Cross-lingual Knowledge Transfer in Multi-lingual Language Models

A Seminar Report

Submitted in partial fulfillment of requirements for the degree of

Master of Science By Research

By

Soumen Kumar Mondal
Roll No.: 23m2157

under the guidance of

Prof. Preethi Jyothi

Center for Machine Intelligence and Data Science
Indian Institute of Technology, Bombay

May 8, 2024

Acknowledgment

I am deeply thankful to my advisor, **Prof. Preethi Jyothi**, for her unwavering guidance and support during my studies. I also extend my appreciation to all the faculty members who have contributed to my education. Their teachings have enriched me with invaluable knowledge, skills, and values. Additionally, I owe a great deal of thanks to my family and friends for their enduring support and encouragement throughout my academic pursuits.

Abstract

This report explores the field of cross-lingual knowledge transfer within multilingual language models, highlighting advancements and challenges in the domain. We begin by defining the problem of cross-lingual knowledge transfer and discuss the motivation behind this research. We delve into various fine-tuning techniques that enhance the performance of these models on specific tasks and languages without compromising their general applicability. These techniques include generalizable fine-tuning, composable sparse fine-tuning, and adapter-based frameworks like MAD-X. The discussion extends to how facts are represented within these models, with a focus on task-specific model editing and the geometry of multilingual representations. Furthermore, the report examines mechanisms for cross-lingual transfer, emphasizing the role of knowledge editing and cross lingual entity alignment that support transfer learning. The conclusion summarizes key findings and proposes future research directions aimed at overcoming current limitations and enhancing the efficacy of cross-lingual knowledge transfer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

In the evolving field of natural language processing (NLP), the ability of multilingual language models to effectively transfer knowledge across different languages remains a significant challenge. Despite the rapid advancements in language model technology, including attention mechanisms and multilingual model development, there are inherent limitations in how these models manage and apply cross-lingual knowledge. This challenge is compounded by the complexity of adapting these models to perform optimally across diverse linguistic contexts.

The problem extends into the fine-tuning processes and the representation of facts within these models, which are crucial for maintaining accuracy and relevance in multilingual applications. The fundamental issue revolves around developing robust methodologies that can enhance the efficacy of knowledge transfer in multilingual settings, ensuring that these models can deliver consistent and reliable performance across all languages they are designed to support.

1.2 Motivation

The internet is full of information in many different languages. However, most language technology today works best with English, leaving those who speak other languages with fewer benefits. This is a big problem because everyone should have equal access to the advancements in artificial intelligence, no matter what language they speak. While monolingual models have reached impressive levels of performance, they inherently lack the ability to operate beyond their target linguistic boundaries. This limitation underscores the critical need for robust multilingual models capable of understanding and processing multiple languages simultaneously.

Multilingual language models, such as mBERT and XLM-R, have emerged as important tools in breaking these language barriers, facilitating a more inclusive digital ecosystem. However, the full potential of these models is yet to be realized, particularly in contexts involving low-resource languages or dialects that are typically underrepresented in data used for training these models. This research is motivated by the pursuit to harness and enhance the capabilities of these models to not only understand but also effectively transfer knowledge across languages.

Furthermore, the dynamic and evolving nature of language necessitates continuous advancements in language models to adapt to new vocabularies, expressions, and usage patterns. This

evolving landscape presents a unique opportunity to explore innovative cross-lingual transfer mechanisms that can significantly improve the adaptability and scalability of language technologies.

By addressing these challenges, this seminar aims to contribute to the development of more generalized, efficient, and inclusive language technologies that are accessible to users worldwide, regardless of language. This motivation drives the exploration of advanced fine-tuning techniques and novel architectural frameworks that promote effective knowledge transfer across linguistic divides, ultimately enhancing the practical utility of multilingual language models in real-world applications.

1.3 Seminar Outline

The rest of the report is structured as follows: **Chapter 2: Background and Foundations** provides an overview of language models, including the development of attention mechanisms and the evolution of language models, as well as the application of multilingual models such as mBERT and XLM-R. **Chapter 3: Language Model Fine-Tuning** explores various fine-tuning techniques that enhance model performance for specific tasks and languages while maintaining general applicability, featuring methods like composable sparse fine tuning, and adapter-based frameworks like MAD-X. **Chapter 4: Fact Representation in Language Models** focuses on the integration and manipulation of factual content within these models, exploring task-specific model editing and the geometry of multilingual representations. **Chapter 5: Cross-Lingual Transfer Mechanisms** examines strategies for knowledge transfer and editing across languages and discusses cross lingual entity alignment for transfer learning. Finally, **Chapter 6: Conclusion and Future Directions** summarizes the key findings and proposes future research directions to address existing challenges and enhance the efficacy of cross-lingual knowledge transfer.

Chapter 2

Background and Foundations

2.1 Overview of Language Models

2.1.1 Evolution of Language Models

Initially, LMs were based on n-gram models and simple neural networks, which were limited by their inability to capture long-range dependencies in text. The introduction of recurrent neural networks (RNNs), particularly Long Short-Term Memory (LSTM) networks by Hochreiter and Schmidhuber [1997], marked a significant advancement, enabling models to remember information over extended sequences. The shift towards attention-based models, particularly the Transformer architecture, has set the current state-of-the-art, enabling the training of large-scale language models like GPT (Generative Pre-trained Transformer) by Radford et al. [2018] and BERT (Bidirectional Encoder Representations from Transformers) by Devlin et al. [2018], which leverage vast amounts of data and computational power to achieve improved performance across a range of language tasks.

2.1.2 Attention Mechanism

The attention mechanism improved language modeling by allowing models to focus on different parts of the input sequence when predicting an output, improving the context sensitivity of the model outputs. Introduced by Bahdanau et al. [2014] in the context of neural machine translation, the mechanism addresses the limitations of earlier sequence-to-sequence models by selecting a subset of the input tokens to pay attention to. This concept was further extended by Vaswani et al. [2017] through the Transformer model, which employs self-attention to weigh the significance of all tokens in the input sequence relative to each other, leading to significant improvements in processing speed and model performance.

2.2 Development of Multilingual Models

2.2.1 mBERT

Multilingual BERT (mBERT) by Devlin et al. [2018] is one of the first large-scale multilingual models that has achieved notable success in cross-lingual understanding. mBERT is a variant of the original BERT model that was trained on the concatenated text corpora of 104 languages from Wikipedia. mBERT utilizes the same model architecture as the original BERT but extends its application to multiple languages without modifying the underlying transformer architecture.

Despite not being explicitly trained to model cross-lingual relationships, mBERT has shown remarkable zero-shot learning capabilities, where a model trained on data in one language performs well on the same task in other languages. This phenomenon is attributed to the shared subword vocabulary and the model's ability to learn language-agnostic representations during training.

2.2.2 XLM-R

Expanding on the idea of multilingual language modeling, Conneau et al. [2020] introduced XLM-R (Cross-lingual Language Model-RoBERTa), which builds upon the RoBERTa model by Liu et al. [2019], an optimized version of BERT that modifies key hyperparameters. XLM-R was trained on 2.5TB of filtered CommonCrawl data across 100 languages, making it one of the most comprehensive and diverse datasets used in multilingual model training to date. Unlike mBERT, XLM-R uses a more robust training regime and a larger, more diverse dataset, leading to significant improvements in multilingual performance, particularly in low-resource languages. XLM-R has demonstrated state-of-the-art performance on a variety of cross-lingual benchmarks, highlighting its effectiveness in handling diverse linguistic features and its robustness in language transfer tasks.

Chapter 3

Language Model Fine-Tuning

3.1 Adapter-Based Fine Tuning

Adapters: Adapter-based fine tuning allows for the efficient transfer of a model to multiple tasks or languages by adding small, task-specific modules called adapters to a fixed, pre-trained model. Adapters are typically small neural networks inserted between the layers of a pre-trained model as shown in Figure 3.1. Each adapter only learns task-specific or language-specific features, leaving the original model weights untouched. This method is particularly advantageous for multilingual models because it enables the customization of a single foundational model for a variety of languages and tasks without the need for extensive retraining. The concept of adapter-based tuning was popularized by Houlsby et al. [2019], who introduced a simple yet effective architecture for adapters as shown in Figure 3.1.

MAD-X for Cross-Lingual Transfer: Adapters have proven especially useful in multilingual settings, such as in the work by Pfeiffer et al. [2020], who extended the adapter approach to cross-lingual models with their MAD-X framework. MAD-X enables efficient transfer learning across languages by leveraging language-specific adapters, which capture unique linguistic characteristics of each language while sharing a common, language-agnostic model backbone.MAD-X (Multiple ADapters for Cross-lingual transfer) represents a significant evolution in the field of cross-lingual transfer, specifically designed to enhance the capabilities of pre-trained multilingual models like mBERT and XLM-R. This framework introduces a modular approach using small, additional parameter-efficient adapters to improve the model's ability to handle multiple languages and tasks efficiently.

Language Adapters: Language adapters as shown in Figure 3.2 are designed to adapt the model to the specificities of a given language. These adapters are trained using Masked Language Modeling (MLM) on unlabelled data from the target language. The language adapter at each layer l of the transformer can be mathematically described by the following equation:

$$\mathrm{LA}_l(h_l, r_l) = U_l(\mathrm{ReLU}(D_l(h_l))) + r_l \tag{3.1}$$

where:

- h_l is the output from the previous layer or the initial input.
- r_l is the residual connection from the previous transformer block.
- D_l and U_l are the down-projection and up-projection matrices respectively at layer l.

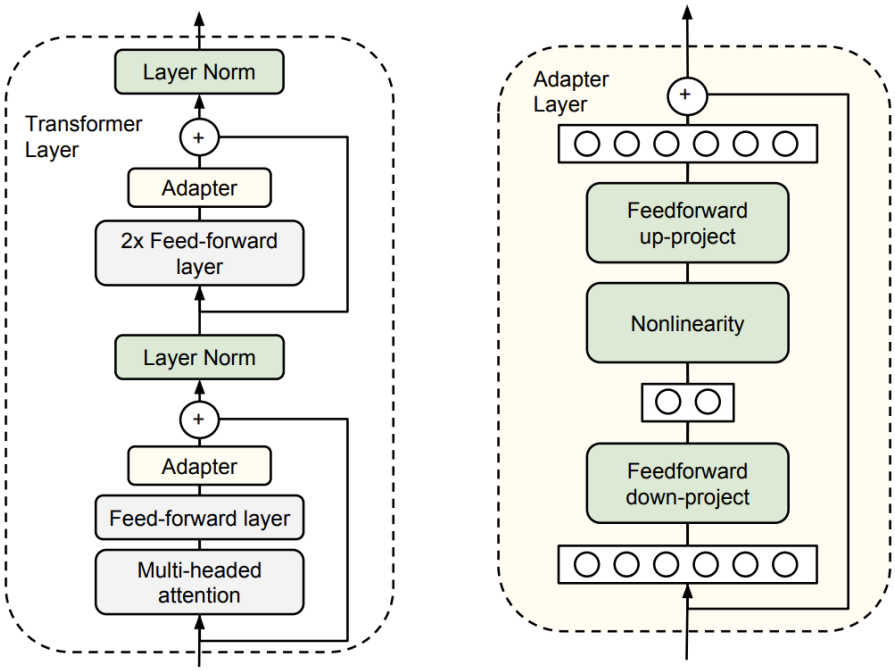

Figure 3.1: Adapter Module Integration in the Transformer Architecture: On the left, adapter modules are incorporated into each Transformer layer twice: once following the multi-headed attention and once after the feed-forward layers. On the right, each adapter features a bottleneck design with relatively few parameters compared to the main attention and feed-forward components of the Transformer. Each adapter includes a skip-connection for efficient training. During the process of adapter tuning, only the components shown in green, which include the adapters, layer normalization parameters, and the final classification layer (not depicted), are updated based on the downstream task data.(Houlsby et al. [2019])

- ReLU is the non-linear activation function.

Task Adapters: Task adapters as shown in Figure 3.2, on the other hand, are used to fine-tune the model for a specific task. They are applied after the language adapters and can be described by the following equation for each layer l:

$$\mathrm{TA}_l(h_l, r_l) = U_l(\mathrm{ReLU}(D_l(\mathrm{LA}_l(h_l, r_l)))) + r_l \tag{3.2}$$

where:

- $\mathrm{LA}_l(h_l, r_l)$ is the output from the language adapter.
- The rest of the parameters function similarly to those in the language adapters.

Applications: Results by Pfeiffer et al. [2020] demonstrate that MAD-X significantly outperforms existing state-of-the-art methods in cross-lingual transfer tasks. MAD-X delivers robust results not only in high-resource languages but also shows effectiveness in low-resource and unseen language scenarios.

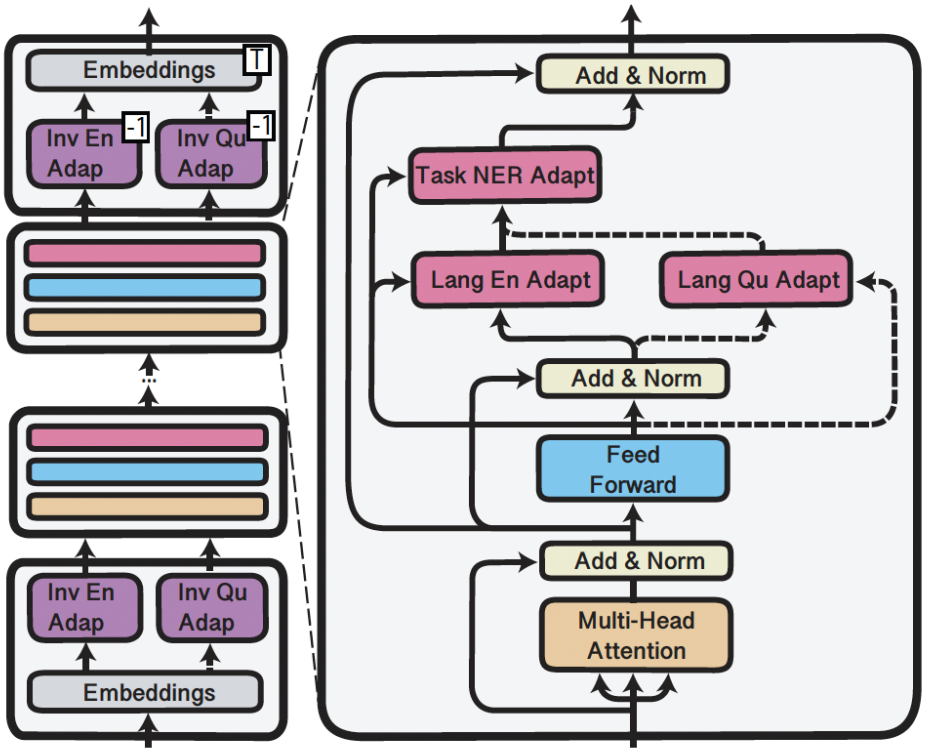

Figure 3.2: Overview of the MAD-X Framework in a Transformer Model: Input data first passes through an invertible adapter, the output of which feeds into the model's output layer. Within each Transformer layer, language and task-specific adapters are incorporated. The language adapters, along with the invertible adapter, are trained using masked language modeling (MLM) while keeping the core multilingual model unchanged. For tasks like Named Entity Recognition (NER), task-specific adapters are added on top of the language adapters for training (illustrated with solid lines). For zero-shot cross-lingual transfer, adapters for the source language are swapped with those for the target language (shown with dashed lines).(Pfeiffer et al. [2020])

Limitations: Despite its advantages, MAD-X faces several challenges:

- **Model Capacity:** Pre-trained models may lack sufficient capacity to adequately represent all languages, especially those not included in the training data, leading to poorer performance in such languages.
- **Dependency on Source Language Quality:** The success of transfer learning heavily relies on the quality and comprehensiveness of the source language models, which can vary significantly.

3.2 Generalizable Fine Tuning

Child-Tuning: Child-Tuning proposed by Xu et al. [2021] is fine-tuning technique designed to adapt large pretrained language models (PLMs) to specific tasks by updating only a subset of the model's parameters. This approach helps to prevent overfitting and improves generalization, especially when training data is limited. Child-Tuning updates a selectively identified subset of parameters, referred to as the *child network*, while keeping the rest of the model's parameters (the *non-child network*) fixed. This selective updating is achieved by applying a mask to the gradients

during the backward pass, effectively zeroing out the gradients for the non-child network. The standard update rule in gradient descent is given by:

$$w_{t+1} = w_t - \eta \frac{\partial L}{\partial w_t} \tag{3.3}$$

where w_t represents the model parameters at iteration t, η is the learning rate, and $\frac{\partial L}{\partial w_t}$ is the gradient of the loss function with respect to the parameters. In Child-Tuning, the update rule is modified to only apply to the child network, which is defined by a binary mask M_t. This mask is applied to the gradients, allowing only a subset of parameters to be updated:

$$w_{t+1} = w_t - \eta(\frac{\partial L}{\partial w_t} \odot M_t) \tag{3.4}$$

Here, $\odot$ denotes element-wise multiplication. The mask M_t has entries of 1 for parameters belonging to the child network and 0 for others.

Detection of the Child Network: The child network can be detected in two ways:

- **Task-Free**: Parameters are selected based on a predefined distribution, often without specific task data. This can be seen as adding a form of regularization to prevent overfitting.

- **Task-Driven**: Parameters are selected based on their relevance to the specific task, typically identified through methods like Fisher Information, which measures how much information a parameter carries about the output.

Types of Child Tuning: The Child-Tuning methodology introduces two variants: Child-Tuning-F and Child-Tuning-D. The former operates independently of specific task data, using a Bernoulli distribution to randomly select the child network, thereby introducing stochasticity that acts as a form of regularization. This variant aims to improve generalization by preventing overfitting to smaller datasets. On the other hand, Child-Tuning-D utilizes task-related data to identify the parameters most crucial for the task, freezing the remainder of the model to its pre-trained state. This approach not only tailors the model more closely to the task requirements but also preserves the generalization capabilities inherent in the original pre-trained model.

Advantages: Child-Tuning offers several significant benefits:

- **Prevention of Overfitting:** By updating only a subset of parameters, Child-Tuning prevents the model from overly adapting to the training data, thus maintaining robustness and preventing overfitting. This method has shown to improve the model's ability to generalize to new, unseen tasks and domains, improving its performance across a variety of downstream applications .

- **Efficient Utilization of Model Capacity:** Child-Tuning makes efficient use of a model's capacity by focusing updates on parameters that are most influential for the task at hand, therefore reducing the computational resources required for training.

Limitations Despite its advantages, Child-Tuning also encounters certain challenges:

- **Dependence on Accurate Parameter Selection:** The success of Child-Tuning heavily relies on accurately identifying which parameters to include in the child network. Incorrect selections can lead to suboptimal performance.
- **Reduced Adaptability:** Since only a subset of parameters is updated, there might be scenarios where the non-updated parameters limit the model's adaptability to tasks significantly different from those seen during training.

3.3 Composable Sparse Fine Tuning

Composable Language and Task Vectors: Composable Sparse Fine-Tuning (CSFT) by Ansell et al. [2023] enhances the adaptability of multilingual models by integrating task-specific and language-specific adaptations through the use of composable vectors. This method enables models to efficiently handle multiple languages and tasks without requiring separate models for each combination. CSFT operates by defining separate vectors for language and task adaptations, which are then combined to adjust the base model parameters for specific cross-lingual and task-oriented performance. The language vector, ϕ_{lang}, and the task vector, ϕ_{task}, are defined as the changes required to adapt the base model parameters $\theta^{(0)}$ for a specific language and task, respectively:

$$\phi_{\text{lang}} = \theta_{\text{lang}} - \theta^{(0)} \tag{3.5}$$

$$\phi_{\text{task}} = \theta_{\text{task}} - \theta^{(0)} \tag{3.6}$$

where θ_{lang} and θ_{task} are the parameters of the model fine-tuned for the specific language and task. To achieve simultaneous adaptation to both language and task, the vectors are combined through element-wise addition:

$$\phi_{\text{combined}} = \phi_{\text{lang}} + \phi_{\text{task}} \tag{3.7}$$

This combined adaptation vector is then used to update the base model parameters:

$$\theta_{\text{final}} = \theta^{(0)} + \phi_{\text{combined}} \tag{3.8}$$

The final model, represented as $F(\cdot; \theta_{\text{final}})$, is thus equipped to perform the specific task in the specific language, utilizing the enhancements from both the language and task vectors as shown in Figure 3.3.

Lottery Ticket Hypothesis: Composable Sparse Fine-Tuning (CSFT) technique is inspired by the Lottery Ticket Hypothesis by Frankle and Carbin [2019]. The Lottery Ticket Hypothesis states that within a dense neural network, there exists a smaller, sparse subnetwork (referred to as a "winning ticket") that, when trained in isolation from the beginning, can achieve comparable or even superior performance to the original, larger network. This hypothesis suggests that by identifying and training these efficient subnetworks, one can significantly reduce the computational resources required without sacrificing performance. Lottery Ticket Sparse Fine-Tuning (LT-SFT) introduces a technique inspired by the Lottery Ticket Hypothesis (LTH), specifically for multilingual models.

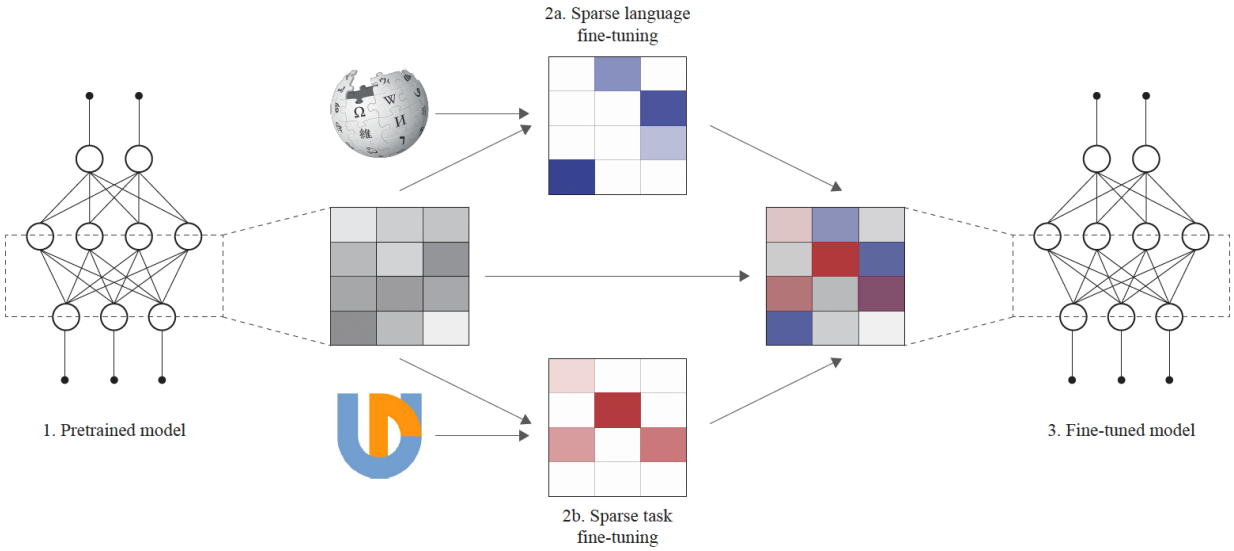

Figure 3.3: Illustration of Composable Sparse Fine-Tuning Process: Starting with the parameters of a pre-trained model shown in gray (left), we introduce task-specific and language-specific sparse fine-tunings, depicted in blue and red respectively (center). These components are then combined through addition (right) to produce the final adapted model. (Ansell et al. [2023])

Fine Tuning Process: The core idea behind LT-SFT is to select a subset of parameters that exhibit significant changes during initial training and fine-tune only these parameters in subsequent phases as per the LTH. LT-SFT involves two main phases:

1. **Parameter Selection:** Initially, the full model parameters $\theta^{(0)}$ are trained on target data, resulting in updated parameters $\theta^{(1)}$. Parameters are then ranked based on the absolute change in their values, and a binary mask μ is constructed to identify the top K parameters, where $\mu_i = 1$ if the i-th parameter is selected and 0 otherwise.

2. **Sparse Fine-Tuning:** Parameters are reset to their original values $\theta^{(0)}$ and only those marked by μ are updated in the subsequent training. This selective training uses the masked gradient $\mu \odot \nabla_\theta L(F(\cdot;\theta), D)$, where $\odot$ denotes element-wise multiplication and L is the loss function.

The sparse fine-tuning can be mathematically represented as:

$$\phi = \theta^{(2)} - \theta^{(0)} \tag{3.9}$$

where $\theta^{(2)}$ are the parameters after sparse fine-tuning. The adaptation for a task can then be expressed as a function of the base model:

$$F(\cdot;\theta + \phi) \tag{3.10}$$

This formulation allows the original model to adapt effectively to new tasks and languages by updating only a crucial subset of parameters, thereby maintaining overall model efficiency and preventing overfitting.

Applications: CSFT integrates the modularity of adapters with the expressivity of sparse fine-tuning, allowing for fine-grained control over which parts of the model are updated during

training. One of the key advantage of CSFT is the ability to compose these sparse tunings, which enables the application of multiple, distinct adaptations to a single model without the overhead of additional parameters at inference time. For instance, a model can be fine-tuned with a language-specific sparse tuning followed by a task-specific sparse tuning, effectively adapting the model for a specific task in a new language. The effectiveness of CSFT has shown significant performance improvements over existing methods, particularly in zero-shot cross-lingual settings.

Limitations: Despite numerous benefits in terms of efficiency and modularity, CSFT also faces several limitations that may affect its applicability and effectiveness in various scenarios.

- **Initial Parameter Selection:** The effectiveness of CSFT heavily relies on the correct identification of parameters that are most impactful for a given task during the initial sparse selection phase. A poor choice in this step can lead to suboptimal performance and fail to utilize the model's pre-trained weights.
- **Adaptability Concerns:** The sparse nature of the fine-tuning in CSFT may limit the model's adaptability to new, unseen tasks or languages that were not part of the initial training set. This is because only a subset of the model's parameters are updated, therefore leaving out critical parameters needed for adapting to new challenges.
- **Risk of Overfitting:** There is an inherent risk of overfitting in CSFT, especially when the fine-tuned model is exposed to small or non-representative datasets. The highly specific nature of the parameter updates can cause the model to overfit to the training data, thereby reducing its generalizability.

Chapter 4

Fact Representation in Language Models

4.1 Task-Specific Model Editing

Task Vector: Task-specific model editing technique by Ilharco et al. [2023] is centered around the concept of **task vectors**. A task vector τ_t in the context of neural networks is essentially a vector that captures the changes made to a pre-trained model's weights when it is fine-tuned to perform a specific task better. This vector is calculated by subtracting the weights of the pre-trained model from the weights of the model after it has been fine-tuned for the task as shown in Figure 4.1(a). Mathematically, the task vector τ_t for task t is given by:

$$\tau_t = \theta^t_{ft} - \theta_{pre} \tag{4.1}$$

Where:

- θ_{pre} are the weights of the pre-trained model.
- θ^t_{ft} are the weights of the model after fine-tuning for the specific task t.

This vector τ_t can then be used to adjust the model weights of another pre-trained model of the same architecture to improve its performance on task t, or combined with other task vectors to achieve multi-task learning or adjust the model's behavior in various ways, such as unlearning a task or mitigating undesirable behaviors.

Forgetting via Negation: Forgetting or unlearning a specific task can be achieved by negating the task vector as shown in Figure 4.1(b). Negating a task vector τ effectively reverses the direction of adjustment made by the original task training, thus degrading the model's performance on that task. Mathematically, this is represented as:

$$\tau_{new} = -\tau \tag{4.2}$$

This operation results in a model whose performance on the target task is decreased, with minimal impact on other tasks.

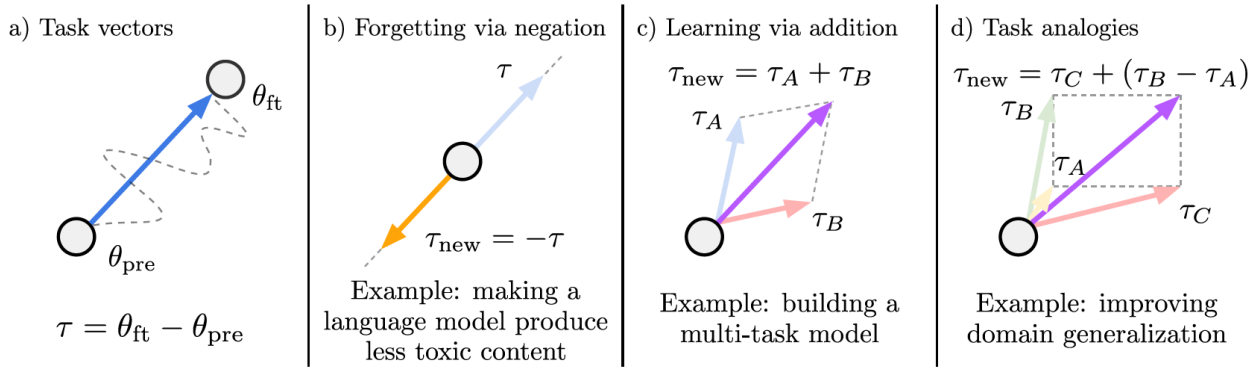

Figure 4.1: Task vectors and their use in model editing: (a) A task vector is calculated by subtracting the weight parameters of a pre-trained model from the weights after it has been fine-tuned on a specific task. (b) By negating a task vector, the model's performance on a particular task can be reduced without significantly affecting its performance on unrelated tasks. (c) A model's capabilities on multiple tasks can be enhanced by summing up their respective task vectors. (d) Task vectors can also be manipulated through analogical relationships between tasks, allowing improvements in one task based on the relationships derived from others (Ilharco et al. [2023]).

Learning via Addition: Learning new tasks or improving performance on multiple tasks can be facilitated by the addition of task vectors as shown in Figure 4.1(c). When task vectors $\tau_1, \tau_2, \ldots, \tau_n$ corresponding to different tasks are added, the resultant model is enhanced for all those tasks. The cumulative task vector is:

$$\tau_{new} = \sum_{i=1}^{n} \tau_i \tag{4.3}$$

Adding task vectors is especially effective for multitasking, allowing a single model to perform well across multiple tasks by leveraging the cumulative adjustments of each vector.

Task Analogies: Task analogies as shown in Figure 4.1(d) allow for the application of learned adjustments in new contexts, drawing on the analogy "A is to B as C is to D". For tasks A, B, C, and D, if the task vectors for A, B, and C are known, the task vector for D can be approximated as:

$$\tau_D = \tau_C + (\tau_B - \tau_A) \tag{4.4}$$

This approach utilizes differences and similarities across tasks to generalize to new tasks or domains without direct training data, effectively using transfer learning through task vector arithmetic.

Application: One of the application of task-specific model editing is in multi-task learning, where adding multiple task vectors can enhance the model's performance across several tasks simultaneously. This not only improves efficiency but also reduces the computational overhead associated with managing multiple distinct models. Furthermore, when tasks exhibit analogous relationships, combining appropriate task vectors can even improve performance on tasks for which the model has not been explicitly trained. This demonstrates the usefulness of this method in utilizing existing data to address data scarcity in new or related tasks.

Limitation By applying simple arithmetic operations on these vectors—such as addition, subtraction, or negation, the model's behavior can be effectively edited as shown in Figure 4.1. However, there are also several limitations to consider:

- **Architecture Dependency:** Task vectors are applicable only within models that share the same architecture. This is because task vectors involve element-wise operations on model weights, which assume a uniform structure across different model instances.
- **Same Pre-trained Initialization:** The effectiveness of task vector arithmetic is demonstrated under the condition that all models are fine-tuned from the same pre-trained initialization. This limits the applicability of task vectors since different initializations may lead to variations in how models learn and adapt, which could affect the applicability of the task vectors calculated from them.
- **Restriction to Linear Transformations**: The method primarily involves linear operations on task vectors. Non-linear interactions between model parameters or tasks may not be effectively captured or modified through simple vector arithmetic.

4.2 Cross-Lingual Fact Representation

Types of Fact Representation: Multilingual Language Models (ML-LMs) employ various strategies for the representation of factual knowledge across languages. Factual knowledge in ML-LMs is represented in three distinct ways as proposed by Zhao et al. [2024]: language-independent, cross-lingual shared, and cross-lingual transferred.

- **Language-Independent Representation:** This type of representation involves encoding factual knowledge using distinct neurons for each language. Each language has a unique set of neurons responsible for the representation of facts, independent of other languages. In some cases, the activity patterns of neurons differed when predicting the same fact in different languages. It means that the model's representation of this type of fact is not tied to any specific language as shown in Figure 4.2(a).
- **Cross-Lingual Shared Representation:** As shown in Figure 4.2(b), in the cross-lingual shared representation, ML-LMs use the same set of neurons to represent the same facts across multiple languages. This shared approach implies that a single neural configuration can encode factual knowledge in a way that is accessible and meaningful across different linguistic contexts, enhancing the model's ability to generalize across languages.
- **Cross-Lingual Transferred Representation:** This representation type involves transferring factual knowledge from one language to others, typically from a high-resource language to low-resource languages as shown in Figure 4.2(c). It is based on the fact that certain factual knowledge, once learned in one language, can be mapped onto other languages using a transformation of the learned representations. This method is particularly useful for efficiently scaling factual knowledge across languages in a model.

Effect of Training Data in Factual Probing: It's challenging to estimate the amount of distinct factual knowledge present in the training data of multi-lingual LMs. It has been found that out of 3 metrics – data-size of abstracts, number of page count and data-size of articles, the highest correlation (0.51) with P@1 is observed for data-size of abstracts. The precision tuple and data-size of abstracts tuple is shown in Figure 4.3.

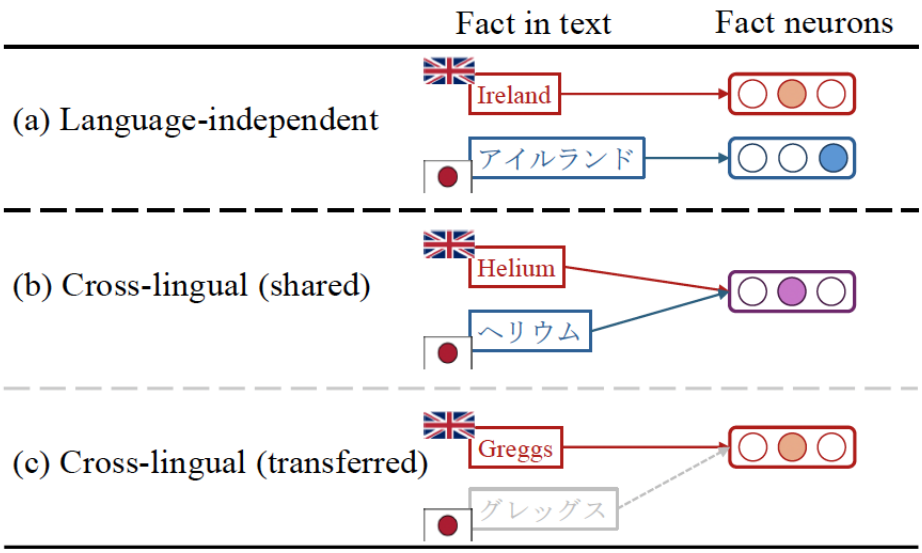

Figure 4.2: Cross-lingual fact representation (b, c) – same set of neurons are active for the same fact encoded in 2 different languages. Language independent representation (a) – different set of neurons are active for the same fact encoded in 2 different languages (Zhao et al. [2024]).

Formation of Cross-Lingual Fact Representation: Since the presence of cross-lingual fact representation is confirmed by neuron activity patterns by Zhao et al. [2024], it is important to find out whether they are shared or transferred as shown in Figure 4.2.

- **Tracing of Roots:** The goal of *tracing the roots of facts* is to check if a given fact originates from the training data (Wikipedia). To achieve this, string matching is performed between the object-subject pair and text (by WikiExtractor). If they co-occur, the fact is considered present, otherwise, it is considered as absent. These representations are identified through advanced probing methods using datasets like mLAMA, which consist of cloze-style queries designed to test the models' factual knowledge.
- **Absent yet Predictable Facts:** This involves assessing whether m-BERT correctly predicts the presence or absence of certain facts, even if they are not verifiable in the training data. Many of the facts that were absent in the knowledge source but correctly predicted were relatively easy to predict because of entity tokens and naming cues as shown in Table 4.1 and 4.2. The remaining facts other than entity tokens and naming cues are difficult to infer from the entities only, indicating the high possibility of cross-lingual transfer as shown in Table 4.3.

Language	Absent yet predictable facts	Naming cues
English	`The native language of Hamidou is French.`	`Hamidou`
Spanish	`Bruno solia comunicarse en frances.`	`Bruno`

Table 4.1: Examples of easy-to-predict facts using naming cues

Limitations: The probing results often reveal that while ML-LMs are capable of recalling facts in high-resource languages, their performance in low-resource languages can be inconsistent, primarily due to limited training data and potential cultural biases in the datasets. Despite its advantages, cross-lingual fact transfer faces challenges such as:

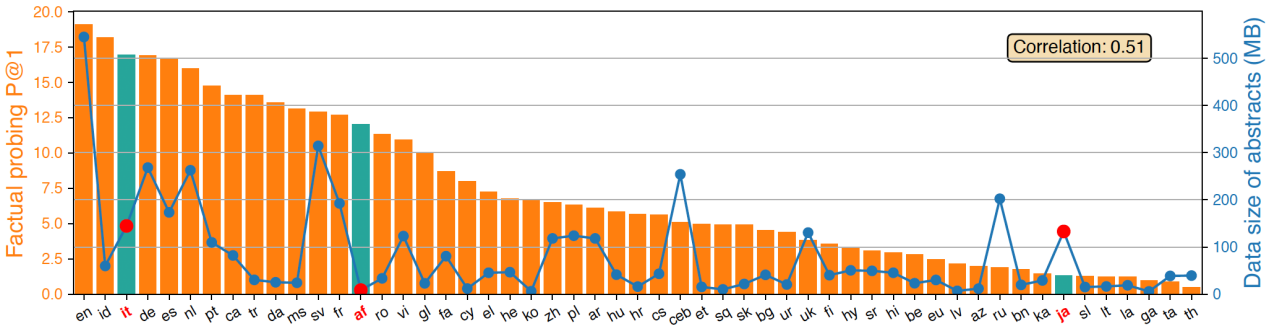

Figure 4.3: P@1 score for factual probing results in m-BERT model with full match approach. Italian and Japanese have P@1 score of 16.94% and 1.34% even though both of these languages are high resource with more than 100 MB of abstracts. This could be due to cultural biases in the mLAMA dataset, affecting non-Latin script languages (biases alone do not explain these substantial difference). Afrikaans language despite being low resource (<20 MB), shows high precision score (12.05%) for factual probing (Zhao et al. [2024]).

Language	Absent yet predictable facts	Entity tokens
English	`Sega Sports R&D is owned by Sega`	`Sega`
Spanish	`Honda Express es producido por Honda.`	`Honda`

Table 4.2: Examples of easy-to-predict facts using entity tokens.

Language	Absent yet predictable facts
English	`Aleksandar Novakovic was born in Belgrade`
Spanish	`Aleksandar Novakovic nacio en Belgrado`

Table 4.3: Examples of non-easy-to-predict facts.

- **Inconsistency Across Languages:** The transfer process might not always maintain accuracy due to linguistic and cultural differences.
- **Reliance on Source Languages:** The success of the transfer often depends heavily on the data quality and availability in the source languages.
- **Complex Representations:** The complexities involved in how different languages structure factual information can limit the effectiveness of the transfer.

4.3 Geometry of Language Model Representation

Language Subspace and Axes: Recent studies, such as those conducted by Chang et al. [2022], have explored how multilingual models like XLM-R maintain shared multilingual spaces while encoding language-sensitive information. Their findings illustrate that languages tend to occupy similar linear subspaces in high-dimensional embedding spaces, especially after processes like mean-centering, which aligns the subspaces by adjusting for language-specific offsets in the representation mean. The space contains embeddings or vectors assigned to tokens based on their semantic or syntactic properties. Subspaces are low-dimensional vector space within the high-dimensional embedding space that capture certain linguistic properties. Mean-centering involves subtracting the mean (over all subspace vectors) from each data point within a subspace.

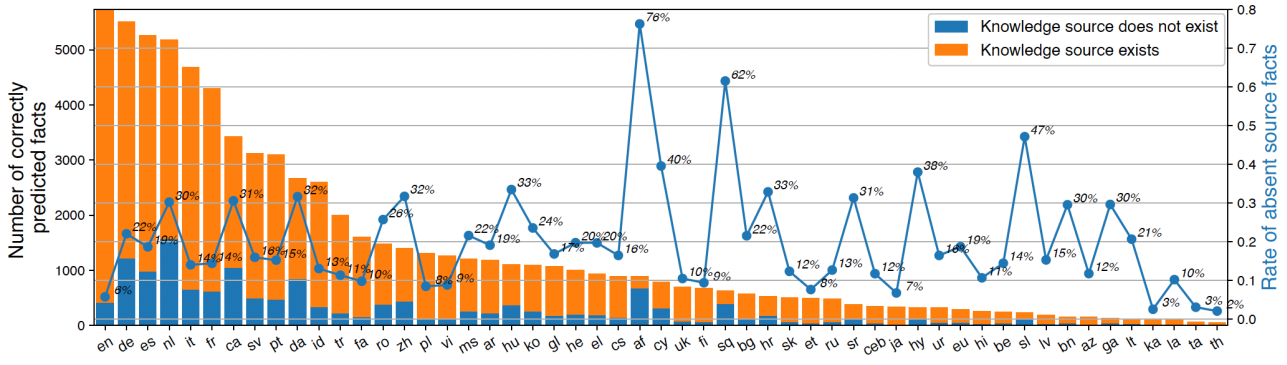

Figure 4.4: Number of correctly-predicted facts with mBERT in terms of existence of knowledge source: As more training data leads to a more comprehensive understanding of language and its associated facts, it is observed that languages with larger training data tend to have better coverage of factual knowledge as anticipated. However, facts in low resource languages like Afrikaans and Albanian are correctly predicted despite not being verifiable in the training corpus indicating a high possibility of effective cross-lingual transfer (Zhao et al. [2024]).

The language sensitive axes are axes (basic vectors) that are within the subspace that capture language-specific information (e.g. grammar). The language neutral axes are axes that are within the subspace encode information that is common across languages, such as word position or parts of speech.

Identification of subspace by SVD: For a particular *language* A, 512 input sentences are taken(each consists of 512 tokens) — therefore giving a total 262K contextual tokens.

$$\mathbf{c}^{(i)} \in \mathbb{R}^d \text{ where } i \in \{1, 2, \ldots, 262K\} \tag{4.5}$$

The mean context vector and data covariance matrix can be calculated as:

$$\boldsymbol{\mu}_A = \frac{1}{262K} \sum_{i=1}^{262K} \mathbf{c}^{(i)} \in \mathbb{R}^d \tag{4.6}$$

$$S = \frac{1}{262K} \sum_{i=1}^{262K} (\mathbf{c}^{(i)} - \boldsymbol{\mu}_A)(\mathbf{c}^{(i)} - \boldsymbol{\mu}_A)^T \in \mathbb{R}^{d \times d} \tag{4.7}$$

After performing the eigenvalue decomposition on S, the top k (corresponding to largest k eigenvalues) eigenvectors of S are given by $V_A \in \mathbb{R}^{d \times k}$. The language subspace is identified by k eigenvector of S. Chang et al. [2022] selected k such that $q/p = 0.9$. Considering the eigenvalues sequence of S in decreasing order as E, the ratio q/p can be calculated as follows:

$$E = (\lambda_1, \lambda_2, \ldots, \lambda_d) \text{ s.t. } \lambda_i \geq \lambda_{i+1} \quad \forall i \in \{1, 2, \ldots, d\} \tag{4.8}$$

$$q = \sum_{i=1}^{k} E[i] \tag{4.9}$$

$$p = \sum_{i=1}^{d} E[i] \tag{4.10}$$

The dimension of the context vector d was originally 768 and if the value of reduced dimension k is considered from each of the 12 layers of the transformer then the median value of k was found to be 335.

Perplexity Ratio: From the LLM, the token $\mathbf{t}^{(i)}$ is predicted given the previous sequence of tokens $(\mathbf{t}^{(j)})_{j=1}^{j=i-1}$. Suppose the LLM predicts the token $\mathbf{t}^{(i)}$ with probability $\mathbb{P}(\mathbf{t}^{(i)} \mid \mathbf{t}^{(i-1)}, \ldots, \mathbf{t}^{(1)})$. Then the perplexity of the LLM would be defined as:

$$pp(\mathbf{t}^{(i)}, \mathbf{t}^{(i-1)}, \ldots, \mathbf{t}^{(1)}) = \prod_{i=1}^{N} \left[\frac{1}{\mathbb{P}(\mathbf{t}^{(i)} \mid \mathbf{t}^{(i-1)}, \ldots, \mathbf{t}^{(1)})} \right]^{\frac{1}{N}} \tag{4.11}$$

Suppose $\mathbf{x}$ is a word vector which originally belongs to *language* A. Then projection of $\mathbf{x}$ after mean centering onto the language subspace of A gives $\mathbf{u}$. Again projecting this vector $\mathbf{u}$ onto the model space, the reconstructed word vector $\hat{\mathbf{x}}$ is obtained as:

$$\mathbf{u} = V_A^T(\mathbf{x} - \boldsymbol{\mu}_A) \tag{4.12}$$

$$\hat{\mathbf{x}} = V_A V_A^T(\mathbf{x} - \boldsymbol{\mu}_A) + \boldsymbol{\mu}_A \tag{4.13}$$

The perplexity of the LLM (with Equation 4.11) for all the languages represented in the original model space would be calculated as follows:

$$\text{Language-A:} \quad \mathbf{x}_A^{(i)[l]} \quad \forall i \in \{1, 2, \ldots, N\},\ \forall l \in \{1, 2, \ldots, 12\} \implies pp_A^{[l]} \tag{4.14}$$

The perplexity of the LLM (with Equation 4.11) for all the languages reconstructed from the language subspace would be calculated as follows:

$$\text{Language-A:} \quad \hat{\mathbf{x}}_A^{(i)[l]} = V_A^{[l]} V_A^{[l]^T}(\mathbf{x}_A^{(i)[l]} - \boldsymbol{\mu}_A^{[l]}) + \boldsymbol{\mu}_A^{[l]} \quad \forall i,\ \forall l \implies \hat{pp}_A^{[l]} \tag{4.15}$$

The perplexity ratio for each of the layer and for each of the language would be calculated as follows:

$$\text{Language-A:} \quad r_A^{[l]} = \frac{\hat{pp}_A^{[l]}}{pp_A^{[l]}} \quad \forall l \in \{0, 1, \ldots, 12\} \tag{4.16}$$

The average perplexity ratio over all the 88 languages for each of the layer would be calculated as follows:

$$r^{[l]} = \frac{1}{88} \sum_{i=A}^{CJ} r_i^{[l]} \quad \forall l \in \{0, 1, \ldots, 12\} \tag{4.17}$$

Justification – Affine subspaces can be used for language modeling:

- To reconstruct the vectors of a particular *language* A from its corresponding *language subspace* A, the following equation can be used.

$$Proj_A(\mathbf{x}_A) = V_A V_A^T(\mathbf{x}_A - \boldsymbol{\mu}_A) + \boldsymbol{\mu}_A \tag{4.18}$$

- From Figure 4.5 ($Proj_A$ curve), it is clear that when the vectors are reconstructed as per Equation 4.18 then the perplexity (calculated by $Proj_A(\mathbf{x}_A)$) does not increase significantly as compared to the no projection perplexity (calculated by $\mathbf{x}_A$).
- This suggests that the information necessary for language modeling task for a particular *language* A is well-preserved within the lower-dimensional *language subspace* A.

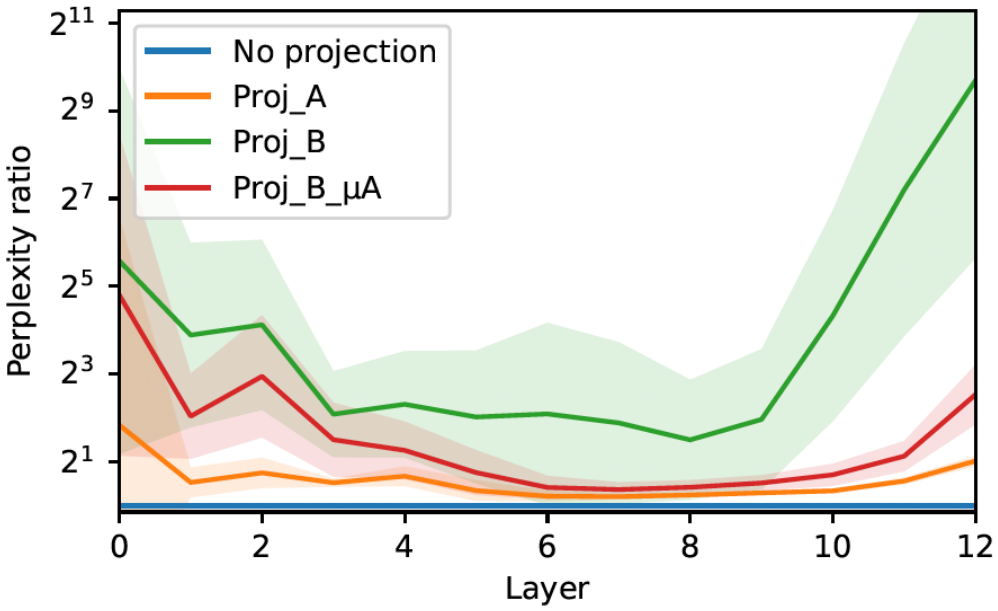

Figure 4.5: Variation of perplexity ratio at different transformer layers of LLM. (Chang et al. [2022])

Justification – Language subspaces differed from one another:

- To reconstruct the vectors of a particular *language A* from another *language subspace B*, the following equation can be used.

$$Proj_B(\mathbf{x}_A) = V_B V_B^T(\mathbf{x}_A - \boldsymbol{\mu}_B) + \boldsymbol{\mu}_B \tag{4.19}$$

- From Figure 4.5 ($Proj_B$ curve), it is clear that when the vectors are reconstructed as per Equation 4.19 then the perplexity (calculated by $Proj_B(\mathbf{x}_A)$) increases significantly as compared to the no projection perplexity (calculated by $\mathbf{x}_A$).

- This indicates that the *language subspace B* is not suitable for representing text in *language A*. This implies that the model has learned to differentiate between languages.

Justification – Mean-shifted subspaces were similar to one another:

- To reconstruct the vectors of a particular *language A* after mean centering from another *language subspace B*, the following equation can be used.

$$Proj_{B,\boldsymbol{\mu}_A}(\mathbf{x}_A) = V_B V_B^T(\mathbf{x}_A - \boldsymbol{\mu}_A) + \boldsymbol{\mu}_A \tag{4.20}$$

- From Figure 4.5 ($Proj_{B,\boldsymbol{\mu}_A}$ curve), it is clear that when the vectors are reconstructed as per Equation 4.20 then the perplexity (calculated by $Proj_{B,\boldsymbol{\mu}_A}$) does not increase significantly as compared to the no projection perplexity (calculated by $\mathbf{x}_A$).

- Projecting onto the affine subspace for *language B* but shifted as per its own language mean was similar to projecting onto the affine subspace for its own *language A*. This means, the affine subspaces for different languages are similar to one another when shifted according to language means.

Geometric Insights: One of the key insights from this research is the identification of patterns and shapes such as spirals, toruses, and curves within these subspaces, which represent different types of linguistic information as shown in Figure 4.6. For example, token position information forms a nearly perfect torus in the projected subspace, suggesting a geometric encoding of sequential information that is consistent irrespective of the language. These geometric insights by Chang et al. [2022] enable more targeted approaches to model training and fine-tuning, where language-specific and task-specific adjustments can be made based on the geometric properties of the model's representational space. This can potentially lead to improvements in model efficiency and effectiveness, particularly in tasks involving multiple languages.

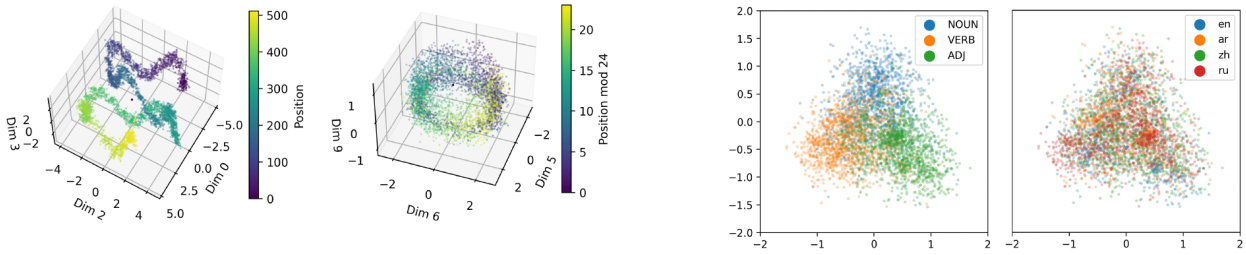

Figure 4.6: Geometric insights: (Left) shows the geometric representation of token positions in layer four of the XLM-R model, illustrating absolute positions with curves and relative positions with toruses. (Right) depicts the clustering of nouns, verbs, and adjectives in layer two, highlighting the model's ability to distinguish and generalize part-of-speech categories across different languages. (Chang et al. [2022])

Limitations: Despite its insights, the study acknowledges several limitations that could impact the generalizability of the findings:

- The linguistic diversity in the dataset is skewed towards Indo-European languages, which may not accurately represent the behavior of the model across globally diverse languages.
- Variations in corpus size and quality across different languages could lead to disparities in the learned representations, potentially biasing the model's performance towards languages with richer data resources.
- The study's reliance on a single model architecture (XLM-R) may limit the applicability of the findings to other multilingual models with different architectures.

Chapter 5

Cross-Lingual Transfer Mechanisms

5.1 Knowledge Editing in LLM

Core Idea: Cross-lingual knowledge editing in large language models (LLMs) by Wang et al. [2023] involves editing the model's knowledge in one language and evaluating its impact on another, addressing the unique challenge of maintaining consistent model behavior across linguistic boundaries. The core of cross-lingual knowledge editing lies in its ability to infuse new or updated knowledge into a model in a source language (e.g., English) and assess how these changes affect the model's performance in a target language (e.g., Chinese). This approach is particularly significant given the dynamic nature of knowledge and the continuous evolution of language and facts. For instance, updating a model's understanding from knowing three Honkai-series games to four, after the release of a new game, without full retraining, illustrates the practical utility of knowledge editing in LLMs as shown in Figure 5.1.

Monolingual Knowledge Editing: Monolingual knowledge editing focuses on updating a model's knowledge within the same language in which it was originally trained or is primarily operated. This approach is beneficial for quickly integrating new facts or correcting outdated or erroneous information. The typical process involves identifying the specific segments of the model that relate to the outdated or incorrect knowledge and applying targeted updates to these segments. The general approach can be mathematically represented as follows:

$$p'_\theta(x) = \begin{cases} ye & \text{if } x \in X_e \\ p_\theta(x) & \text{otherwise} \end{cases} \tag{5.1}$$

where:

- $p_\theta(x)$ is the original model's output,
- $p'_\theta(x)$ is the updated model's output,
- X_e is the set of inputs for which the knowledge should be updated,
- ye is the new correct output for inputs in X_e.

This equation ensures that only the specified aspects of the model are changed, preserving the performance on unrelated tasks or queries.

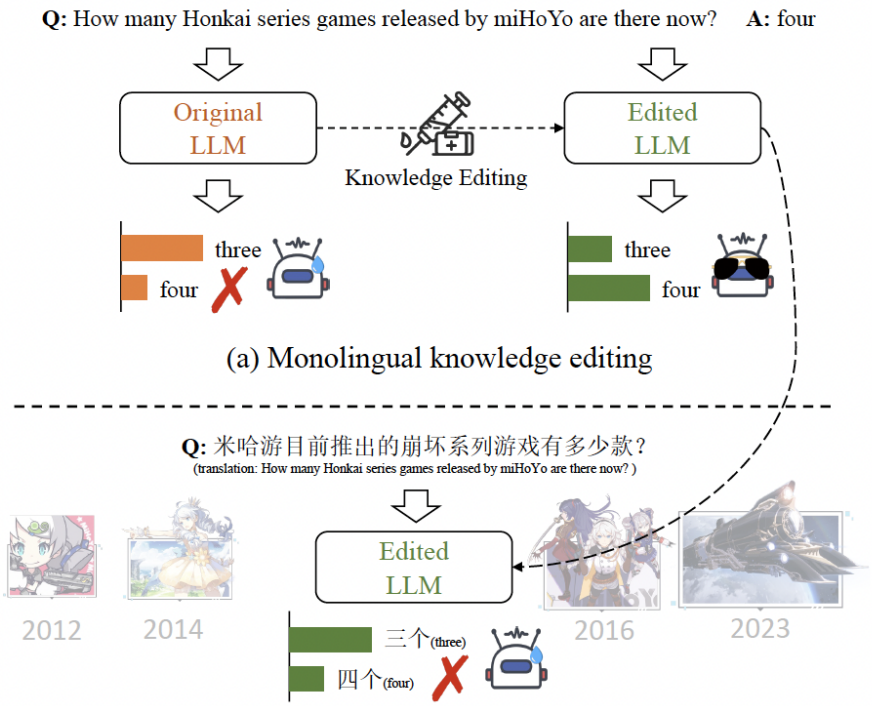

Figure 5.1: Knowledge editing: (a) This shows a language model being updated within the same language. It visually represents the model before and after knowledge editing. For example, before editing, the model might answer "three" to the question "How many Honkai series games released by miHoYo are there now?" After editing, to reflect current knowledge, it correctly answers "four," demonstrating the model's ability to update its information. (b) This depicts the process of editing knowledge in one language (e.g., English) and assessing the update in another language (e.g., Chinese). This shows how the model's response changes from "three" to "four" in Chinese after the model has been edited in English, emphasizing the cross-lingual transfer of updated knowledge. (Wang et al. [2023])

Cross-Lingual Knowledge Editing: Cross-lingual knowledge editing extends the concept of monolingual editing by enabling updates made in one language to be reflected in other languages supported by the model. This is particularly challenging due to differences in linguistic structure, semantics, and cultural nuances across languages. The model is first edited in a source language, and the effects of these edits are then assessed in a target language. The mathematical model for cross-lingual knowledge editing involves mapping the edits across languages:

$$p'_\theta(x_s) = \begin{cases} y_{se} & \text{if } x_s \in X_{se} \\ p_\theta(x_s) & \text{otherwise} \end{cases} \tag{5.2}$$

$$p'_\theta(x_t) = \begin{cases} I_t(y_{se}) & \text{if } x_t \in I_t(X_{se}) \\ p_\theta(x_t) & \text{otherwise} \end{cases} \tag{5.3}$$

where:

- x_s and x_t represent the input in the source and target languages, respectively,

- X_{se} is the edit scope in the source language,
- y_{se} is the new knowledge in the source language,
- I_t is a function translating source language edits into the target language context.

These processes ensure that the model not only learns the new information in the source language but also translates this learning effectively across other languages, thereby maintaining consistency and reliability in multilingual contexts.

Limitations: While knowledge editing provides a powerful method for updating and refining large language models (LLMs) post-training, it faces several limitations that can affect the efficacy and scope of its application.

- **Granularity of Edits:** Knowledge editing often requires precise identification of the information needing updates, which can be challenging. The granularity needed for effective edits might not always be achievable, especially in complex models where information is distributed across numerous parameters.
- **Dependency on Edit Quality:** The success of knowledge editing heavily relies on the accuracy and relevance of the edits themselves. Poorly executed or irrelevant edits can lead to model degradation rather than improvement.
- **Translation Inaccuracies:** In cross-lingual settings, translating edits from one language to another can introduce errors due to subtleties in language that are difficult to capture, affecting the model's performance in the target language.
- **Cultural and Contextual Differences:** Edits that are culturally or contextually specific to one language may not be directly applicable or appropriate in another, complicating the transfer of knowledge across languages.
- **Risk of Model Drift:** Frequent and extensive edits risk altering the foundational structures of the model, leading to what is known as model drift where the core capabilities of the model degrade over time.

5.2 Cross-Lingual Entity Alignment

Core Idea: Jiang et al. [2023] proposed a novel unsupervised method for cross-language entity alignment, which is useful in multilingual knowledge graphs for various applications such as information retrieval and question answering systems. The approach, named Unsupervised Deep Cross-Language Entity Alignment (UDCEA), aims to align semantic entities across different language knowledge graphs without relying on labeled data. The UDCEA approach utilizes a deep learning multi-language encoder combined with a machine translation tool to encode text from knowledge graphs, significantly reducing the dependence on labeled datasets. Unlike traditional methods, which focus on either global or local alignment strategies separately, UDCEA integrates both strategies effectively. It treats the alignment task as a bipartite matching problem and employs a re-exchanging technique to refine the alignment process. This method allows for generating ranked matching results, which are more flexible and useful for various downstream tasks. Figure 5.2 shows how specific entities (things or concepts) that exist in knowledge graphs of different languages can be aligned or matched to each other.

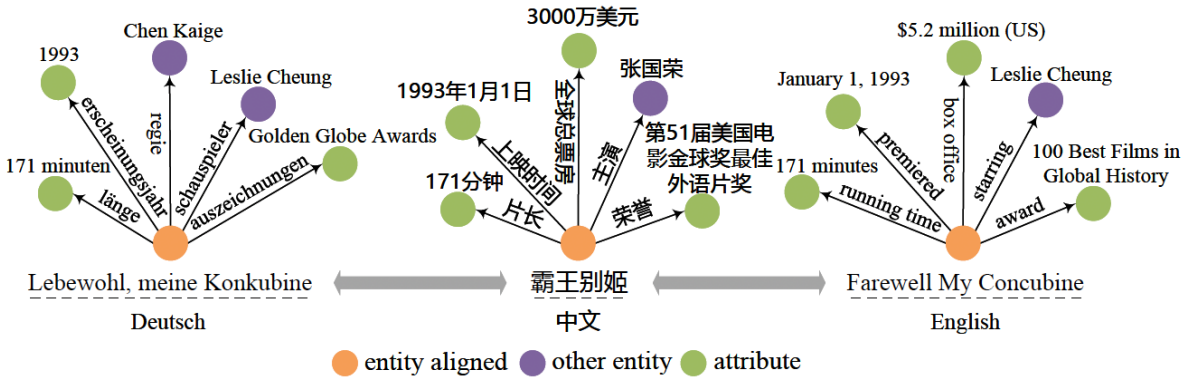

Figure 5.2: Cross-lingual entity alignment across multiple knowledge graphs: Orange nodes represent entities that are aligned across different knowledge graphs. An entity can be a thing, like a person, place, or any specific subject. Green nodes are attributes associated with the orange nodes. Attributes provide more details or characteristics about the entities. For example, attributes could be the date of birth of a person, the director of a movie, or the budget of a film. Lines (edges) between nodes show the relationship or connection between an entity and its attributes. This helps to understand how entities are related to their specific characteristics or to other entities. Other nodes (Gray and Yellow) represent other entities and attributes that are part of the knowledge graphs but are not the primary focus for the alignment shown in this figure. (Jiang et al. [2023])

Feature Embedding: The feature embedding module utilizes both machine translation and deep learning encoders to generate embeddings for entities. Let G_1 and G_2 be two KGs in different languages with entity sets E_1 and E_2 respectively. The embedding for an entity e in G_1 or G_2 is computed as follows:

$$\mathbf{v}_e = \text{Encoder}(\text{Translate}(e)) \tag{5.4}$$

Here, Translate$(\cdot)$ function translates non-English entity descriptions into English, and Encoder$(\cdot)$ is a pre-trained multilingual model that converts text into a high-dimensional vector space.

Alignment Module: Once entities are embedded into a vector space, the alignment module aims to find correspondences between entities in G_1 and G_2. This is formulated as an optimization problem where the distance between supposed matching entities is minimized. The similarity between entities $e_1 \in E_1$ and $e_2 \in E_2$ is given by:

$$\text{sim}(e_1, e_2) = \mathbf{v}_{e_1} \cdot \mathbf{v}_{e_2} \tag{5.5}$$

A binary variable $x_{e_1e_2}$ is defined which equals 1 if entities e_1 and e_2 are considered a match and 0 otherwise. The optimization can thus be expressed as:

$$\max_{x_{e_1e_2} \in \{0,1\}} \sum_{e_1 \in E_1, e_2 \in E_2} x_{e_1e_2} \cdot \text{sim}(e_1, e_2) \tag{5.6}$$

$$\text{subject to} \quad \sum_{e_2 \in E_2} x_{e_1e_2} \leq 1 \quad \forall e_1 \in E_1 \tag{5.7}$$

$$\sum_{e_1 \in E_1} x_{e_1 e_2} \leq 1 \quad \forall e_2 \in E_2 \tag{5.8}$$

These constraints ensure that each entity in G_1 is matched to at most one entity in G_2 and vice versa.

Limitations: Despite the advantages of our unsupervised deep cross-language entity alignment method, it is subject to several limitations:

- **Dependency on Translation Quality:** The performance of the method heavily depends on the quality of machine translation. Errors in translation can lead to incorrect embeddings and consequently, poor entity alignment.
- **Scalability Issues:** The method may not scale well to very large knowledge graphs due to increased computational demands, particularly in the optimization phase. This limitation poses challenges in handling datasets with a large number of entities.
- **Difficulty with Polysemous Entities:** Entities that have multiple meanings or are context-dependent can be a challenge. The method may align such entities incorrectly if their textual descriptions in the knowledge graphs do not distinguish between different contexts or meanings.

Chapter 6

Conclusion and Future Directions

6.1 Conclusions

- This study highlights the significance of language model fine-tuning techniques such as adapter-based fine-tuning, generalizable fine-tuning, and composable sparse fine-tuning. These methods offer efficient ways to adapt pre-trained language models to specific tasks and domains, contributing to improved performance.

- Moreover, the analysis of fact representation in language models highlight on the importance of task-specific model editing, cross-lingual fact representation, and the underlying geometry of language model representation. These insights helps understanding of how language models encode and manipulate factual knowledge, facilitating more effective utilization in various natural language processing applications.

- Furthermore, the exploration of cross-lingual transfer mechanisms, particularly knowledge editing in large language models and cross-lingual entity alignment shows the usage of multilingual models to bridge language barriers and facilitate knowledge transfer across different linguistic contexts.

6.2 Future Directions

- We can explore different versions of the Lottery Ticket algorithm to improve efficiency. Additionally, experimenting with other pruning methods like DiffPruning by Guo et al. [2021] and ChildTuning by Xu et al. [2021] could help refine our approach. Finally, we could adapt our method for tasks beyond cross-lingual transfer, such as multimodal learning and domain adaptation, to broaden its applicability.

- We can plan to improve how multilingual language models represent factual knowledge across languages. This includes developing better methods for cross-lingual fact representation learning and creating more accurate datasets for probing factual knowledge.

- We can aim to address the challenges highlighted in the study regarding the cross-lingual effect of knowledge editing. We can plan to investigate how the locality of language models in one language may impact editing outcomes in other languages, to enhance our understanding and improve cross-lingual knowledge editing techniques.

Bibliography

Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. 2023.

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

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

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. 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. 2019.

Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. 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 *Proceedings of the 36th International Conference on Machine Learning*, pages 2790–2799, 2019.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2023.

Chuanyu Jiang, Yiming Qian, Lijun Chen, Yang Gu, and Xia Xie. Unsupervised deep cross-language entity alignment. 2023.

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. 2019.

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)*, 2020.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

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

Jiaan Wang, Yunlong Liang, Zengkui Sun, Yuxuan Cao, and Jiarong Xu. Cross-lingual knowledge editing in large language models. 2023.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. 2021.

Xin Zhao, Naoki Yoshinaga, and Daisuke Oba. Tracing the roots of facts in multilingual language models: Independent, shared, and transferred knowledge. 2024.