Performance Evaluation of NLP Models for European Portuguese: Multi-GPU/Multi-Node Configurations and Optimization Techniques

Daniel Santos$^{[0000-0002-9906-0358]}$, Nuno Miquelina$^{[0000-0002-3202-2242]}$, Daniela Schmidt$^{[0000-0002-0076-1462]}$, and Paulo Quaresma$^{[0000-0002-5086-059X]}$ and Vítor Beires Nogueira$^{[0000-0002-0793-0003]}$

VISTA Lab, ALGORITMI Research Center, University of Évora, Portugal
`dfsantos@uevora.pt`, `d37384@alunos.uevora.pt`,
`{daniela.schmidt,pq,vbn}@uevora.pt`

Abstract. Natural Language Processing (NLP) research has predominantly focused on the English language, leading to a wealth of resources and advancements tailored to English. However, there is a growing need to extend these capabilities to other languages, such as European Portuguese, to ensure the inclusivity and accessibility of NLP technologies. In this study, we explore the evaluation of NLP models in the European Portuguese language using a multi-GPU/multi-node machine. We utilized various tools such as PyTorch, Accelerate, Transformers, and DeepSpeed with ZeRO Stage 3 to handle the computational demands of large-scale model training. We provide all the key aspects of our methodology to evaluate various models on translated GLUE tasks. Additionally, we introduce AiBERTa, a base model with 110 million parameters, developed and pre-trained on a corpus tailored for European Portuguese. This research highlights the effectiveness of advanced tools and distributed computing in scaling NLP model training, providing a foundation for future enhancements in European Portuguese language processing.

Keywords: NLP · Model Evaluation · Distributed Training.

1 Introduction

Natural Language Processing (NLP) has made significant strides over the past decade, primarily driven by advancements in machine learning techniques and the availability of large-scale datasets. However, the majority of NLP research has predominantly focused on the English language, leading to a wealth of resources and advancements tailored specifically for English. This focus has created a disparity in NLP capabilities across different languages, with Portuguese being notably underrepresented despite its status as one of the most widely spoken languages globally [3].

The need to develop robust NLP models for Portuguese is critical, to ensure the inclusivity and accessibility of NLP technologies. Recent efforts have begun to address this gap by translating key NLP benchmarks, such as the GLUE benchmark, into Portuguese. [9] The GLUE benchmark, introduced by Wang et al.[15], is a collection of diverse tasks designed to evaluate and compare the performance of NLP models on tasks such as sentiment analysis, textual entailment, and similarity scoring.

In this study, we explore the evaluation of NLP models in the European Portuguese language using a multi-GPU/multi-node machine. We employed various state-of-the-art tools to handle the computational demands of small to large-scale model training. These tools enabled us to efficiently manage and scale the training of multiple NLP models, leveraging the computational power of our supercomputer setup.

Furthermore, we introduce AiBERTa, our under-development base model with 110 million parameters, developed and pre-trained on a corpus tailored for European Portuguese.

This research highlights the effectiveness of advanced tools and distributed computing in scaling NLP model training. By providing detailed insights into our methodology and experimental setup, we aim to contribute to the growing body of work dedicated to enhancing NLP for European Portuguese and other underrepresented languages.

2 AiBERTa Model Development

AiBERTa [7] is a BERT-based language model tailored for European Portuguese. It aims to fill the gap in natural language processing resources specific to this language variant. The text source comes from Arquivo.pt [1], a project initiated by the Foundation for National Scientific Computing (FCCN), focuses on preserving Portugal's digital heritage by archiving web content related to the Portuguese cultural and scientific sphere.

2.1 Corpus creation

The process to retrieve content from Arquivo.pt, involved several steps to ensure the collected data is relevant, clean, and suitable for building the AiBERTa language model. Here is a detailed explanation of the process:

1. Accessing Arquivo.pt APIs: this repository offers several APIs designed to facilitate the retrieval and analysis of archived web content. The main APIs used include:
   – Search API: allows performing text searches within the archived web content.
   – URL Search API: retrieves archived versions of a specific web page by providing its URL.
   – CDX Server API: provides access to the CDX index files.

[1] https://arquivo.pt

2. Index Files - CDX Files: CDX files act as detailed indexes for archived web content, enabling efficient lookup based on specific criteria like URL, timestamp, and more. Each line in a CDX file represents a single archived document and contains metadata such as:
   - url of the captured content
   - timestamp of the capture
   - digest (hash) of the content
   - MIME type of the content
   - HTTP status code
   - offset and filename within the archive
3. Processing and Filtering CDX Files: to ensure only relevant content is processed:
   - Decompressing ZIP Files: each ZIP file contains multiple index files.
   - Filtering by MIME Type and HTTP Status: only content with specific MIME types (e.g., text/plain, text/html, application/pdf) and HTTP 200 status are considered.
4. Retrieving Content: once the relevant URLs are identified:
   - Parallel Processing: multiple processes and threads retrieve content in parallel.
   - Text Extraction: based on MIME type, different parsers are used:
     - text/HTML: uses the Trafilatura Python library.
     - binary Files (PDF, RTF, Word): Uses the Apache Tika Python library.
   - hash calculation: ensures uniqueness of the extracted text by comparing hashes.
   - database insertion: The cleaned text is stored in a database for further processing.
5. Text Processing: after retrieval:
   - sentence Extraction: The text is divided into unique sentences.
   - perplexity calculation: Perplexity values are calculated to ensure linguistic quality.
   - hash calculation for sentences: ensures uniqueness at the sentence level.

This structured process ensures that the corpus built from Arquivo.pt is diverse, with high-quality, and suitable for training the AiBERTa language model.

2.2 Pre-training

Pre-training is the initial phase of training a language model on a large corpus of text data before it is fine-tuned for specific downstream tasks. The objective is to learn general language representations that capture syntax, semantics, and various nuances of the language. With the corpus collected, using content from Portuguese sites provided by Arquivo.pt repository, is expected to capture the essence of the European-Portuguese language.

A tokenizer was built using the collected corpus. This tokenizer was created using WordPiece technique and has a size of 20K tokens. This helps in handling out-of-vocabulary words and reduces the size of the vocabulary.

The Masked Language Modeling (MLM) objective was performed during the pre-training. This objective randomly masks some tokens in the input and trains the model to predict the masked tokens based on the context. For example, the sentence “Está um bom jogo. O jogador deveria de ter marcado aquele [MASK]” / "It’s a good game. The player should have scored that [MASK]", is suggested the token “golo” / "goal" to replace the [MASK] hidden word.

The pre-traning resorted to the BERT base model (uncased) with the following parameters 10 epochs, learning rate of $2e-5$ and weight decay of 0.01.

2.3 Base Model Overview

The Bert base model (uncased) is one of the most widely used configurations of the BERT (Bidirectional Encoder Representations from Transformers) model. AiBERTa was pre-trained from the start and only sharing the same configuration as this model, avoiding any bias from the previous trainings. It is a powerful and versatile transformer-based language model. Having approximately 110 million of parameters, the model has the following technical specifications:

- layers: 12
- hidden size: 768
- attentions heads: 12
- maximum input length: 512 tokens

3 Tools and Infrastructure

In this section, we provide an overview of the tools and infrastructure that were essential for our experiments in evaluating NLP models in European Portuguese. We utilized several key libraries and frameworks to handle the complexities of training and fine-tuning large-scale language models.

3.1 PyTorch, Transformers and Accelerate

PyTorch [2] is an open-source deep learning framework widely adopted for its flexibility and ease of use. Developed by Facebook’s AI research group, PyTorch provides a Python package with high-level features such as tensor computation with robust GPU acceleration [10].

Transformers [16], developed by Hugging Face[3], is a library that provides a general-purpose framework for developing and deploying NLP models. It supports a wide array of models such as BERT and GPT, facilitating easy access to pre-trained models and efficient fine-tuning on specific tasks. The Transformers library integrates seamlessly with PyTorch.

[2] https://pytorch.org
[3] https://huggingface.co

Accelerate [4] is a library designed to streamline the deployment of deep learning models on various hardware configurations [4]. It provides a high-level interface to manage multi-GPU and multi-node setups, significantly reducing the complexity involved in scaling up machine learning procedures. Accelerate was created for PyTorch and enables researchers and practitioners to focus more on model development and experimentation rather than on the specifics of distributed computing.

Accelerate optimises tensor operations across devices, utilizing techniques such as gradient accumulation and mixed precision training. These optimisations are critical in HPC environments to maximise computing efficiency and resource utilisation. We configured Accelerate to manage our distributed training tasks by specifying the appropriate device allocations, gradient accumulation, and precision levels, ensuring that each training process was optimized for the available hardware.

In our experiments, we utilized PyTorch and Transformers library to load pre-trained models and fine-tune them for specific NLP tasks, enhancing the performance and speed of our experimentation process. The Accelerate library was employed to handle the distribution of computations across multiple GPUs and nodes, ensuring efficient use of available resources and minimizing training time.

3.2 DeepSpeed and ZeRO

DeepSpeed [5] is a deep learning optimization library developed by Microsoft, to enable the efficient training of large-scale models. It offers a wide range of features, including mixed precision training, gradient accumulation, and advanced memory optimization techniques. One of its most notable contributions is the ZeRO [11] (Zero Redundancy Optimizer) technology, which significantly reduces memory and computational load during training.

ZeRO is structured into three stages, each providing progressively greater memory efficiency:

- Stage 1: The optimizer states are partitioned across the processes.
- Stage 2: Adds gradient partitioning.
- Stage 3: Incorporates parameter partitioning, allowing the training of large-scale models by distributing memory and compute load across multiple GPUs.

In our experiment we focused on ZeRO Stage 3, which enables the training of large models by partitioning all model states (optimizer states, gradients, and parameters) across the available devices. This allows for a substantial reduction in memory consumption, enabling the training of models that would otherwise be too large to fit into GPU memory.

[4] https://github.com/huggingface/accelerate
[5] https://github.com/microsoft/deepspeed

DeepSpeed with ZeRO Stage 3 was crucial for handling the largest models in our experiments while maintaining high training speeds. By leveraging these tools, we could efficiently manage memory and computation, enabling us to fine-tune models with millions of parameters across a multi-GPU/multi-node environment.

4 Experimental Setup

In this section, we outline the key components and configurations of our experimental setup. This includes the hardware specifications, the tools' setup, and the preparation of the dataset. These elements are crucial for replicating our experiments and understanding the context in which our results were obtained.

4.1 Hardware Configuration

Our experiments were conducted on a supercomputer, which comprises two compute nodes. The detailed specifications are as follows:

- CPU: Each node features Dual AMD Rome 7742, with 128 cores per node.
- System Memory: 1TB of RAM per node.
- GPUs: 8 NVIDIA A100 Tensor Core GPUs per node, each with 40GB of GPU memory, combine to total 320GB of GPU memory across the node.

Slurm [17] was used to control the scheduling and distribution of computational tasks on the supercomputer. Slurm is a scalable cluster management and job scheduling system. Slurm enables the allocation of resources and balances the computational load among multiple users of the supercomputer. In particular, Slurm's advanced scheduling capabilities allowed us to prioritize jobs, manage queues, and ensure fault tolerance, all of which are critical in maintaining the efficiency of large-scale HPC environments.

However, this also introduced an additional layer of complexity. Implementing and integrating all the necessary libraries and tools, such as PyTorch, Accelerate, Transformers, and DeepSpeed with ZeRO Stage 3, required ensuring compatibility within the Slurm-managed environment.

4.2 Tools Setup

An important component of our setup was the script provided by Hugging Face for text classification task examples [6], which served as the foundation for our workflow. Specifically, we used the `run_glue_no_trainer.py` script from the Hugging Face Transformers repository. This script, similar to `run_glue.py`, allows fine-tuning of any model available on the Hugging Face Hub for a text classification task, whether it is a GLUE task or custom data in a CSV or JSON

[6] https://github.com/huggingface/transformers/tree/main/examples/pytorch/text-classification

file. The main difference is that this script exposes the bare training loop, enabling quick experimentation and the addition of any necessary customizations. This flexibility was crucial for adapting the script to meet our specific needs, including handling different model architectures and training configurations.

To run this script efficiently in our multi-GPU setup managed by Slurm, we configured the accelerate library. We found the examples from accelerate library particularly useful as a baseline for making accelerate work with Slurm:

- `submit_multigpu.sh` [7]
- `submit_multinode.sh` [8]

These examples provided a starting point to use both the Slurm job scripts and the configuration of accelerate using the accelerate launch command. With these resources, we were able to effectively set up and manage our distributed training environment, ensuring optimal utilization of our hardware resources. The scripts and configurations detailed in these examples served as a robust starting point, allowing us to modify and extend them according to the specific requirements of our experiments.

In summary, the combination of the accelerate library, and the Slurm job management system enabled us to perform efficient and scalable NLP model training and evaluation on a multi-GPU and multi-node setup.

4.3 Dataset Preparation

In the domain of Natural Language Processing, rigorous evaluation is crucial for assessing the capabilities and generalization of models. One benchmarking tool widely recognized in the NLP community is the General Language Understanding Evaluation (GLUE). This dataset is designed to provide a comprehensive suite of tests that measure a model's performance across multiple linguistic tasks.

GLUE Benchmark The GLUE benchmark, introduced by Wang et al. [15], is a collection of nine diverse tasks designed to evaluate and compare the performance of NLP models on tasks such as sentiment analysis, textual entailment, and similarity scoring. These tasks include:

- CoLA - The Corpus of Linguistic Acceptability, measuring grammatical correctness.
- SST-2 - The Stanford Sentiment Treebank, for sentiment analysis.
- MRPC - The Microsoft Research Paraphrase Corpus, for paraphrase detection.

[7] `https://github.com/huggingface/accelerate/blob/main/examples/slurm/submit_multigpu.sh`

[8] `https://github.com/huggingface/accelerate/blob/main/examples/slurm/submit_multinode.sh`

- STS-B - The Semantic Textual Similarity Benchmark, assessing the similarity between sentences.
- QQP - Quora Question Pairs, evaluating whether a pair of questions are semantically equivalent.
- MNLI - The Multi-Genre Natural Language Inference, a task of textual entailment across multiple sources.
- QNLI - Question Natural Language Inference, adapted from the Stanford Question Answering Dataset.
- RTE - Recognizing Textual Entailment, focused on entailment tasks.
- WNLI - Winograd NLI, a test of coreference resolution based on the Winograd schema.

These tasks collectively challenge the model's understanding of language, testing both breadth and depth of linguistic features.

Detailed Overview of Selected GLUE Tasks

Microsoft Research Paraphrase Corpus (MRPC) The MRPC task is aimed at identifying whether two sentences are paraphrases of each other, essentially evaluating a model's ability to detect semantic equivalence. The corpus consists of sentence pairs automatically extracted from online news sources, with human annotations indicating whether each pair captures the same informational content. Evaluating on MRPC requires models to understand nuances in word choice and syntax that may change the surface form of a sentence while retaining or altering its meaning.

Recognizing Textual Entailment (RTE) The RTE task involves determining whether a given hypothesis can logically be inferred from a premise sentence. This task tests a model's ability to handle logical reasoning and understand context within a text. Models are judged on their ability to accurately assess entailment or contradiction in diverse contexts.

Semantic Textual Similarity Benchmark (STS-B) The STS-B task requires models to assign a similarity score to pairs of sentences on a continuous scale from 1 (not similar at all) to 5 (semantically equivalent). This task assesses a model's ability to interpret and compare the meanings of sentences. Performance on STS-B reflects a model's nuanced understanding of semantic relationships.

Winograd Schema Challenge (WNLI) The WNLI task is designed to test a model's ability to handle coreference resolution within the framework of the Winograd Schema Challenge. This task presents sentences containing pronouns whose antecedents are ambiguous and requires the model to determine the correct reference. It is a test of language understanding that probes a model's common-sense reasoning and context processing capabilities.

The training of the models for these tasks was made through their online submission system which enabled us to upload the models predictions for evaluation on the secret test set labels. By using this platform, we obtain the standardized metrics for performance comparison and ensured that our evaluations were consistent with other research.

5 Overview of Evaluated Models

To assess the performance of Natural Language Processing (NLP) models tailored for European Portuguese, we employed the GLUE benchmark, adapted to Portuguese through the ExtraGLUE dataset [9]. This allowed for a consistent and robust comparison across a diverse set of models pre-trained for European Portuguese, each designed with unique capabilities and scale. The models evaluated include:

– GlórIA [6]: A large generative language model specifically focused on European Portuguese. GlórIA is built on the GPTNeo architecture, featuring 1.3 billion parameters, 24 layers, a hidden size of 2048, and functions as a decoder-only language model (LLM).
– Albertina PT[12]: This model family, an adaptation of the BERT architecture influenced by the DeBERTa model, consists of three variants:
  • Base model with 100 million parameters
  • Mid-size model with 900 million parameters
  • Large model with 1.5 billion parameters
– Gervásio PT [13]: Developed from the LLaMA-2 7B model, this decoder-only model of the LLaMA family boasts 7 billion parameters. It includes 32 hidden layers, 32 attention heads, and utilizes the Byte-Pair Encoding (BPE) algorithm via SentencePiece for its tokenizer, supporting a vocabulary size of $32,000$.
– BLOOM [1]: As an autoregressive LLM, BLOOM generates coherent text in 46 languages and 13 programming languages. 1.1 billion parameter model was used.
– BLOOMZ mt 7.1B [8]: The BLOOMZ MT (Multitask) are multilingual models which are particularly suited for non-English prompting.
– Multilingual GPT model [14]: Utilizing GPT-2 sources along with sparse attention mechanisms provided by Deepspeed and Megatron, this model architecture reproduces GPT-3's functionality. It covers multiple languages, enhancing capabilities for low-resource languages.
– Carvalho 1.3B [2]: A 1.3 billion parameter model, which was trained using a GPT architecture and specifically fine-tuned on a Galician-Portuguese corpus to enhance their regional linguistic adaptability. The lexical and syntactic similarity between these two language variations is very high.
– AiBERTa Base: Our base model with 110 million parameters trained for European-Portuguese.

6 Experiments with Distributed Configurations

In this section, we present our experiments conducted with various distributed configurations to evaluate the impact on training performance and efficiency. The configurations tested include different numbers of GPUs and nodes to determine the optimal setup. In our experiment, we fine-tuned the Albertina-PTPT 900m on the MRPC task and kept the total batch size constant at 64 across all configurations by adjusting the batch size per device.

Table 1. Time and memory comparison across multiple single and multi-gpu configurations

	1 GPU	**2 GPUs**	**4 GPUs**	**8 GPUs**
Multi-GPU				
Time	04:24	04:22	04:21	04:32
Peak Mem. Alloc. (Per GPU, MB)	28 285	16 795	9 999	6 582
Total Mem. Alloc. (All GPUs, MB)	28 285	33 589	39 995	52 658

Table 2. Time and memory comparison across various multi-node configurations

	1 GPU per node	**2 GPUs per node**	**4 GPUs per node**
Time	04:56	06:31	07:00
Peak Mem. Alloc. (Per GPU, MB)	16 794	9 999	6 582
Total Mem. Alloc. (All GPUs, MB)	33 589	39 996	52 655

According to the results shown in Table 1, increasing the number of GPUs did not reduce the training time. This could be attributed to potential inefficiencies introduced by small batch sizes per device, as well as the same number of training steps across experiments, as the total batch size remained constant. This means that while more GPUs process data in parallel, the overall training duration remains similar due to the fixed number of steps required to complete an epoch. Additionally, communication overhead can also affect training time.

The peak memory allocated per GPU decreased as the number of GPUs increased, from 28,285 MB for 1 GPU to 6,582 MB for 8 GPUs. This trend indicates effective memory distribution across GPUs, which is crucial for training larger models which may not fit on a single GPU. With more GPUs, the memory load is shared, reducing the peak memory demand on each individual GPU.

In multi-node configuration (see Table 2), the overhead of inter-node communication and synchronization can contribute to longer training times when compared to single node setup.

In conclusion, while using multiple GPUs and nodes can facilitate the training of larger models by distributing memory load, the training time benefits are limited when the total batch size remains constant.

Table 3. Time, memory and performance comparison across various multi-gpu configurations with variable total batch size

	1 GPU	**2 GPUs**	**4 GPUs**	**8 GPUs**
Total Batch Size	64	128	256	512
Time	04:24	02:15	01:14	00:39
Peak Mem. Alloc. (Per GPU, MB)	28 285	23 632	20 250	18 557
Total Mem. Alloc. (All GPUs, MB)	28 285	47 263	81 001	148 457
Accuracy	0.86	0.85	0.81	0.70
F1	0.90	0.89	0.85	0.81

In our second experiment, we fine-tuned the Albertina-PTPT 900m model on the MRPC task, keeping the batch size per device constant, which means the total batch size will vary across different multi-GPU configurations. The results provide valuable insights into the trade-offs between training efficiency, memory usage, and model performance as shown in Table 3.

As expected, the training time decreased significantly with the increase in the number of GPUs. Peak Memory Allocation per GPU has also decreased slightly. This trend shows that while each GPU handles a constant batch size, the distribution of the total memory load becomes more balanced across multiple devices. Interestingly, the model performance, as measured by accuracy and F1 score, showed a decreasing trend with increasing total batch size. This decline in performance could be attributed to the impact of larger batch sizes on model convergence. Larger batch sizes often lead to noisier gradient estimates, which can hinder the model's ability to converge to an optimal solution [5]. In conclusion, this experiment emphasizes the significance of balancing batch size, training time, and model performance. While multi-GPU configurations can significantly decrease training time, careful attention is required to maintain model performance.

Table 4. Time, memory, and performance comparison across different mixed precision configurations

	FP32	**FP16**	**BF16**
Time	08:06	04:22	04:20
Peak Mem. Alloc. (Per GPU, MB)	28 124	16 795	16 794
Total Mem. Alloc. (All GPUs, MB)	56 249	33 589	33 588
Accuracy	0.87	0.86	0.87
F1	0.90	0.89	0.90

In our third experiment, we compared the performance, memory usage, and training time of the Albertina-PTPT 900m model on the MRPC task using different precision formats: FP32, FP16, and BF16. Table 4 shows us that the training time and memory bandwidth is significantly reduced when using FP16 and BF16 while model performance remains consistent across all mixed precision formats. This analysis indicates that using mixed precision formats, specifically FP16 and BF16, can substantially improve training efficiency by reducing training time and memory usage without major drawbacks. Additionally, the reduced memory usage can allow for larger batch sizes or more complex models to be trained on the same hardware.

In our last experiment, we explored the impact of offloading the optimizer and parameters to the CPU on training time and memory usage. We used the Albertina-PTPT 900m model on the MRPC task with a single GPU, comparing the scenarios of no offloading versus CPU offloading.

Table 5. Time and memory comparison with and without optimizer/parameter offloading to CPU

	No Offload	**CPU Offload**
Time	04:24	14:53
Peak Mem. Alloc. (MB)	28 285	16 437

Table 5 shows us that the training time increased significantly when offloading to the CPU. This substantial increase can be attributed to the introduced latency as parameters and optimizer states need to be continuously moved between the CPU and GPU during training, slowing down the process.

However, the peak memory allocated has decreased considerably. This reduction in memory usage demonstrates the effectiveness of offloading in freeing up GPU memory, making it possible to train larger models or use larger batch sizes within the same GPU memory constraints.

In conclusion, offloading optimizer and parameter states to the CPU can be a useful strategy for managing GPU memory usage, particularly in scenarios where GPU memory is a constraint. However, the significant increase in training time necessitates careful consideration. In cases where training speed is a priority, avoiding offloading may be preferable.

Figure 1 provides a comparative analysis of training time and memory usage across various configurations, including different precision formats (FP32, FP16, BF16), the use of 2 GPUs with double the batch size, and CPU offloading. As illustrated, using mixed precision formats like FP16 and BF16 significantly reduces training time and memory usage compared to FP32. Additionally, while the 2 GPU configuration with a batch size of 128 further improves training time, it comes at the cost of increased memory consumption. CPU offloading, while effective in reducing GPU memory usage, leads to a substantial increase in training time due to the overhead associated with data transfer between the GPU and CPU.

7 Evaluation and Results

In this section, we present the evaluation metrics and results of our experiments conducted on various NLP models.

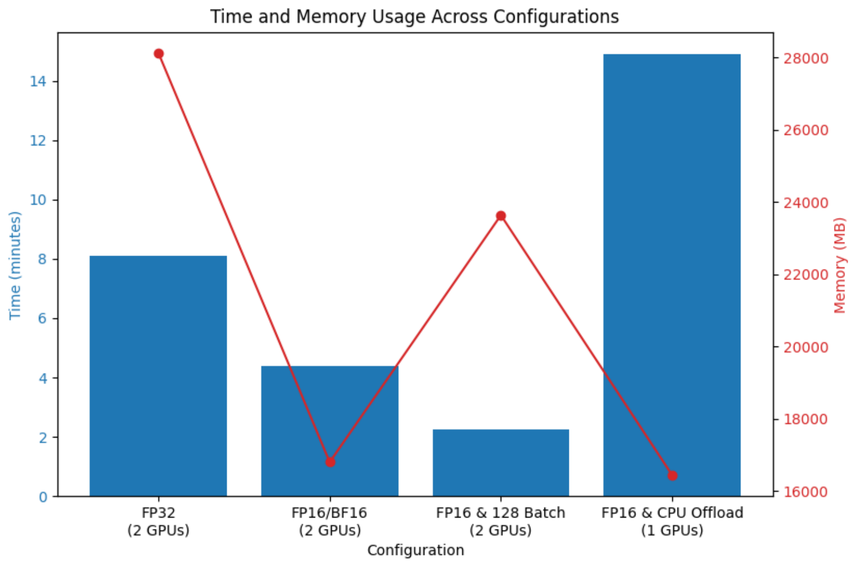

Fig. 1. Time and Memory Usage Across Configurations

The models were benchmarked using a variant of the GLUE dataset called ExtraGLUE, which is adapted to Portuguese through automatic translation of selected tasks from the GLUE and SuperGLUE benchmarks. This translation ensures the models are evaluated in a language-specific context, thereby providing more accurate and relevant results for Portuguese.

To optimize the training of our NLP models, particularly given the high computational overheads, we utilized a batch size of 16 per device and 2 gradient accumulation steps. For most models 2 GPUs were used, while Gervásio PT, due to its extensive parameter count (7 billion), we adjusted the batch size down to 4 per device to accommodate the model's demands within the available GPU memory and used 8 GPUs. Likewise, for Bloomz with 7.1 billion parameters, 8 GPUs were utilized with CPU offloading. Training epochs were set to five across all experiments.

7.1 Hyperparameter Tuning

The models underwent fine-tuning over a range of hyperparameters:

- Learning rates of $1e-4$ and $1e-5$
- Utilization of either a linear or constant scheduler to manage learning rate adjustments
- Fixed seeds for training reproducibility (41, 42, 43)

These hyperparameters were selected to optimize performance while ensuring that the models could adequately learn from the translated tasks.

7.2 Results

Table 6. Evaluation results on the Portuguese ExtraGLUE tasks.

Models	**RTE**	**MRPC**		**STS-B**		**WNLI**
	Acc	F1	Acc	Pearson	Spearman	Acc
Encoders						
AiBERTa Base	55.3	83.2	75.9	80.2	79.4	58.9
Albertina-PTPT 100m	55.4	87.6	83.1	84.5	84.1	65.1
Albertina-PTPT 900m	80.6	89.8	86.6	88.7	88.3	65.1
Albertina-PTPT 1.5B	82.9	90.3	87.2	88.7	88.4	59.6
Decoders						
Carvalho_pt-gl 1.3B	68.0	86.0	79.9	82.6	81.9	65.1
Gloria 1.3B	63.8	85.2	79.1	82.0	81.2	65.1
Gervásio 7B	83.2	90.5	87.0	87.9	87.6	64.4
mGPT 1.3B	58.9	85.5	79.3	78.3	76.9	65.1
Bloom 1.1B	71.5	87.7	82.7	85.1	84.3	63.7
Bloomz mt 7.1B	81.4	89.1	85.4	86.4	85.5	65.1

The evaluation results are shown in Table 6, where both encoders and decoders were evaluated in the four tasks previously described.

Decoder models, such as those designed for generative tasks, typically excel in generating text based on the input they receive. However, their architecture may not be inherently optimized for classification tasks like those found in the GLUE benchmark, where the goal is often to predict a label or a class based on input text. These tasks require precise understanding and categorization rather than open-ended text generation. Instead of adapting the decoder models to output a single classification label, another approach could be instruction tuning. This involves training the model on a range of NLP tasks given as instructions in natural language. For example, prompting a decoder model with instructions like "Classify the sentiment of the following text:" could help it apply its generative capabilities within a classification framework.

Additionally, given the limited size of the WNLI dataset, leveraging knowledge transfer from larger datasets through multi-task learning or transfer learning could significantly improve model robustness and generalization. Training models on other tasks and then fine-tuning on WNLI may provide the necessary depth of understanding navigating its complex coreference resolutions. Otherwise, our results suggest that the models are unable to outperform a baseline model.

8 Conclusion

In this study, we have explored the evaluation of NLP models for European Portuguese using a multi-GPU/multi-node machine, focusing on the impact of various distributed training configurations, mixed precision formats, and optimizer/parameter offloading strategies. Our experiments were conducted using state-of-the-art tools, including PyTorch, Accelerate, Transformers, and DeepSpeed with ZeRO Stage 3, to handle the computational demands of large-scale model training.

In our experiments with different multi-GPU/multi-Node configurations, we found that while distributing memory load across more GPUs can support the training of larger models, it is crucial to balance batch size, training time, and model performance to achieve optimal results. We also explored the impact of mixed precision training and optimizer/parameter offloading strategies on training efficiency and memory usage. Mixed precision training with FP16 and BF16 significantly reduced training time and memory usage without compromising model performance, highlighting its efficiency for large-scale NLP model training. Additionally, offloading optimizer and parameter states to the CPU effectively reduced GPU memory usage, allowing for larger models or batch sizes to be trained. However, this benefit came with increased training times due to the latency introduced by data transfer between the GPU and CPU. These findings highlight the significance of balancing memory management strategies with training efficiency in order to improve overall performance.

Additionally, we introduced AiBERTa, a custom base model with 110 million parameters developed and pre-trained on a corpus tailored for European Portuguese. AiBERTa's initial performance marks our first step towards improving NLP capabilities for Portuguese, contributing to the inclusivity and accessibility of NLP technologies for the language.

Having gained a better grasp and insight into distributed training and optimizations, we can now maximize the usage of system used to further improve our model. Future work will focus on leveraging these insights to enhance the performance of AiBERTa.

In conclusion, our study highlights the effectiveness of advanced tools and distributed computing in scaling NLP model training. By providing detailed insights into our methodology and experimental results, we aim to contribute to the expanding research dedicated to enhancing NLP for European Portuguese and other underrepresented languages. This research lays a foundation for future enhancements and optimizations in large-scale NLP model training and evaluation.

Acknowledgments. This work was supported by the Portuguese Foundation for Science and Technology (FCT) in the framework of the project AiBERTa - 2022.03882.PTDC.

References

1. BigScience: Bigscience language open-science open-access multilingual (bloom) language model. International (May 2022), project duration: May 2021-May 2022
2. Gamallo, P., Rodríguez Fernández, P., Santos, D., Sotelo, S., Miquelina, N., Paniagua, S., Schmidt, D., de Dios-Flores, I., Quaresma, P., Bardanca, D., Pichel,

J.R., Nogueira, V.: A galician-portuguese generative model (2024), submitted to publication

3. Garcia, G.L., Paiola, P.H., Morelli, L.H., Candido, G., Júnior, A.C., Jodas, D.S., Afonso, L.C.S., Guilherme, I.R., Penteado, B.E., Papa, J.P.: Introducing bode: A fine-tuned large language model for portuguese prompt-based task (2024)
4. Gugger, S., Debut, L., Wolf, T., Schmid, P., Mueller, Z., Mangrulkar, S., Sun, M., Bossan, B.: Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate (2022)
5. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima (2017)
6. Lopes, R., Magalhaes, J., Semedo, D.: GlórIA: A generative and open large language model for Portuguese. In: Gamallo, P., Claro, D., Teixeira, A., Real, L., Garcia, M., Oliveira, H.G., Amaro, R. (eds.) Proceedings of the 16th International Conference on Computational Processing of Portuguese. pp. 441–453. Association for Computational Linguistics, Santiago de Compostela, Galicia/Spain (Mar 2024), https://aclanthology.org/2024.propor-1.45
7. Miquelina, N., Quaresma, P., Nogueira, V.B.: Generating a european portuguese bert based model using content from arquivo.pt archive. In: Yin, H., Camacho, D., Tino, P. (eds.) Intelligent Data Engineering and Automated Learning – IDEAL 2022. pp. 280–288. Springer International Publishing, Cham (2022)
8. Muennighoff, N., Wang, T., Sutawika, L., Roberts, A., Biderman, S., Scao, T.L., Bari, M.S., Shen, S., Yong, Z.X., Schoelkopf, H., et al.: Crosslingual generalization through multitask finetuning. arXiv preprint arXiv:2211.01786 (2022)
9. Osório, T., Leite, B., Cardoso, H.L., Gomes, L., Rodrigues, J., Santos, R., Branco, A.: Portulan extraglue datasets and models: Kick-starting a benchmark for the neural processing of portuguese (2024)
10. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library (2019)
11. Rajbhandari, S., Rasley, J., Ruwase, O., He, Y.: Zero: Memory optimizations toward training trillion parameter models (2020)
12. Rodrigues, J., Gomes, L., Silva, J., Branco, A., Santos, R., Cardoso, H.L., Osório, T.: Advancing neural encoding of portuguese with transformer albertina pt-* (2023)
13. Santos, R., Silva, J., Gomes, L., Rodrigues, J., Branco, A.: Advancing generative ai for portuguese with open decoder gervásio pt-* (2024)
14. Shliazhko, O., Fenogenova, A., Tikhonova, M., Mikhailov, V., Kozlova, A., Shavrina, T.: mgpt: Few-shot learners go multilingual (2022). https://doi.org/10.48550/ARXIV.2204.07580, https://arxiv.org/abs/2204.07580
15. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: Glue: A multi-task benchmark and analysis platform for natural language understanding (2019)
16. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45. Association for Computational Linguistics, Online (Oct 2020), https://www.aclweb.org/anthology/2020.emnlp-demos.6

17. Yoo, A.B., Jette, M.A., Grondona, M.: Slurm: Simple linux utility for resource management. In: Feitelson, D., Rudolph, L., Schwiegelshohn, U. (eds.) Job Scheduling Strategies for Parallel Processing. pp. 44–60. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)