

NNTI Project WS 2023/2024s

Nischal Maharjan
7058343

Jaykumar Bhagiya
7055903

Hevra Petekkaya
7055462

jabh00002@stud.uni-saarland.de

nima00018@stud.uni-saarland.de

hepe00001@stud.uni-saarland.de

1 Introduction

In this project, performance evaluation of pre-trained autoregressive language models specifically XGLM-564M and GPT2 was done. Followed by fine-tuning these models on Ayacucho Quechua languages based on several fine-tuning techniques, such as full fine-tuning, BitFit, LoRA, and IA3. Finally, we investigated how the model's performance improved in Ayacucho Quechua whereas, possibly degraded in other languages.

2 Running inference on a pre-trained LM

XGLM-564M is a multilingual autoregressive language model, which has 564 million training parameters, that is trained in 30 different languages composing 500 billion sub-tokens. Whereas, GPT2, which has 1.5 billion training parameters, was trained on 8 million web pages, which is approximately 40 billion tokens. Note that for XGLM "sub-tokens" are specified because it was trained on various languages where some of them could be agglutinative languages, hence the tokenization breaks down the words/tokens into simpler parts because they themselves carry information.

Given that XGLM was trained on more data, one can expect that it will outperform GPT2. However, one should keep in mind other factors such as the number of training parameters, quality of data, etc. GPT2 has more parameters that provide it with more learning capacity. Also, regarding data quality, in a podcast with Lex Fridman, Ilya Sutskever said that the web pages used for training were obtained from links in Reddit posts that received more than three upvotes. This can be seen as an indicator of quality data.

2.1 Loss evaluation over various languages

As can be seen in Figure 1, the loss of GPT2 and XGLM-564M has been evaluated over several languages. From a quick glance, it can be inferred that

GPT2 outperforms XGLM. It is worth noting that since XGLM was trained on a balanced corpus of languages, the loss obtained is also quite similar among different languages. On the other hand, for GPT2, the loss for languages with the Latin script is higher than the non-Latin script ones such as Arabic, Tamil, and Telugu. It can be argued that this is because these languages have their unique scripts, and it is easier for the language model to distinguish them. However, there are 4 languages with the Latin script, hence, it becomes challenging to distinguish them.

The loss for Ayacucho Quechua is higher than other languages both for XGLM and GPT2. That could be because Ayacucho Quechua has a high degree of morphological complexity [5] and hence the tokenization method useful for other languages could fail with Ayacucho Quechua. In task 3, we have fine-tuned the models on Ayacucho Quechua to see how the performance improves for Ayacucho, possibly degrading in other languages.

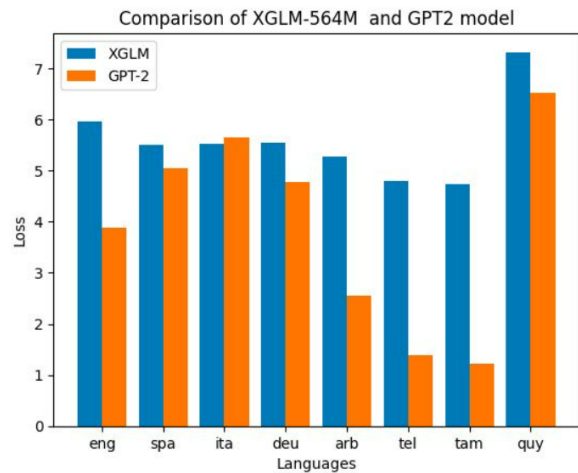


Figure 1: Pretrained GPT2 and XGLM-564M performances

2.2 Performance on Gujarati Language

We chose the Gujarati language to compare the performances of the two models from a multilingualism perspective. Note that Gujarati was not present in the training dataset of XGLM-564M. As can be seen in Figure 2, the loss attained on GPT2 was much lower than XGLM-564M when evaluated on Gujarati. One reason could be that the Gujarati language was not in the dataset that XGLM was trained on so the model did not get the chance to learn it. On the other hand, GPT2 was trained on the web pages irrespective of the language, hence, it is likely that it saw data in the Gujarati language and learned it. Also given that GPT2 has a huge number of training parameters, it could easily pick up the details of the Gujarati language even if it didn't see a lot of data on it.

Comparison of Language Modeling Loss: XGLM vs. GPT2

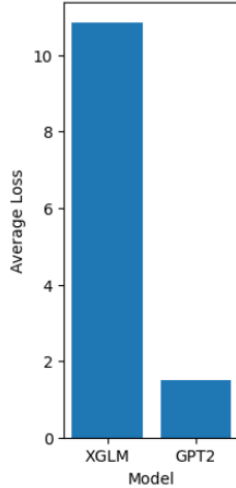


Figure 2: GPT2 vs XGLM-564M - Gujarati

3 Exploring multi-lingual representation spaces

Visualizing hidden representations is crucial for understanding the behavior and learning dynamics of neural networks. It helps in gaining insights into how the model processes and represents input data. Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) are two popular techniques used for visualizing high-dimensional data, such as hidden representations in neural networks and we have implemented both the methods to visualize token representation as well as sentence representation by different layers of the network.

PCA retains maximum variance in the data when

reducing the dimensions. On the other hand, t-SNE focuses on maintaining similar distribution of data in original space and lower dimension space i.e., distances between neighboring data points are preserved, potentially forming clusters that represent similarities between languages. This difference can be observed in the distinct clusters formed at different layers, with similar languages grouped together.

3.1 Token Representation

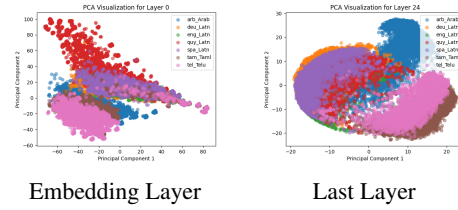


Figure 3: Token Representation Using PCA

Figure 3 shows the visualization of hidden representation of tokens for first embedding layer and the last layer. Interpreting the representation of last layer we can see that the languages that are from Latin script i.e., English, Spanish, Deutsch are clustered in similar space whereas Tamil, Telugu and Arabic which are follow different script are separated in the embedding space.

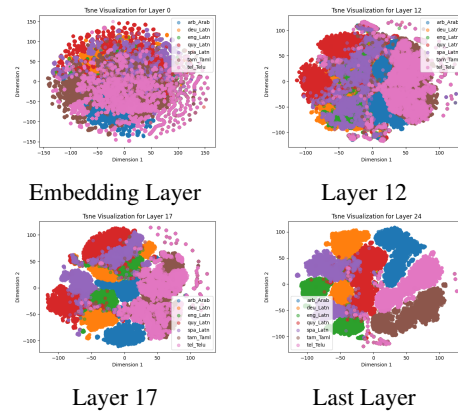


Figure 4: Token Representation Using TSNE

Similar to PCA, TSNE representation for Embedding layer, Last layer and two other intermediate layer is shown in figure 4. Unlike PCA, TSNE not only tries to measure the variance but also tries to maintain the distance thus resulting in proper clusters of for different languages. We see as we move on to later layers the formation of clusters are more distinct and prominent. This shows that as

we move on to the deeper layer, the model learns to distinguish between the different languages.

Looking closer we also see that the Telegu and Tamil languages are clustered nearby. Since both language are descended from ancient Dravidian dialect, they are quite similar thus the representation for them are similar as well which is evident from the visualizations.

3.2 Sentence Representation

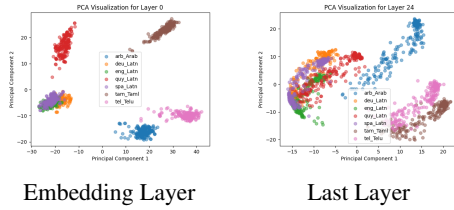


Figure 5: Sentence Representation Using PCA

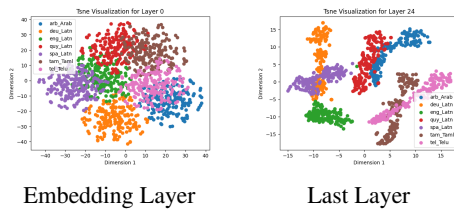


Figure 6: Sentence Representation Using TSNE

For sentence embedding, we have taken average of all tokens present in the given sentence. We see similar pattern for sentence representations like that of token representations as shown in figure 5 and 6

3.3 Interactive visualization using Altair

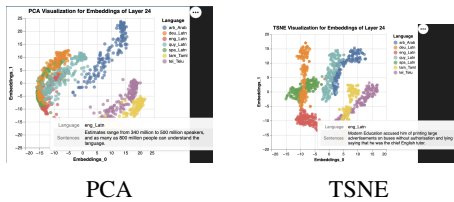


Figure 7: Interactive visualization using Altair

For more interactive visualization we also have utilized Vega-Altair [6], a declarative visualization library for Python. We can now also visualize the datapoints with their original sentence and tokens as shown in figure 7. However there is a limit of 5000 samples hence for token representation we required to sample the data points.

4 Language model adaptation

As can be seen in the model evaluation section, both GPT2 and XGLM-564M perform poorly in the Ayacucho Quechua. It was argued that this could be due to the high degree of morphological complexity that it has. Hence, fine-tuning the model, XGLM, in this language could give the model a chance to grasp the complexity present in this language. Four fine-tuning techniques were used, which are full fine-tuning, BitFit, LoRA, and IA3. The following subsections will examine them in more detail.

4.1 Full Fine-tuning

Full fine-tuning [3] is one of the simplest technique to apply because all the trainable parameters of the model are allowed to be updated. However, there are two main issues with this method. First, it can be very costly and sometimes infeasible for huge models such as GPT2, where there are 1.5 billion trainable parameters. Second, it could lead to catastrophic forgetting [4] which is a phenomenon that happens when the model forgets what it has previously learned due to adjusting all of its parameters based on the specific dataset that it was fine-tuned on.

The loss of the XGLM-564M on the Ayacucho Quechua language, which was used for fine-tuning, decreased from approximately 7.5 to approximately 4, which can be considered an improvement in performance. On the other hand, for languages that it was not fine-tuned on, there are no signs of catastrophic forgetting. The performance mostly stayed approximately the same except for Italian where the loss increased from approximately 5.5 to 7.

4.2 BitFit

BitFit is fine-tuning method [7], where the weight parameters are kept frozen, whereas, the biases are allowed to be updated. This significantly reduces the computational burden and makes it more feasible to fine-tune models with a large number of trainable parameters.

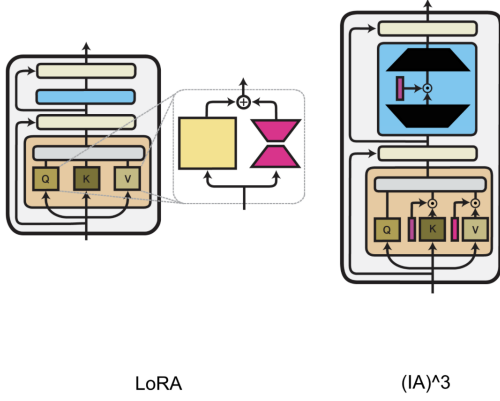


Figure 8: PEFT Methods (Lora and IA3)

4.3 LoRA (Layer-wise Relevance Adjustment)

LoRA[1] is another method for fine-tuning language models. It helps these models adapt to new tasks or languages without forgetting what they already know. LoRA does this by adjusting how much attention each part of the model pays to different information.

We utilize a low-rank decomposition to update a pre-trained weight matrix W_0 . This decomposition, represented as $h = W_0x + \Delta Wx = W_0x + BAx$, involves matrices B and A with trainable parameters. During training, W_0 remains fixed while A and B are updated. We initialize A randomly and B as zeros, ensuring $\Delta W = BA$ is zero initially. The scaling factor α adjusts ΔWx during training, where α is a constant scaled by the rank r of the decomposition.

4.4 IA3 (Infused Adapter by Inhibiting and Amplifying Inner Activations)

IA3 [2] is one of the parameter-efficient fine tuning method for LLM model which rescales inner activations with learned vectors. We inject three learned vectors l_k , l_v and l_{ff} to activations of key vector, value vector, and the feed-forward dense layer in the transformer layer as shown in figure 8. We carry out element-wise multiplication between these learned vectors and the activations. We then freeze all the pre-trained weights and only learn these parameters hence reducing the number of trainable parameters. Compared to Lora the number of trainable parameters is even lower in IA3. The architecture is modified as

$$\text{softmax} \left(\frac{Q(l_k \odot K^T)}{\sqrt{d_k}} \right) (l_v \odot V) \quad (1)$$

$$(l_{ff} \odot \gamma(W_1x))W_2 \quad (2)$$

5 Results

See the appendix for the results. Figure 9 and Table 1 shows the performance of different fine-tuning methods. We see all fine-tuning methods perform well in the Ayacucho Quechua language on which the model is fine-tuned.

6 Conclusion

In general, we investigated the performance of XGLM-564M and GPT2 model performance on different languages for autoregressive tasks. We see that GPT2 performs comparatively better than the XGLM-564M model. The performance of both models were low for **Ayacucho Quechua** language. We investigated the effect of finetuning on a dataset of this particular language and saw an improvement in the performance of the model. We implemented methods such as Bitfit, Lora, and IA3 for finetuning. The since model contains a large number of parameters we see the high time complexity for full fine-tuning. Using PEFT methods the time complexity of model training is reduced for similar amount of model performance. We see that the finetuning improves the performance of the model in the particular language but there is some level of degradation in the performance of the model in other languages.

6.1 References

References

- [1] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.
- [2] Haokun Liu, Derek Tam, Muqeeth Mohammed, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- [3] Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources, 2023.
- [4] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.
- [5] John E. Ortega, Richard Castro Mamani, and Kyunghyun Cho. Neural machine translation with a

polysynthetic low resource language. *Machine Translation*, 34(4):325–346, 12 2020.

- [6] Jacob VanderPlas, Brian Granger, Jeffrey Heer, Dominik Moritz, Kanit Wongsuphasawat, Arvind Satyanarayan, Eitan Lees, Ilia Timofeev, Ben Welsh, and Scott Sievert. Altair: Interactive statistical visualizations for python. *Journal of open source software*, 3(32):1057, 2018.
- [7] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models, 2022.

6.2 Appendices

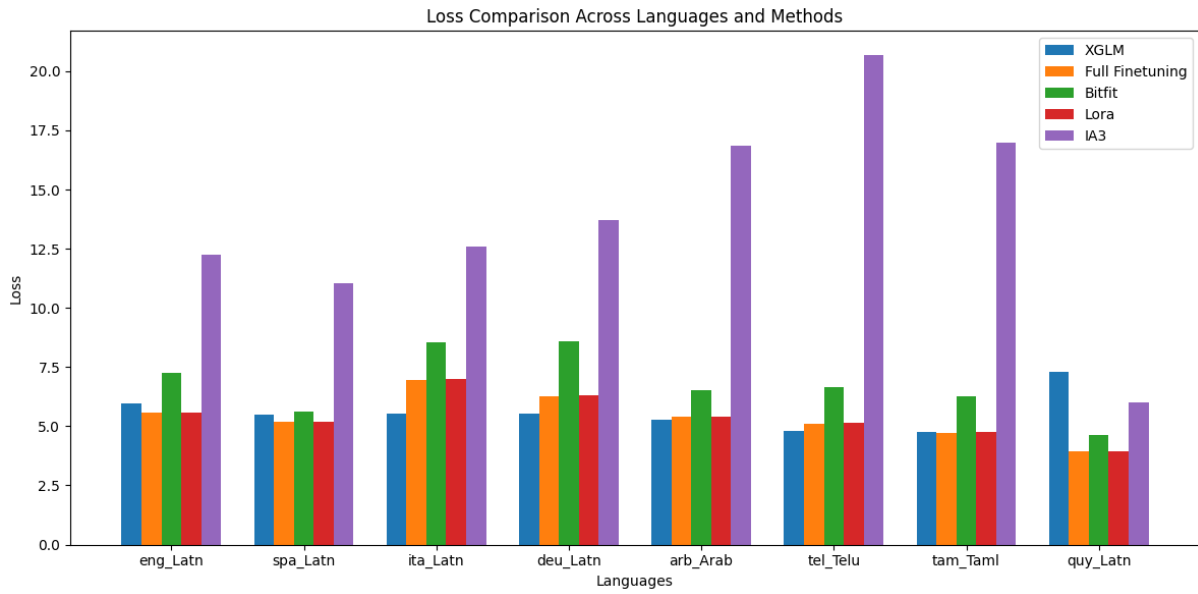


Figure 9: Comparison of different models

Table 1: Loss using different fine tuning methods

Language	XGLM	Full finetuned	Bitfit	Lora	IA3
engLatn	5.95	5.59	7.27	5.57	12.26
spaLatn	5.50	5.20	5.62	5.18	11.03
itaLatn	5.52	6.93	8.53	6.98	12.61
deuLatn	5.54	6.27	8.56	6.32	13.72
arbArab	5.28	5.38	6.50	5.41	16.83
telTelu	4.79	5.10	6.65	5.13	20.66
tamTaml	4.74	4.72	6.27	4.75	16.99
queQuec	7.31	3.96	4.63	3.95	6.02