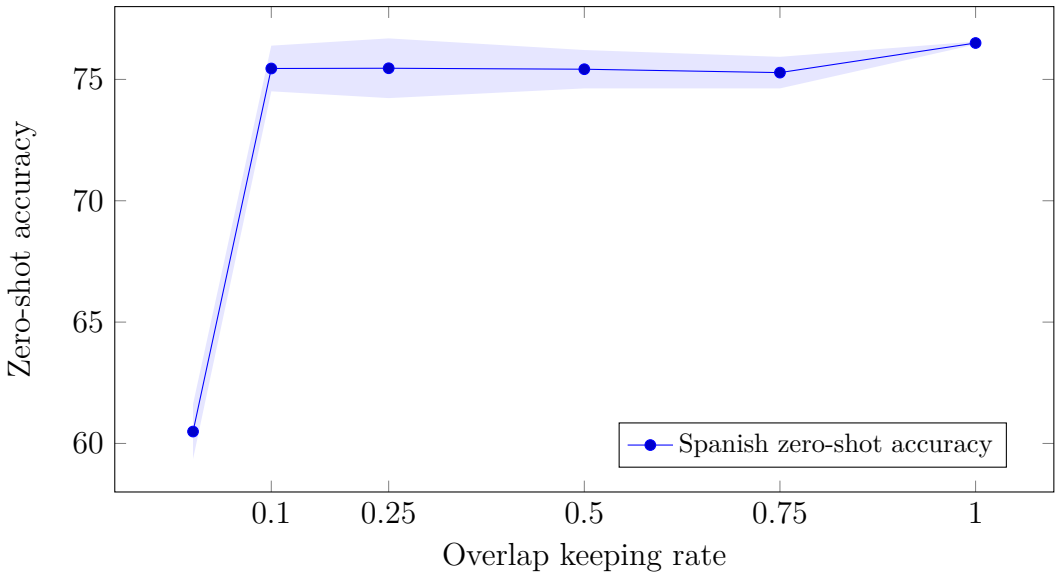

Figure 3.2: Overlap in vocabulary initialize using original embedding matrix vs. performance. The x-axis represents what rate of words are *not discarded*, i.e., not randomly initialized.

sub-token is initialized with the values from the original embedding matrix or at random. This means that r_k represents the rate of sub-tokens kept, i.e., handled as normal. The lower the rate, the more randomly initialized sub-tokens there are.

Table 3.4 shows the effective rates after applying the keep rate for the different r_k values. It is important to note that an overlap of 4.5% is far below the average overlap rate for the languages tested in the previous experiments. With these aggressive rates, we try to test whether X-MOD can handle extreme cases of low overlap, which may be the case for distant from high-resource languages.

Figure 3.2 shows the results of this experiment when testing for three different keep rates. We include the baseline case of not discarding any vocabulary items in this analysis. The resulting numbers for this baseline case are taken from the reproduction experiment results described in Section 3.2.

Discarding elements, regardless of the rate, has an appreciable negative impact in performance. However, we can see from the results that even drastic discarding of overlapping elements in the vocabulary (i.e., as low as $r_1 = 0.1$), performance does not different significantly when compared against results for other keep rates. There is no clear pattern in the slight variability in performance. The standard deviation increases once we start discarding elements, but even variability remains stable throughout the keep rates after a certain threshold. For this reason, it may be worth fine-tuning more models (with different seeds) in order to maximize the probability of obtaining the best performance possible, especially if the overlap is low. Defining a specific threshold remains something outside the scope of this work.

Figure 3.3 shows how the validation perplexity evolves as training continues for the different vocabulary restricted models. Our results show that as long as there is some overlap present, perplexity will plateau around the same value. However, it may take models a different number of steps to reach this convergence,

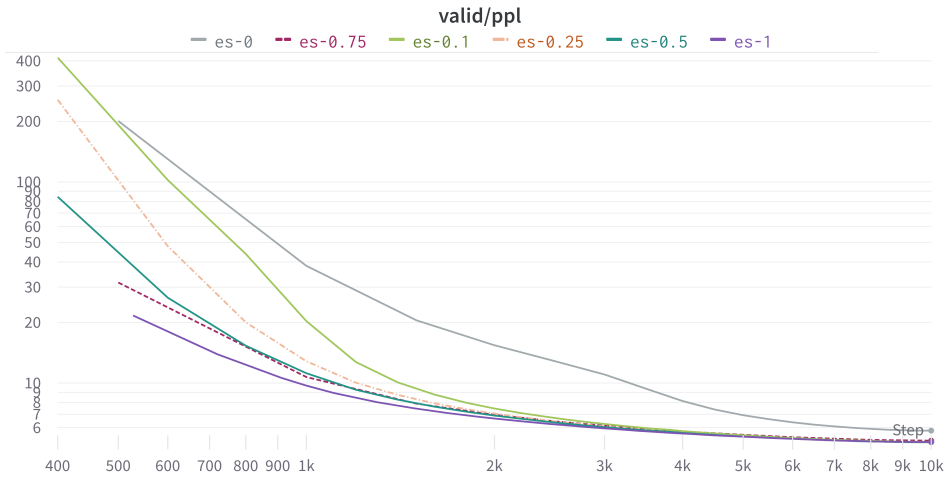

Figure 3.3: Perplexity against the number of training steps for the different artificially limited models. (log-log scale)

depending on the overlap. The exception to the previous statement is the model for which there is no overlap. This model plateaus at a slightly higher value and takes longer to reach it. It remains unclear how training for longer might affect this behavior.

Due to the initialization method of the new embedding matrix, a bigger effect is expected as the keep rate varies. This finding indicates that an overlap, regardless of how minimum it is, and its small size will not affect performance heavily. Though cross-lingual learning happens even when no overlap is found, performance degrades significantly.

3.4.3 Conclusion

From the experiments, we can conclude that a large overlap with the original XLM-R vocabulary is not an element that is absolutely necessary to make X-MOD work. This may be good news for languages that do not have a big lexicon overlap with high-resource languages usually used for pre-training, such as the ones used to pre-train X-MOD. However, at least a minimal overlap is desired, as our results show that disallowing an overlap significantly affects the downstream performance negatively. Ways of inducing even a minimal overlap remains something to be explored in future work. We hypothesize that domains where borrowed words or code-switching often occur may be good sources of monolingual data for the language adaptation step if no sufficient overlap is otherwise found.

The above finding coincides with previous cross-lingual learning analysis in non-contextual embeddings [Artetxe et al., 2017, Smith et al., 2016].

However, from our experiments, we can also infer that a higher overlap does not hurt performance and might even aid it slightly. For this reason, it may be the case that, when training a tokenizer, the sub-token inclusion decision process could be informed with overlap information. i.e., give sub-tokens that overlap with the pre-training vocabulary a higher weight in their chance of being added to the vocabulary. We leave the exploration of this possible enhancement to X-MOD for future work.

	Fraction	Size (MB)	Tokens (M)
Spanish	1	82,000.00	9,374
`centiSpanish`	1/100	836.00	93
`miliSpanish`	1/1000	84.00	9
`microSpanish`	1/10,000	8.40	0.9
`nanoSpanish`	1/100,000	0.85	0.09

Table 3.5: Variations of restriction in amount of data for the Spanish language. Sizes and counts presented here are for tokenized data.

Our experiment aims to quantify the *Coverage Claim* from Section 2.4. We propose that X-MOD can significantly support languages that have at least minimal overlap in vocabulary with the original XLM-R tokenizer.

3.5 Simulating a Low-resource Setting

X-MOD was tested originally only on languages that have an abundant amount of data. In Chapter 2 we show that the average number of tokens for the languages tested is of around 2 billion tokens and a median of 300 million tokens.

The X-MOD proponents claim that X-MOD could potentially cover all languages of the world. We call this the *Coverage Claim* in the previous section. They however did not test the performance of adapting languages with little data to the model. Low-resource languages usually have orders of amount of fewer data than what the languages of X-MOD have to offer. In this experiment, we explore the limits of data scarcity in the language adaptation step in order to quantify the original claim.

3.5.1 Setup

For this experiment, we take Spanish and simulate having significantly fewer data available for it than originally. We pick Spanish for the following reasons:

- Spanish is one of the languages with the largest number of tokens available in the CC100 dataset, with over 9 billion tokens present.
- The quality of the Spanish CC100 data is something that the authors of this work can directly verify
- Spanish has high quality data and is diverse in content

For these factors, we believe that selecting Spanish reduces the number of confounding factors in this experiment. In less constrained scenarios, all languages would be tested. We leave this further exploration to future work.

We limit the amount of data artificially by sampling sentences at random from the corpus. Table 3.5 shows the fraction of data used for each variation of data available. As sentences are drawn randomly, the exact rates and sizes do not reflect the fraction exactly. This is mostly the case for the smaller quantities.

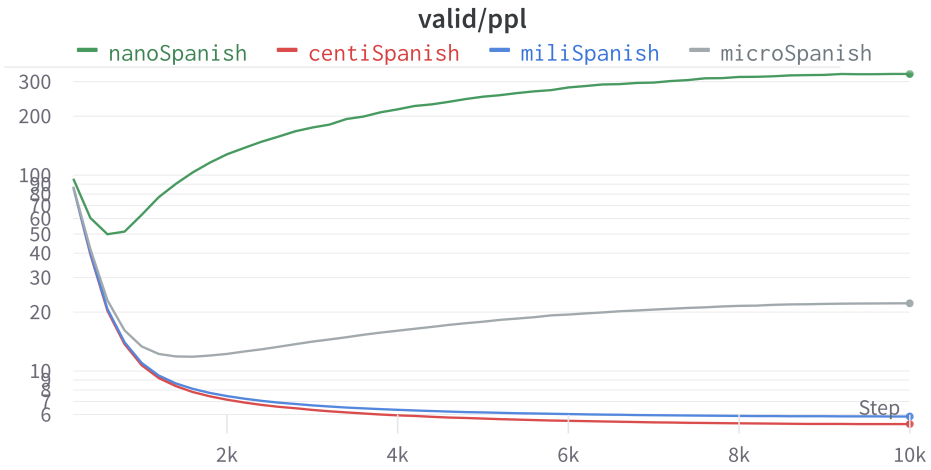

Figure 3.4: Perplexity (logarithmic scale) for each limited version of Spanish as training steps progress.

The smallest fraction of data for this experiment is roughly $1/100,100$th of the original available data. This equates roughly to 90,000 tokens available for training.

Using the limited amount of data for each variation, we train a new tokenizer using only the data available. This further simulates a real scenario where the limited amount of data is the only available data for all parts of the process. However, all vocabularies are of size 30,000.

Regardless of the available data, we train for 10,000 steps using the same hyperparameters from the main reproduction experiment. The batch size however is influenced by the amount of data available once all examples available all fit in one batch of distributed training. At this point, the batch size will be limited by the available data.

We use the same validation set as for the original reproduction experiment in this setup. This means that we will be evaluating against the same distribution. However, perplexity values are not directly comparable, as we are using a different vocabulary for each model.

3.5.2 Results

Figure 3.4 shows how perplexity evolves as we keep training. We can see that after a certain threshold, the validation perplexity begins to rise. It also stabilizes after a certain number of steps. This divergence point varies across the different language variations. The `centiSpanish` and `miliSpanish` variations do not suffer from this divergence. Perplexity continues to decrease as the training continues, albeit very slowly. This is not unlike what is observed for training Spanish with all available data.

Due to this divergence effect, we decide to evaluate the best checkpoint (per validation perplexity), and the last checkpoint of the model after training is done.

For inference, we use the same checkpoints for three different seeds of the fine-tuned model. This makes the downstream performance entirely dependent

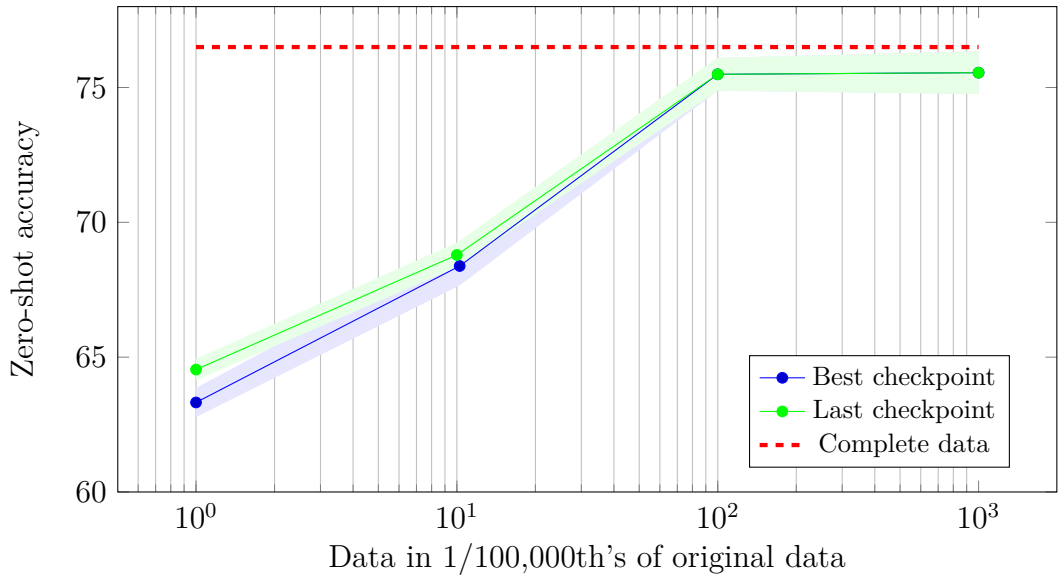

Figure 3.5: Downstream performance measured in zero-shot accuracy for each amount of data available in 1/100,000ths of the original data. This would make the first data point `nanoSpanish`, and so on. For the first two data points, the last and the best checkpoint are one and the same.

on the language adaptation step of X-MOD, which aims to add a language to the model.

In Figure 3.5 we show the results of zero-shot accuracy following the setup of the original reproduction experiment.

We can see from the results that the "best" checkpoint, as per validation perplexity, does not always translate to better downstream performance. In fact, in both instances, the "last" checkpoint performs slightly better despite perplexity higher perplexity.

3.5.3 Conclusion

When fixing the number of steps to be trained for in the language adaptation step, the amount of data available has a direct impact on performance. This impact might be drastic when reaching the lowest amounts of possible data (less than 1 million tokens).

On the lower end of the spectrum of data availability, unexpected behavior occurs: the perplexity performance of the language model does not necessarily reflect performance on the downstream task. The model might even start diverging soon after training begins.

On the higher end of the spectrum, however, gains obtained by adding more data start to decrease. Assuming the monolingual data is of good quality, validation perplexity will continue to decrease as the training continues.

We can directly relate these experiments to the *Coverage Claim* from Section 2.4. It is unclear how many of the world's languages would reach the threshold necessary to experience the expected behavior of X-MOD. Our experiments suggest that this threshold lies somewhere between 1 and 10 million tokens in monolingual data. This assumes quality and diversity, as this is what was ob-

served on the Spanish data used for the experiment. Thus, we can conclude that the claim must be restricted in terms of available data and its quality.

Chapter 4

Low-resource Setting Experiments

In Chapter 3, we simulate a low-resource scenario by arbitrarily and artificially restricting available data and other parameters. In this chapter, we present experiments and our analysis on using X-MOD on a set of languages, for which there is not abundant monolingual data available. The chapter is structured in the following way:

1. We introduce the languages to be tested on and their properties
2. An analysis of the quantity and quality of the data available for the languages
3. We naively test an adaptation strategy and try to apply findings from previous chapters
4. We test an alternative adaptation strategy proven successful in other contexts
5. We present recommendations on using X-MOD with low-resource languages

4.1 Languages

In this work, we will deal with languages from the Americas, namely those which were studied and for which data was collected in the AmericasNLI work [Ebrahimi et al., 2022]. In this section, we present the languages and their main attributes that are relevant to this work.

Table 4.1 shows a summary of the languages presented. When possible, we use the local name for each of the languages, much like it is done in the main AmericasNLI publication [Ebrahimi et al., 2022]. We adapt summaries for each language from Ebrahimi et al. [2022].

Aymara Aymara is a language spoken in Bolivia, Chile, and Peru. It has around 2 million speakers. Due to the terrain in the region, the language has many dialects. The AmericasNLI data was translated into the Aymara La Paz dialect specifically. The language features an SOV word order in main clauses.

Language	ISO	Family	Regions
Aymara	aym	Aymaran	Bolivia, Peru, Chile, Argentina
Asháninka	cni	Arawak	Peru, Brazil
Bribri	bzd	Chibchan	Costa Rica
Guaraní	gn	Tupi-Guaraní	Paraguay, Bolivia, Argentina, Brazil
Nahuatl	nah	Uto-Aztecan	Mexico
Otomí	oto	Oto-Manguean	Mexico
Quechua	quy	Quechuan	Throughout Andes Mountains
Rarámuri	tar	Uto-Aztecan	Mexico
Shipibo-Konibo	shp	Panoan	Peru
Wixarika	hch	Uto-Aztecan	Mexico

Table 4.1: AmericasNLI [Ebrahimi et al., 2022] languages, their ISO code, linguistic family, regions where the languages are spoken.

Asháninka This language is spoken in Peru and has around 70,000 speakers. It features VSO word order. The language spoken in a small region delimited by the Andes mountains and the Amazon River.

Bribri Bribri is a tonal and endangered language spoken in the southern part of Costa Rica. It is only spoken by roughly 7,000 people. It features SVO word order. Much like with other indigenous languages of the Americas, there exist several orthography standards for the language. To further complicate things, sometimes the same diacritics are encoded differently depending on the author. The AmericasNLI team worked on standardizing these discrepancies in their released data. They translated the data into the Amubri dialect.

Guaraní Guaraní is one of the biggest languages analyzed in this work. It features an SVO word order and is the language of 6 to 10 million speakers. It has an official orthography standard.

Nahuatl Nahuatl is regarded as a language family with around 30 different variants spoken in Mexico. It is spoken by over 1 million people. It features SVO, VSO and SOV word order depending on the emphasis of the sentence. It has no orthography standard. The AmericasNLI team translates their data into the central dialect of Náhuatl de la Huasteca. They also normalize the orthography to a version "close to Classic Nahuatl".

Otomí Otomí is a language spoken in Mexico and has 9 different variants. The AmericasNLI standardized their translations to a dialect spoken by fewer than 100 people. Otomí is a tonal language and is spoken by around 300,000 speakers. It features an SVO word order.

Quechua Quechua is a language family as opposed to an individual language. It is mainly spoken in Peru and is spoken by over 8 million people. The language uses an SOV word order. The AmericasNLI examples were translated to the Quechua Chanka dialect.

Rarámuri Rarámuri is a language spoken in Mexico and has around 90,000 speakers. It features SOV word order. The AmericasNLI examples are translated into the Highlands variant and orthographically standardized.

Shipibo-Konibo Unlike other languages, Shipibo-Konibo has an official orthography standard, which the AmericasNLI examples adapt. This language is

Language	Size (KB)	Tokens (K)
Aymara	1,100	155.9
Asháninka	346	39.5
Bribri	386	59.3
Guaraní	4,600	660.1
Nahuatl	3,400	462.7
Otomí	608	99.4
Quechua	15,000	1,729.3
Rarámuri	858	123.2
Shipibo-Konibo	684	91.0
Wixarika	565	98.3

Table 4.2: Monolingual data available for the languages used in AmericasNLI. The size is given in kilobytes and the number of tokens in thousands.

spoken in Peru by around 35,000 speakers. It has SOV word order.

Wixarika Wixarika, or Huichol, has 4 variants spoken in Mexico by approximately 48,000 people. It features SOV word order. The AmericasNLI examples are translated into the Northern variant. They adopt what seems to be the more common orthography standard.

These languages cover an almost continuous territory from Mexico to Argentina. Languages are diverse in their morphology: we have languages for all analytic, polysynthetic, and agglutinative types. Similarly, their syntax varies greatly. For a more exhaustive description of the languages, refer to our main source of the summaries in Ebrahimi et al. [2022].

4.2 Available Data

In this work, we aim to compare our results with the results obtained in Ebrahimi et al. [2022]. For this reason, we aim to use only the data provided by them.

For their adaptation experiments, the AmericasNLI team gathered parallel data for translation augmentation of data. They restrict themselves to using only the monolingual data from this parallel set to further adapt XLM-R.

For this reason, we restrict ourselves to using only this dataset. Table 4.2 shows the amount of data available for each language that on which we test in this section. As AmericasNLI represents a true low-resource scenario, we can see how available data is orders of magnitude smaller than for the languages of the CC100 dataset.

The way to collect data varies depending on the language. As the authors gathered parallel data, they relied on sources of different domains. For Bribri, for instance, spontaneous conversations were recorded and translated [Solórzano, 2017]. For other languages, translations of books or news are used as a source, e.g., Aymara uses parallel data from Global Voices [1]. Table A.4 shows a sample sentence per language. There appears to be a variety of tokenization standards, sentence lengths, and overall quality. With these differences, it is hard to evaluate

[1] `https://globalvoices.org/about/`

the quality of the data. We can determine that religious, historical and official texts are predominant in the data, which may harm diversity and, in turn, overall quality.

It is important to note that all languages are closest to our `nanoSpanish` variant from Section 3.5. We therefore expect to see some of the same effects we saw in previous experiments.

4.2.1 AmericasNLI

The AmericasNLI project [Ebrahimi et al., 2022] is a collaborative effort to extend XNLI [Conneau et al., 2018]. This entails that XNLI data is translated into new languages, in this case to the indigenous languages of the Americas. This translation were manually done by human translators. The source language for the translation was Spanish, as most translators are speakers of Spanish. A domain restriction was also applied in order to avoid code-switching and borrowed words from Spanish to be prevalent in the data.

The resulting AmericasNLI dataset consists of 750 NLI examples for most languages, with an even split among each class of the NLI problem. This means that the random baseline would lie around 33%.

The AmericasNLI team also experimented with different zero-shot approaches to the high-level task of NLI using their gathered data. They were able to gain significant accuracy points over the random baselines.

In this work, we will be comparing our performance against the variant of their results called XLM-$R_{+MLM\ (en)}$. This variant entails further adaptation of XLM-R using monolingual data for each of the AmericasNLI languages. They then fine-tune XLM-R on the **English** data for NLI. Finally, they evaluate, zero-shot on the target language.

4.3 Adaptation

In this section, we will discuss the adaptation phase, i.e., adding the AmericasNLI languages to X-MOD. More specifically, we will naively add languages to different versions of X-MOD to evaluate how X-MOD performs when the languages added are "truly low-resource" languages [Ebrahimi et al., 2022].

4.3.1 Setup

For the adaptation stage, we follow the same steps as Section 3.2 with a few modifications:

1. Restrict monolingual data to that used by the AmericasNLI team
2. Train tokenizers using a vocabulary size of 15,000. Some languages do not have enough data to train a tokenizer with a vocabulary of 30,000 tokens.
3. Run the language adaptation step as usual with the same hyperparameters, i.e., add language modules at each layer and initialize a new embedding layer to be trained as outline in previous chapters.

ISO	Overlap	Overlap (%)
aym	5219	34.79%
bzd	1958	13.05%
cni	1343	8.95%
gn	5457	36.38%
hch	1554	10.36%
nah	2824	18.82%
oto	2566	17.10%
quy	2605	17.36%
shp	1466	9.77%
tar	1585	10.57%

Table 4.3: Lexical overlap between AmericasNLI languages and the vocabulary published for XLM-R. We present the numbers in raw form and in percentage.

4. Our language modeling validation data is drawn from the validation set of the AmericasNLI dataset. Meaning, we evaluate directly the perplexity in the AmericasNLI domain.

For evaluation, we evaluate using the AmericasNLI released dataset, readily available on the HuggingFace website [2].

4.3.2 Vocabulary Overlap

In previous sections, we discuss the effect the lexicon overlap has on the performance of X-MOD. For this reason, we evaluate the lexicon overlap between the different indigenous languages relevant to this paper and the languages of XLM-R. We utilize the same setup as in Section 3.4.1.

Table 4.3 shows the overlap between the original XLM-R vocabulary and the vocabulary obtained after training a tokenizer with a vocabulary size of 15,000. Larger amounts of data available does not seem to directly translate to higher overlap percentages.

Most overlapping rates are, as expected, lower than for languages that are part of the CC100 dataset. According to our findings in Section 3.4, however, some overlap is needed for X-MOD to perform well. All languages from AmericasNLI seem to have some existent overlap.

We leave experimenting on artificially limiting the overlap for these languages as an experiment for future work.

4.3.3 60-language X-MOD

So far in this work, we have worked with a variant of X-MOD that was pre-trained on 60 languages. What is particularly interesting about this, in the context of testing indigenous languages from the Americas, is that Spanish is not part of this 60-language version. We hypothesize that this will hurt performance.

[2] `https://huggingface.co/datasets/americas_nli`

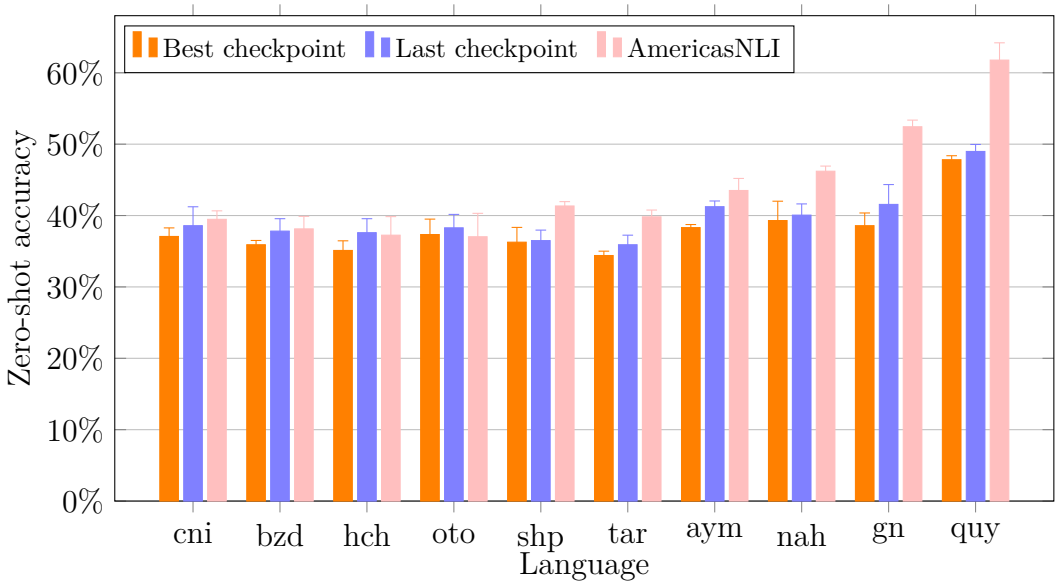

Figure 4.1: Performance for zero-shot accuracy on the AmericasNLI test set. We show results when evaluating with the best checkpoint, the last checkpoint, and the reported AmericasNLI numbers [Ebrahimi et al., 2022]. The languages are ordered by amount of data available in ascending order.

Figure 4.1 shows the results obtained. We evaluate on the last checkpoint, on the best checkpoint (as per validation perplexity) and the results reported by the AmericasNLI team, specifically the XLM-$R_{+MLM\ (en)}$ approach.

We report performance on both checkpoint selection approaches, as the amount of data available for all languages is significantly low. We saw in the previous chapter that performance may vary between the two approaches. Furthermore, we saw that the last checkpoint performs better than the best one according to validation perplexity. We see the same effect with AmericasNLI data.

Despite all languages being able to significantly beat a random baseline (of 33%), none of the models were able to surpass the performance reported by the AmericasNLI team.

4.3.4 81-language X-MOD

For this experiment, we add the Americas languages to the 81-language version of X-MOD. It is worth noting that this model has the same capacity as the 60-language version. The only additional capacity pertains to the 21 additional languages. Since we will only be using the shared parameters for our experiments, the additional capacity is not used.

This version of X-MOD features Spanish as one of the 81 languages. The ISO codes for the other 20 additional languages are the following: *as, br, bs, fy, gd, jv, kn, mg, mr, om, or, pa, su, xh, yi, bg, de, el, es, tr, ur, zh.*

Figure 4.2 shows our results for this experiment. We see from these results that adding the AmericasNLI languages to the 81-language version of X-MOD helps to bridge the gap between our previous results and the results reported by the AmericasNLI team. Figure 4.3 further corroborates this by directly presenting the difference between our results and the XLM-$R_{+MLM\ (en)}$ results from Ebrahimi

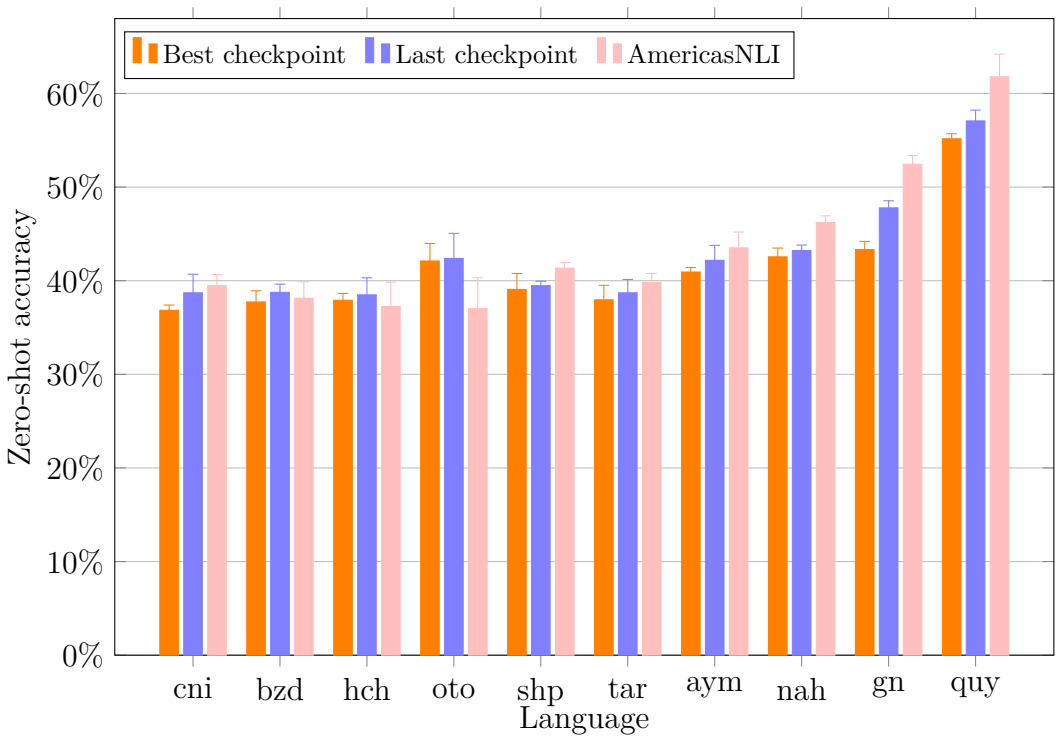

Figure 4.2: Performance for zero-shot accuracy on the AmericasNLI test set when adapting the **81-language version of X-MOD**. Again, we show results when evaluating with the best checkpoint, the last checkpoint, and the reported AmericasNLI numbers [Ebrahimi et al., 2022]. The languages are ordered by amount of data available in ascending order.

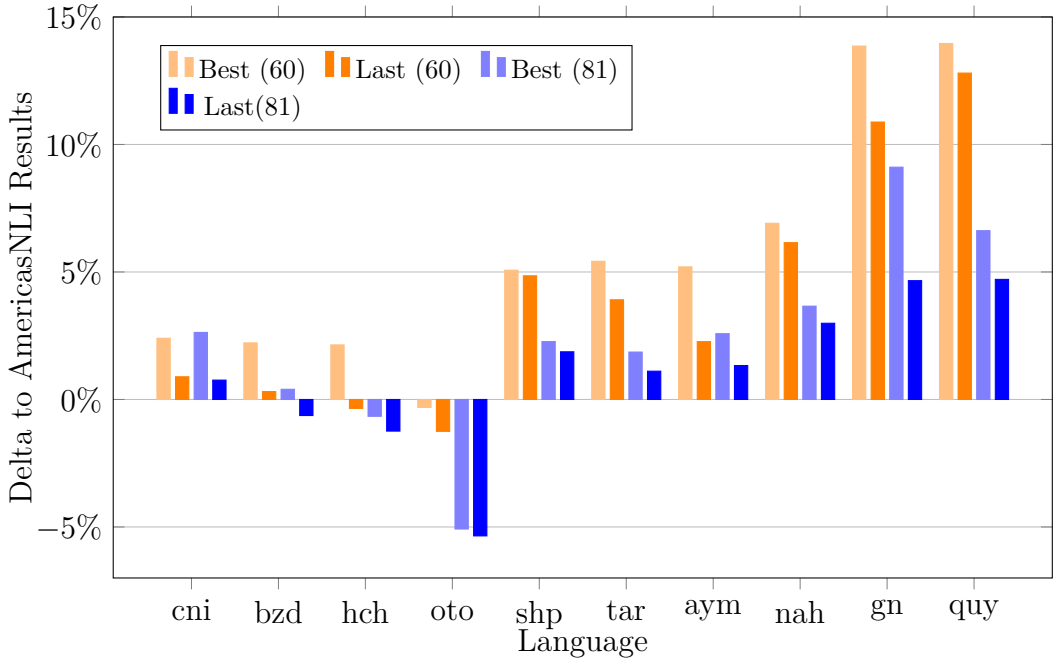

Figure 4.3: Difference between our achieved performance and the reported performance in Ebrahimi et al. [2022]. Lower is better. The languages are ordered by amount of data available in ascending order.

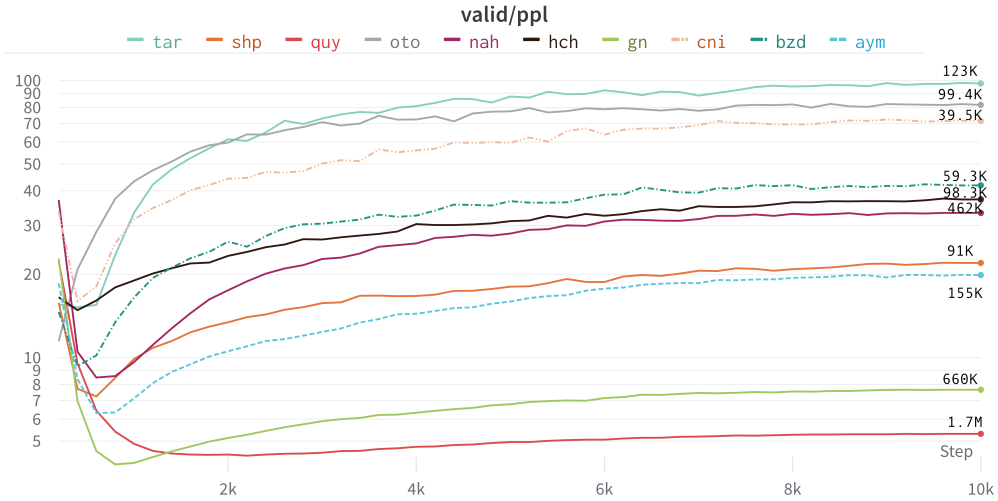

Figure 4.4: Perplexity (in a logarithmic scale) when adapting languages to the 81-language version of X-MOD. All 10 languages of Ebrahimi et al. [2022] are shown. We also annotate the number of tokens available for each language.

et al. [2022]; we can see how the delta decreases when comparing the 60-language version and the 80-language version in almost every case. In some cases, such as for the case of Otomí and Wixarika, we surpass the reported results. In other cases, such as Guaraní, Nahuatl, and Quechua, we only bridge the gap but do not surpass the results. Coincidentally, the latter languages are languages that have some of the highest data availability from the AmericasNLI languages.

4.3.5 Conclusions

From these adaptation experiments, we can infer some conclusions about how X-MOD performs on low-resource settings.

- The findings from our artificial setup from Chapter 3 seem to apply to real life low-resource settings.
- Adapting the model to new languages might require more than 10,000 steps, as seen in previous sections. It is unclear if this is the reason for the gap between our results and those reported in Ebrahimi et al. [2022].
- The phenomena found in Chapter 3 applies to the monolingual corpora used in this chapter. This might mean that the quality of the data might not be as important to these factors as the quantity.
- As seen in Figure 4.4, the behavior of perplexity seems explicitly linked to the quantity of data. Languages that have more data than or close to the amount of `microSpanish` from the previous section, seem to escape the perplexity divergence phenomenon.

4.4 Merged Adaptation

In the previous section, we run the language adaptation phase in a naive and direct way. We do not deviate from previous setups. In this section, we attempt to adapt our X-MOD models using multilingual adaptive fine-tuning (MAFT) [Alabi et al., 2022]. This approach shows great promise in African languages, which are not high-resource languages. In the original MAFT approach, the language is adapted as a whole. The authors also experimented with a modular approach, but this was used after the language adaptation phase was done, i.e., the modular approach made use of the adapted pre-trained language model.

4.4.1 Setup

In this experiment, we attempt to run the language adaptation phase on all languages at once. That is, our adaptation data is a combination of all data available for all languages available. This entails adding only one adapter, which handles multiple languages, to the model at each layer. The capacity of the adapter shall remain the same as for previous experiments. We do this in two ways:

- **Merged Adaptation** We naively combine data by concatenating all datasets and randomizing their order. We do this in hopes of guaranteeing a mix of languages per batch.
- **Balanced Merged Adaptation** We combine data, but we try to mitigate imbalances in the proportion of data available per language. This is done because some languages have significantly more data than others. With this approach, we try to prevent a subset of languages to dominate the language adaptation phase.

We also exclusively combine AmericasNLI languages in this experiment without mixing in any high-resource languages, as done in the original proposal of MAFT. We do this because we experiment with both the 60 and the 80-language version of X-MOD. Experimenting with combining languages from the 21 languages with languages of the Americas may be something worth exploring, and we leave this as possible future work.

4.4.2 Results

Figure 4.5 shows the obtained results. There are no clear trends that indicate the MAFT approach surpasses individual adaptation. However, this approach poses an advantage in terms of computational resources, as one single adaptation run results in all languages being added. The performance, however, seems to drop for some languages significantly.

The bridge between the best results in Ebrahimi et al. [2022] and our merge approach grows for the languages that have the most resources. Finally, it is worth noting that this merging contrasts with the original purpose of X-MOD, which is to isolate languages into modules and combat the curse of multilinguality.

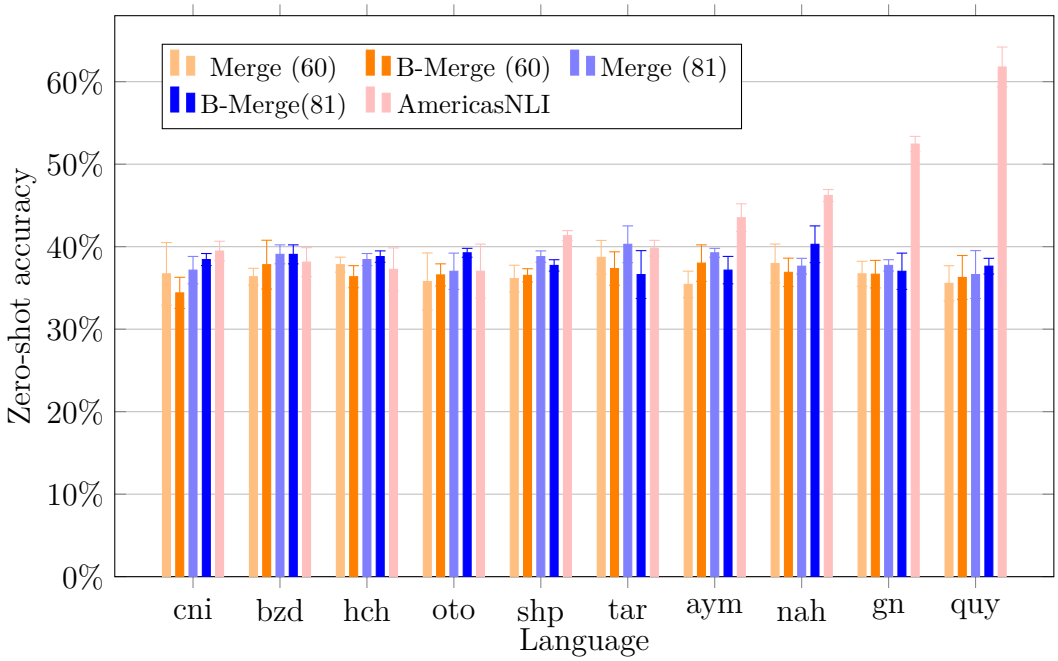

Figure 4.5: Performance for zero-shot accuracy on the AmericasNLI test set. We show results for our simple merge approach (Merge) and our balanced merge approach (B-Merge) on the two different versions of X-MOD, the 60 and the 81 language versions, and the reported AmericasNLI numbers. The languages are ordered by amount of data available in ascending order.

4.5 Summary of Results

In Table 4.4, we present a summary of our results. We show the average zero-shot accuracy across all languages for all seeds for each approach. We can see that our X-MOD setup was not able to surpass the best results presented by the AmericasNLI team. Using the last checkpoint of the adaptation phase of the 81-language version of X-MOD performs best in our setup. Although our setup was unable to beat the average performance, it was able to surpass the performance of Ebrahimi et al. [2022] in some individual languages, like Otomí and Wixarika.

4.6 Recommendations on Using X-MOD in Low-resource Scenarios

Through our experiments, we have gained insights into how to best use X-MOD in low-resource scenarios. In this section, we present these insights in the form of recommendations on how to best make use of X-MOD.

1. As with other methods, gathering monolingual data is of paramount importance. Our experiments show that 10 million tokens should suffice to achieve the expected performance.

2. If absolutely no vocabulary overlap is found, artificially introducing it would be ideal. This could happen by gathering more monolingual data or crafting sentences that include some overlapping vocabulary.

Approach	X-MOD Version	Avg. Accuracy
Naive (last)	60	39.65%
Naive (best)	60	38.01%
Naive (last)	81	<u>42.68%</u>
Naive (best)	81	41.36%
Merge	60	36.72%
Balanced Merge	60	36.69%
Merge	81	<u>38.20</u>%
Balance Merge	81	38.01%
Ebrahimi et al. [2022]	N/A	**43.70%**

Table 4.4: Summary of results for all approaches tested. Average zero-shot accuracy across all seeds and languages are reported. The best approach is bolded and the best result in each subdivision is underlined.

3. Defaulting to a very large number of training steps is desired, i.e., a number in the order of dozens of thousands, hundreds of thousands if resources allow for this.

4. Use the last checkpoint from the adaptation step to obtain the parameters for the language modules and embedding matrix, despite what the validation perplexity may indicate. This is especially true if you do not have abundant monolingual data for the language modeling task.

Conclusion

In this work, we have analyzed X-MOD and have tried to quantify the claims in the original work. Here we summarize our main findings.

Summary and Discussion

We test X-MOD in contexts where it was not originally tested by artificially changing the available data. We find that a high vocabulary overlap may not be paramount to good cross-lingual transfer learning, though some overlap must be present for performance to not degrade drastically. We also find that specific amounts of monolingual data may result in X-MOD not behaving as expected, namely that, despite divergence in the language adaptation step, the model may still successfully learn representations of the language that allows for cross-lingual transfer learning to happen.

Furthermore, we explore its sensitivity to hyperparameters, such as total training steps and checkpoint selection strategies. We find that the number of **final** number of steps is an important hyperparameter due to the nature of the learning rate scheduler. We show that picking an intermediate checkpoint at the n-th step performs better than training for n steps for low values of n.

Additionally, we test X-MOD in a real-life low-resource scenario. We observe similar phenomena from our artificially restricted scenario. This further indicates that X-MOD is sensitive to the quantity of data. However, we may also observe that X-MOD is sensitive to the quality of the data in the adaptation stage; the `nanoSpanish` experiment shows that limited amounts of data of high quality can perform significantly better than comparable amounts of poor data.

Finally, we compile a set of recommendations on how to best use X-MOD, specifically in a low-resource scenario, by taking into account data and resource limitations. These can be summarized as gathering at least 10 million tokens worth of monolingual data for the adaptation phase. These 10 million tokens need not be of optimal quality. Some word overlap is desired, even if it's not large. If there is no word overlap, introducing some could be very beneficial to performance. Training for tens of thousands of training steps of possible is ideal for the best performance, even if not a lot of data is available.

Here, we repeat the second claim introduced in Chapter 2:

Coverage Claim: "X-MOD has the potential to cover all languages of the world, as the model has the capability to be adapted to new languages post-hoc", from Pfeiffer et al. [2022].

We are successful in quantifying and constraining these claims as per our recommendations above. The X-MOD approach remains promising given the

right application and context, and when using the right hyperparameters.

We leave further quantifying of the claims to future work.

Future Work

Among other factors, we leave to future work repeating these same analyses on larger numbers of training steps, for different tasks (such as NER) and applying the different restrictions to more than one language.

Due to time constraints, the *Pre-Training vs Adding Claim* from Chapter 2 remains to be analyzed further. Here, we repeat the Pre-Training vs. Adding for reference:

"It makes little difference in downstream performance whether there are related languages or not in the pre-training set of languages when adding a language post-hoc.", from Pfeiffer et al. [2022].

In the following section, we present a preliminary analysis on language similarity that goes beyond language family as analyzed in Pfeiffer et al. [2022].

Language Similarity Analysis

Previous work Dufter and Schütze [2020], Conneau et al. [2020], Wang et al. has shown that language similarity might not be crucial to successful cross-lingual abilities, but rather lexical overlap and syntactical similarities may have a bigger impact in performance. These factors can not always be controlled for by simply analyzing genealogy.

In this analysis, we discuss the results of exploring language similarity beyond language family. For this work, we make use of a newly released dataset of structural features in languages Skirgård et al. [2023]. 40 of the features from GramBank were used to compute our similarity scores. The list of features used is available in Appendix A. We use these vectors to represent each language as a 40-dimensional vector. The similarity between two vectors was then calculated by using the hamming distance:

$$(H(e,f))_i = \begin{cases} 1 & \text{if } e_i = f_i \\ 0 & \text{if } e_i \neq f_i \end{cases}$$

Where e and f are vectors of the languages to be compared. Finally, average the dimensions of the vector representing the similarity of the two languages.

We analyze languages that were relevant to the experiments in Pfeiffer et al. [2022], in particular experiments that support the claim.

Figure 4.6 shows our preliminary results. Our results indicate how syntactical features can result in high similarity scores for unrelated languages (e.g., Japanese and Korean).

With this preliminary result, we motivate further exploration of language similarity and its effect on X-MOD and related models.

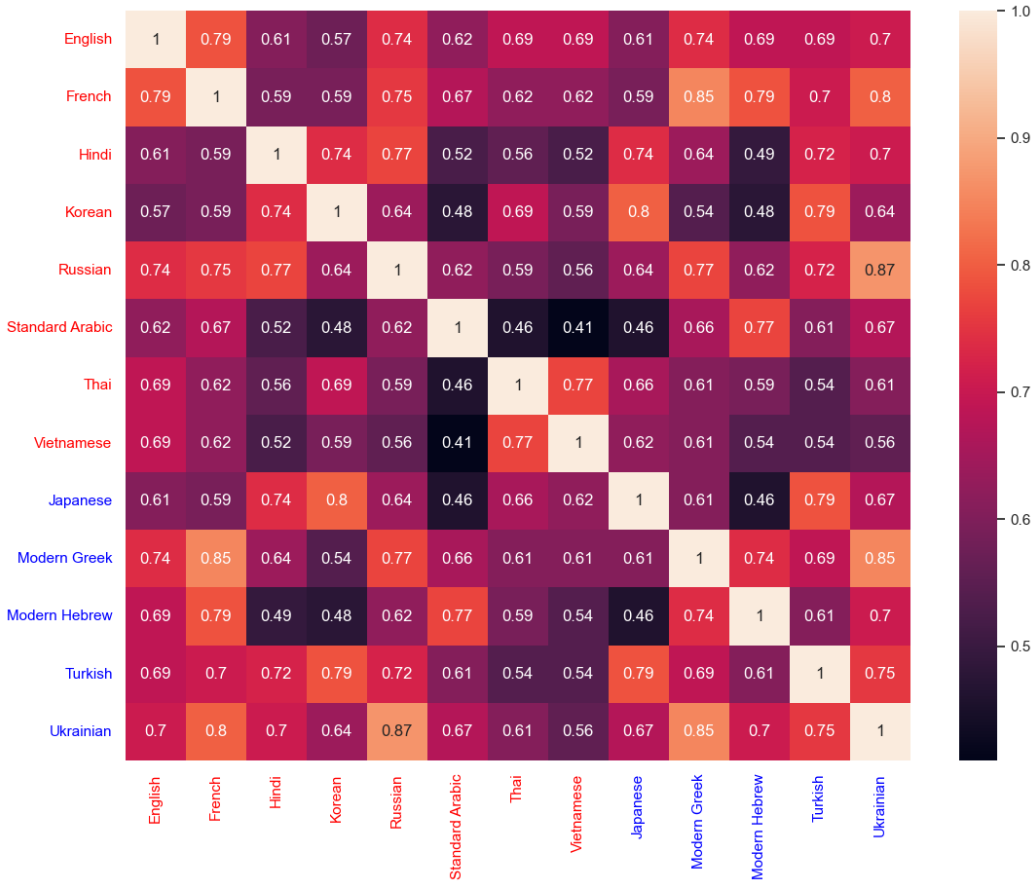

Figure 4.6: Language similarity scores calculated using a collection of GramBank Skirgård et al. [2023] features.

Bibliography

Orevaoghene Ahia, Julia Kreutzer, and Sara Hooker. The low-resource double bind: An empirical study of pruning for low-resource machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3316–3333, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.282. URL https://aclanthology.org/2021.findings-emnlp.282.

Jesujoba O. Alabi, David Ifeoluwa Adelani, Marius Mosbach, and Dietrich Klakow. Adapting pre-trained language models to African languages via multilingual adaptive fine-tuning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4336–4349, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.382.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1042. URL https://aclanthology.org/P17-1042.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Translation artifacts in cross-lingual transfer learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7674–7684, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.618. URL https://aclanthology.org/2020.emnlp-main.618.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online, July 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.421. URL https://aclanthology.org/2020.acl-main.421.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32, 2019.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1269. URL https://aclanthology.org/D18-1269.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747.

Ameet Deshpande, Partha Talukdar, and Karthik Narasimhan. When is BERT multilingual? isolating crucial ingredients for cross-lingual transfer. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3610–3623, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.264. URL https://aclanthology.org/2022.naacl-main.264.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Philipp Dufter and Hinrich Schütze. Identifying elements essential for bert's multilinguality. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4423–4437, 2020.

Abteen Ebrahimi, Manuel Mager, Arturo Oncevay, Vishrav Chaudhary, Luis Chiruzzo, Angela Fan, John Ortega, Ricardo Ramos, Annette Rios, Ivan Vladimir Meza Ruiz, Gustavo Giménez-Lugo, Elisabeth Mager, Graham Neubig, Alexis Palmer, Rolando Coto-Solano, Thang Vu, and Katharina Kann. AmericasNLI: Evaluating zero-shot natural language understanding of pretrained multilingual models in truly low-resource languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6279–6299, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.435. URL https://aclanthology.org/2022.acl-long.435.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. The state and fate of linguistic diversity and inclusion in the nlp world. *arXiv preprint arXiv:2004.09095*, 2020.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.653. URL https://aclanthology.org/2020.acl-main.653.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5747–5763, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.463. URL https://aclanthology.org/2020.emnlp-main.463.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020b. doi: 10.1162/tacl_a_00343. URL https://aclanthology.org/2020.tacl-1.47.

Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

Antonis Maronikolakis, Philipp Dufter, and Hinrich Schütze. Wine is not v i n. on the compatibility of tokenizations across languages. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2382–2399, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.205. URL https://aclanthology.org/2021.findings-emnlp.205.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*, 2020.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://aclanthology.org/N19-4009.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1178. URL https://aclanthology.org/P17-1178.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June

2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.617. URL https://aclanthology.org/2020.emnlp-main.617.

Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. Lifting the curse of multilinguality by pre-training modular transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.255. URL https://aclanthology.org/2022.naacl-main.255.

Jonas Pfeiffer, Francesco Piccinno, Massimo Nicosia, Xinyi Wang, Machel Reid, and Sebastian Ruder. mmt5: Modular multilingual pre-training solves source language hallucinations. *arXiv preprint arXiv:2305.14224*, 2023.

Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1493. URL https://aclanthology.org/P19-1493.

Martin Popel and Ondřej Bojar. Training tips for the transformer model. *arXiv preprint arXiv:1804.00247*, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Afshin Rahimi, Yuan Li, and Trevor Cohn. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1015. URL https://aclanthology.org/P19-1015.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2021.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. Square one bias in NLP: Towards a multi-dimensional exploration of the research manifold. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2340–2354, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.184. URL https://aclanthology.org/2022.findings-acl.184.

Hedvig Skirgård, Hannah J Haynie, Damián E Blasi, Harald Hammarström, Jeremy Collins, Jay J Latarche, Jakob Lesage, Tobias Weber, Alena Witzlack-Makarevich, Sam Passmore, et al. Grambank reveals the importance of genealogical constraints on linguistic diversity and highlights the impact of language loss. *Science Advances*, 9(16):eadg6175, 2023.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *International Conference on Learning Representations*, 2016.

Sofía Flores Solórzano. Corpus oral pandialectal de la lengua bribri, 2017. URL https://bribri.net.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Zihan Wang, Stephen Mayhew, and Dan Roth. Cross-lingual ability of multilingual bert: An empirical study. In *International Conference on Learning Representations*.

Zihan Wang, Stephen Mayhew, Dan Roth, et al. Cross-lingual ability of multilingual bert: An empirical study. *arXiv preprint arXiv:1912.07840*, 2019.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.494.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101.

Shijie Wu and Mark Dredze. 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, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1077. URL https://aclanthology.org/D19-1077.

Ningyu Xu, Tao Gui, Ruotian Ma, Qi Zhang, Jingting Ye, Menghan Zhang, and Xuanjing Huang. Cross-linguistic syntactic difference in multilingual BERT: How good is it and how does it affect transfer? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8073–8092, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.552.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 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, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.41. URL https://aclanthology.org/2021.naacl-main.41.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

List of Figures

List of Tables

Appendix A

A.1 Language Similarity Features

Feature Question
Are there definite or specific articles?
Are there prenominal articles?
Are there postnominal articles?
What is the order of numeral and noun in the NP?
What is the order of adnominal demonstrative and noun?
Is there a gender distinction in independent 3rd person pronouns?
Is there a dual or unit augmented form (in addition to plural or augmented) for all person categories in the pronoun system?
Are there three or more distance contrasts in demonstratives?
Do demonstratives show an elevation distinction?
Do demonstratives show a visible-nonvisible distinction?
Is there productive overt morphological singular marking on nouns?
Is there productive morphological dual marking on nouns?
Is there productive morphological plural marking on nouns?
Is there a productive morphological pattern for deriving an action/state noun from a verb?
Is there a productive morphological pattern for deriving an agent noun from a verb?
Is there a productive morphological pattern for deriving an object noun from a verb?
Is there a gender/noun class system where sex is a factor in class assignment?
Is there a gender/noun class system where shape is a factor in class assignment?
Is there a gender/noun class system where animacy is a factor in class assignment?
Is there a gender/noun class system where plant status is a factor in class assignment?
Are there numeral classifiers?

Table A.1: Features used for language similarity analysis. Part I

Feature Question
What is the pragmatically unmarked order of adnominal possessor noun and possessed noun?
Are there morphological cases for non-pronominal core arguments (i.e. S/A/P)?
Are there morphological cases for pronominal core arguments (i.e. S/A/P)?
Are there morphological cases for oblique non-pronominal NPs (i.e. not S/A/P)?
Are there prepositions?
Are there postpositions?
Is there productive infixation in verbs?
Is there overt morphological marking of present tense on verbs?
Is there overt morphological marking on the verb dedicated to past tense?
Is there overt morphological marking on the verb dedicated to future tense?
Can the S argument be indexed by a suffix/enclitic on the verb in the simple main clause?
Can the S argument be indexed by a prefix/proclitic on the verb in the simple main clause?
Can the A argument be indexed by a suffix/enclitic on the verb in the simple main clause?
Can the A argument be indexed by a prefix/proclitic on the verb in the simple main clause?
Can the P argument be indexed by a suffix/enclitic on the verb in the simple main clause?
Can the P argument be indexed by a prefix/proclitic on the verb in the simple main clause?
Are variations in marking strategies of core participants based on TAM distinctions?
Are variations in marking strategies of core participants based on verb classes?
Are variations in marking strategies of core participants based on person distinctions?
Is there a benefactive applicative marker on the verb (including indexing)?
Is there an instrumental applicative marker on the verb (including indexing)?
Can standard negation be marked by an affix, clitic or modification of the verb?
Is there directional or locative morphological marking on verbs?
Is there a phonologically bound reflexive marker on the verb?
Is there a phonologically bound reciprocal marker on the verb?

Table A.2: Features used for language similarity analysis. Part II

Feature Question
Do verbs classify the shape, size or consistency of absolutive arguments by means of incorporated nouns, verbal affixes or suppletive verb stems?
Are there serial verb constructions?
Is verb compounding a regular process?
Is incorporation of nouns into verbs a productive intransitivizing process?
Is there an existential verb?
What is the pragmatically unmarked order of S and V in intransitive clauses?
Is a pragmatically unmarked constituent order verb-initial for transitive clauses?
Is a pragmatically unmarked constituent order verb-medial for transitive clauses?
Is a pragmatically unmarked constituent order verb-final for transitive clauses?
Is there a morphological antipassive marked on the lexical verb?
Is there a morphologically marked inverse on verbs?
Is there clause chaining?

Table A.3: Features used for language similarity analysis. Part III

Language	Sample sentence
aym	XVI Benedict Papa-n Santiago de Cuba-r purïwipax Caribeño markaruw Twitter Trending Topics-aruw uchi.
bzd	Ie' stsòq bua'.
cni	Imokaatiro Kamaiteri jantari rantantyaari ibito.
gn	Upépe, ambue mitârusu ñane retâyguáva ndive, Maneco omoheñói peteî mba'epu'aty ha'e opuraheiháp e ha ombopuháp e mbaraka.
hch	meta y+wa+ka m+pa+ mepei yuriekai,
nah	¿ Tlen tonalli yalhua ?
oto	ra B'ondo M'onda , nuua
quy	Nitaq niychu llakinmanta haykapikama willakunanmantapas.
shp	Moátianronqui huáiroroboqui joshín sáhuetiabo icáticanai.
tar	we apechia eyena kepu marí

Table A.4: Samples from each language of the monolingual corpus used for further MLM adaptation used by the AmericasNLI [Ebrahimi et al., 2022] team.