Instructions for ACL 2023 Proceedings

Martínez, Camilo
Universität des Saarlandes
cama00005@uni-saarland.de

Ma, Honglu
Universität des Saarlandes
homa00001@uni-saarland.de

Abstract

This paper presents a comprehensive evaluation of multilingual representation spaces using pretrained models, conducted as part of the Neural Networks: Theory and Implementation at Universität des Saarlandes. We examine the hidden representations of language models across various languages. Our study focuses on a subset of six languages and employs dimensionality reduction techniques, PCA and t-SNE, to visualize and interpret high-dimensional spaces on the context of language embeddings. We acknowledge the limitations posed by computational resources, which restrict batch sizes and the number of languages analyzed. We fine-tuned the model xglm-564M on a Quechua dataset and compared the performance of the original model with the fine-tuned versions. We also implemented and compared the performance of different fine-tuning methods, including full fine-tuning, BitFit, LoRA, and $(IA)^3$.

1 Experiments and Results

1.1 Dimensionality Reduction Techniques

For PCA visualizations, we used the standard PCA implementation from the scikit-learn library. We used the default parameters for the PCA algorithm, recommended by the library's documentation. Unlike t-SNE, PCA does not need to be fine-tuned, as it is a deterministic algorithm.

On the other hand, for the t-SNE visualizations, we used the parameters recommended on various papers such as (Gove et al., 2022), (Weber, 2023), where they recommended to try out perplexities on the range of [1, 50], focusing on bigger numbers for big datasets. ALso, for the learning-rate of the algorithm, they recommend to take $\max(200, n/12)$ as an appropriate value, where n is the number of samples in the dataset.

1.2 Fine-tuning the XGLM-564M Model

Full-finetuning. We implemented a full finetuning approach, where the entire model's weights are relearned on the target task from scratch. A full fine tuning on a pretrained model is a common practice in NLP. It is a simple and effective way to adapt a model to a specific task. To do a full fine tune, we first need to supply the existing model new data to train from. As its name suggested, a full fine tuning can be very time consuming and computationally expensive As the model is traning every parameters again specifically for the new task, it is also more prone to overfitting.

We started Task 3 by finding a Quechua dataset on HuggingFace where we found a dataset with pure Quechua sentences. After the fine tuning, our model gained a significant drop in loss (5.5) with only 8192 data points. (The original dataset has 80k data points which is too big for our machine to handle)

Although the loss is lower than the previous pretrained model, it takes a long time to train with the reason specified above.

BitFit. We implemented Simple Parameter efficient Fine-tuning for Transformer-based Masked Language-models, BitFit (Zaken et al., 2022). BitFit is a simple method to fine tune a model. Since it only performs gradient decent on bias term, it is much faster than full fine tuning. However, it is also result in a less accurate model with a higher loss.

LoRA. We implemented a Low-Rank Adaptation of Large Language Models or LoRA, introduced in the paper (Hu et al., 2021). Instead of training the whole matrix which has a size of nxm, LoRA only trains two smaller matrices with a size of nxr and mxr where r is the smaller than the rank of the original matrix. This results in a smaller number of trainable parameters and thus, a faster training time. Our model's training result corresponds to the paper's result. Since LoRA approximates the weight matrix, it still has a higher loss than the full

fine tuning method but a better result than BitFit. The training time is similar but slightly more than BitFit which is also caused by number of trainable parameters as expected.

. We implemented a PEFT method called $(IA)^3$ that scales activations by learned vectors, attaining stronger performance while only introducing a relatively tiny amount of new parameters, as introduced in the paper (Liu et al., 2022). $(IA)^3$ is a method very similar to LoRA. Instead of training two smaller matrices to approximate the original one, it trains a vector to scale the activations of the model, i.e. a vector being multiplied to the output of activation function. Since an element-wise multiplication is also some sort of matrix transformation, it is an analog to LoRA. For each layer, IA3 trains three vectors which corresponds to scale the result of the key projection, the value projection and the activation function. Unfortuneately, due to a lack of computational resource and time, our IA3 model is not showing promising result. As the paper stated, we embeded three trainable vectors to each layer.

1.3 Plots

Some of the visualizations of the hidden representations of the sentences and tokens for the XGLM-564M model are shown in the Appendix.

The model performance plots were as follows (the rest of the plots are in: https://wandb.ai/hwga-cj/xglm-full)

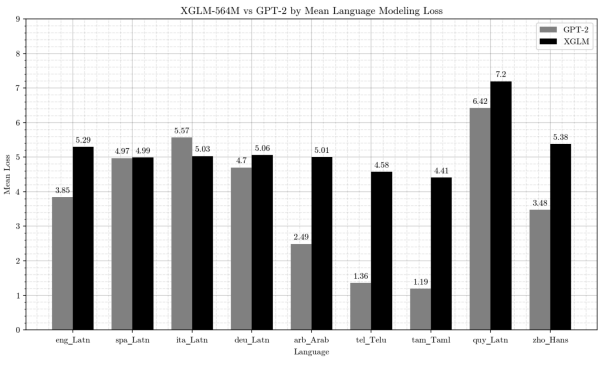

Figure 1: Loss of the XGLM-564M Model compared to GPT-2

2 Conclusions

Limitations

Due to the extensive computational requirements, the study was limited to a batch size of 2. This constraint could potentially affect the generalization of the findings. For instance, larger batch sizes may yield different embeddings and therefore, the representations of the hidden-layers of the model and potentially alter the provided PCA and t-SNE visualizations. Additionally, the analyses were focused on a selection of six languages: English, Spanish, German, Arabic, Tamil, and Quechua. Therefore, the results presented may not be universally applicable, particularly to languages with much different structures.

Furthermore, for the sensible feasibility of t-SNE computations on our datasets, we employed the fast implementation provided via the openTSNE library (Poličar et al., 2019), given our computational resources constraints. Although our study utilized the CS cluster, the requirement for substantial GPU resources limits the scalability of our methods, particularly for longer texts or larger datasets. This presents an avenue for further research to develop more resource-efficient algorithms capable of handling more extensive and linguistically diverse data without compromising on the fidelity of the representations.

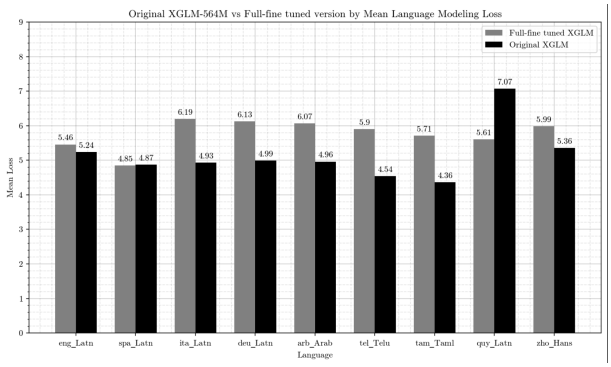

Figure 2: Loss of XGLM-564M Model compared to its fine-tuned versions

Ethics Statement

Our review work strictly adheres to the ACL Ethics Policy.[1] We acknowledge that while our evaluation contributes to the understanding of language model behaviors and their applications, it does not introduce novel methodologies or data. We have carefully considered the ethical implications of our work, ensuring that our analyses do not misrepresent capabilities, and we recognize the importance of responsible AI development and the potential impact of language technologies on society. Our

[1] https://www.aclweb.org/portal/content/acl-code-ethics

work aims to provide a comprehensive, unbiased evaluation to foster informed discussions within the research community. This review has been conducted with an emphasis on transparency and reproducibility, encouraging the ethical advancement of natural language processing research.

Acknowledgements

This work was conducted as a final project for the course "Neural Networks: Theory and Implementation," instructed by Prof. Dietrich Klakow at Universität des Saarlandes. We extend our gratitude to the course's Team for their guidance on this project and to the University. This document format and styling has been adapted by Jordan Boyd-Graber, Naoaki Okazaki, Anna Rogers from the style files used for earlier ACL, EMNLP and NAACL proceedings, including those for EACL 2023 by Isabelle Augenstein and Andreas Vlachos, EMNLP 2022 by Yue Zhang, Ryan Cotterell and Lea Frermann, ACL 2020 by Steven Bethard, Ryan Cotterell and Rui Yan, ACL 2019 by Douwe Kiela and Ivan Vulić, NAACL 2019 by Stephanie Lukin and Alla Roskovskaya, ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Mitchell and Stephanie Lukin, BibTEX suggestions for (NA)ACL 2017/2018 from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, ACL 2010 by Jing-Shin Chang and Philipp Koehn, ACL 2008 by Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, ACL 2005 by Hwee Tou Ng and Kemal Oflazer, ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

References

Robert Gove, Lucas Cadalzo, Nicholas Leiby, Jedediah M. Singer, and Alexander Zaitzeff. 2022. New guidance for using t-sne: Alternative defaults, hyperparameter selection automation, and comparative evaluation. *Visual Informatics*, 6(2):87–97.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning.

Pavlin G. Poličar, Martin Stražar, and Blaž Zupan. 2019. opentsne: a modular python library for t-sne dimensionality reduction and embedding. *bioRxiv*.

Philipp Weber. 2023. Uncertain choices in method comparisons: An illustration with t-sne and umap. Bachelor's thesis, Ludwig-Maximilians-Universität München, Munich, Germany, April. Department of Statistics: Technical Reports.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models.

A Appendix: Plots

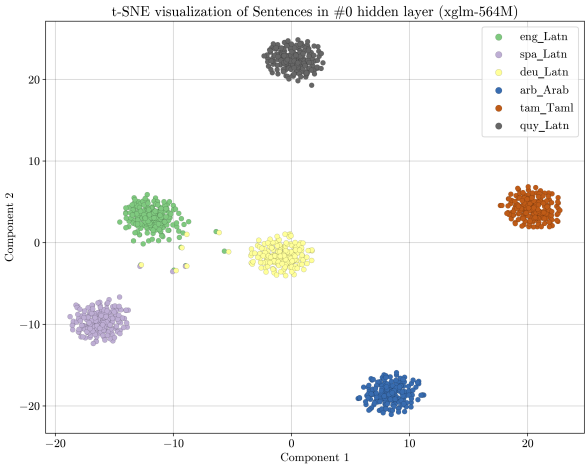

Figure 3: t-SNE Visualization of Hidden Representations of Sentences for Layer 0 of the XGLM-564M Model

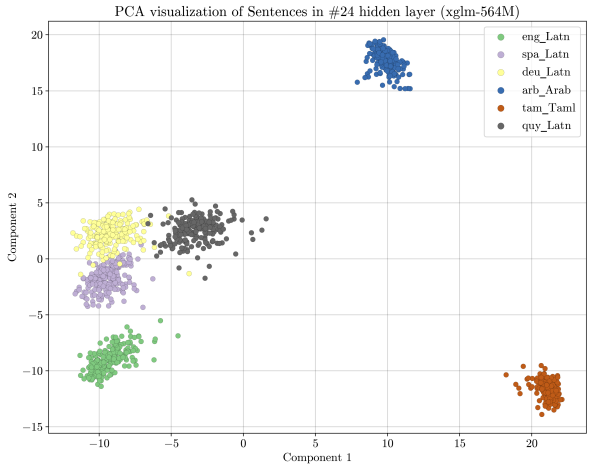

Figure 4: PCA Visualization of Hidden Representations of Sentences for Layer 24 of the XGLM-564M Model

Figure 5: t-SNE Visualization of Hidden Representations of Sentences for Layer 24 of the XGLM-564M Model

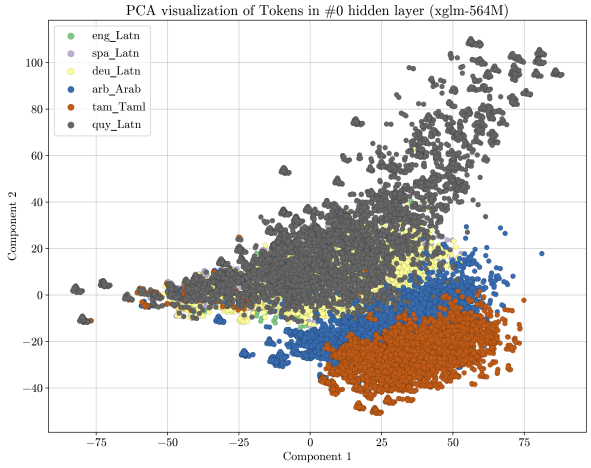

Figure 6: PCA Visualization of Hidden Representations of Tokens for Layer 0 of the XGLM-564M Model

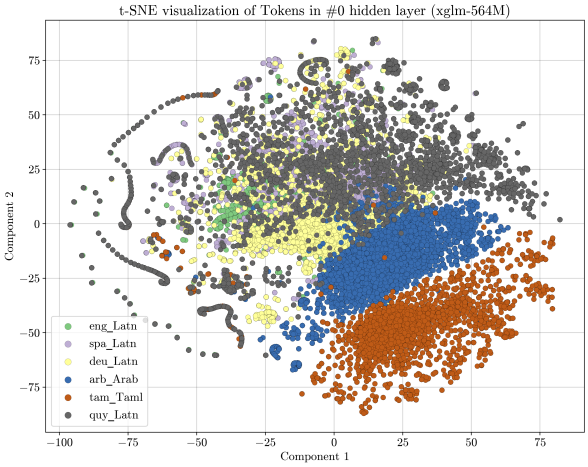

Figure 7: t-SNE Visualization of Hidden Representations of Tokens for Layer 0 of the XGLM-564M Model

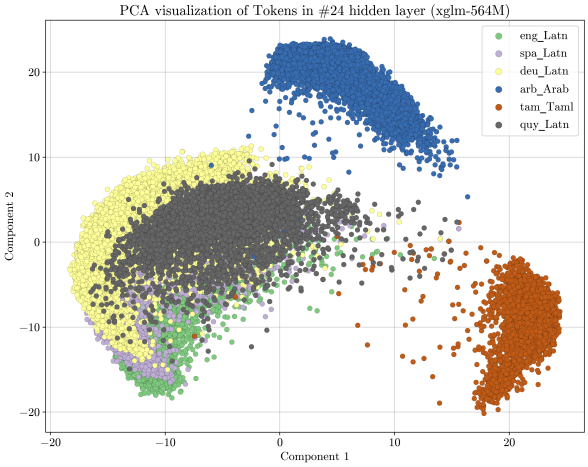

Figure 8: PCA Visualization of Hidden Representations of Tokens for Layer 24 of the XGLM-564M Model

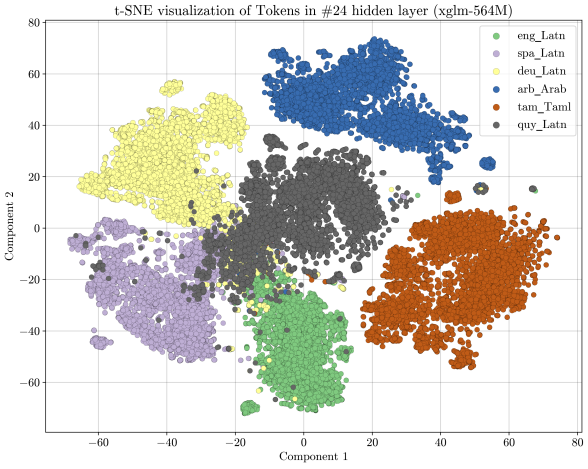

Figure 9: t-SNE Visualization of Hidden Representations of Tokens for Layer 24 of the XGLM-564M Model