

**ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA**

**DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING**

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

**FINETUNING COMMERCIAL LARGE
LANGUAGE MODELS WITH LORA FOR
ENHANCED ITALIAN LANGUAGE
UNDERSTANDING**

CANDIDATE

Jens Hartsuiker

SUPERVISOR

Prof. Paolo Torroni

CO-SUPERVISORS

Anna Elisabetta Ziri

Dario Fioravante Alise

Federico Ruggeri

Academic year 2022-2023

2nd Session

Contents

1	Introduction	1
2	Problem description & Thesis Aim	4
3	Related papers and Theory	7
3.1	Transformer Pipeline	7
3.1.1	Tokenizer	8
3.1.2	The Transformer Itself	9
3.1.2.1	The Embedding Layer	10
3.1.2.2	Positional Encoding	11
3.1.2.3	Multi-Head Attention	12
3.1.2.4	Attention-Heads	13
3.1.2.5	Masked Attention-Heads	14
3.1.2.6	Feed Forward	16
3.1.2.7	Add & Normalization	16
3.1.2.8	Cleanup	17
3.1.3	Loss Functions for Transformers	17
3.1.4	Temperature	18
3.2	Decoder-Only Transformer	19
3.3	LoRA - low rank adaption	20
3.4	ALiBi - Attention with Linear Biases	24
4	Models & Data used	28

4.1	Models	28
4.1.1	MPT-7B-Instruct	28
4.1.2	LLaMA 2	30
4.2	Data	31
4.2.1	Exploring Stambecco	32
4.2.2	Quality remarks on Stambecco	34
4.2.3	The Conception of Stambecco	35
4.3	Camoscio	36
5	Pipeline & Practical insights	38
5.1	HuggingFace and the PEFT Library	38
5.2	MPT Architecture	39
5.3	Practical Insights	41
5.3.1	Infinite Generation Issue	42
5.3.2	Memory Usage Considerations	43
5.3.3	Background Processes	45
6	Real business usecase: The Bandi project	47
6.1	Project Overview	47
6.2	Pre-AI Team Process	48
6.3	The AI Team's Contribution	49
6.4	Examples	49
7	Experiments run	54
7.1	Training Models	54
7.1.1	Finetuning MPT	56
7.1.2	Finetuning LLaMA 2	58
7.1.3	Summary	60
7.2	Evaluating Language Comprehension	61
7.3	Evaluating Models on the Bandi Project	62
7.3.1	Temperature	66

7.4	Analyzing Model Outputs	66
8	Results	68
8.1	Exploratory analysis	68
8.2	Finetuning’s effectiveness	70
8.3	Prompt language and response quality	72
8.4	Answer language and response quality	73
8.5	Instruction location and response quality	75
8.6	Instruction formulation and response quality	76
8.7	Correct answers per tender/topic pair	78
8.8	Patterns	80
9	Discussion	81
9.1	Base model comparison	81
9.2	Models learning	82
9.3	Overfitting	83
9.3.1	Too potent/complex model	83
9.3.2	Insufficient amount of training data	83
9.3.3	A noisy/inaccurate dataset	84
9.3.4	Summary	84
9.4	Results finetuning	86
10	Conclusion	87
11	Future work	89
Bibliography		91
A	Tables of finetuning losses	95
B	Remaining prompts used for testing	100
Acknowledgements		103

List of Figures

3.1	The original transformer architecture from the 'attention is all you need' [33] paper.	12
3.2	An alternate depiction of the dataflow within a transformer.	15
3.3	The original attention mechanism illustration from the 'attention is all you need' [33] paper.	16
3.4	The decoder-only transformer architecture from the 'Improving Language Understanding by Generative Pre-Training' [19] paper.	20
3.5	The LoRA mechanism illustrated.	22
3.6	A comparison of various extrapolation methods.	27
5.1	The architecture of the repeating blocks of the MPT model.	40
7.1	Both the training- and validation loss from our 3 MPT Finetuning trials.	56
7.2	Both the training- and validation loss from our 3 LLaMA 2 Finetuning trials.	59
8.1	The number of correct, semi correct and wrong answers for each model. The total number of questions answered for each model is 648.	70

8.2	We show for each model the number of times it responds in the requested language, separated by the requested language itself. The total number of questions answered for each model is 648.	72
8.3	We show for each model the number of correct answers separated by the prompt language. The total number of questions answered for each language/model combination is 324.	73
8.4	We show for each model the number of correct answers separated by the language in which we ask the model to respond. The total number of questions answered for each language/-model combination is 324.	74
8.5	We show for each model the number of correct answers separated by the prompt location. The total number of questions answered for each location/model combination is 216.	75
8.6	The number of correct responses for different prompt formulations, shown for each topic. he total number of questions asked for each prompt/model combination is 72	76
8.7	The number of correct responses for different instruction/bandi (input) pairs	79

List of Tables

3.1	An example of a tokenizer vocabulary.	9
3.2	An overview of the parameters relevant for the transformer.	10
3.3	An example of the $b \times n \times n$ matrix in the non masked attention head.	13
3.4	An example of the $b \times n \times n$ matrix in the masked attention head.	15
3.5	Loss values and their corresponding confidence levels.	18
3.6	An overview of the LoRA parameters.	24
3.7	Standard attention mask.	26
3.8	ALiBi attention mask.	26
4.1	Entries from the Stamblecco dataset.	33
7.1	The hyperparameter differences between trials.	55
9.1	Loss of various models on the Stamblecco- and Alpaca dataset.	85
A.1	The table of the training- and validation losses corresponding to the first MPT trial, figure 7.1a.	95
A.2	The table of the training- and validation losses corresponding to the second MPT trial, figure 7.1b.	96
A.3	The table of the training- and validation losses corresponding to the third MPT trial, figure 7.1c.	96
A.4	The table of the training- and validation losses corresponding to the first LLaMA trial, figure 7.2a.	97

A.5	The table of the training- and validation losses corresponding to the second LLaMA trial, figure 7.2b.	98
A.6	The table of the training- and validation losses corresponding to the third LLaMA trial, figure 7.2c.	99

Chapter 1

Introduction

The world of Artificial intelligence is booming, especially the fields of natural language processing (NLP) and Computer Vision (CV). Recently the chatGPT models from OpenAI have caused another shock-wave. This was a clear sign to the world of what AI was capable of.

In our own environment we see the new course of Artificial Intelligence at the University of Bologna (UniBo), currently in its fifth year since its inception, and the current hiring surge at PriceWaterhouseCoopers (PwC) among its other investments in AI, as clear signs that the field is growing rapidly, both in industry and in academics.

It is then also with these three parties that we entered in a curricular internship agreement.

PwC has many projects dealing with text and documents, think about their judicial-, tax- and audit branches, which provide plenty of possibilities for automation from an AI/NLP point of view. Currently they are using OpenAI's GPT models to automate many of these usecases.

We personally chose to apply to PwC as we wished to gain industry performance. As intern, we were required to select our thesis topic from the range of subjects aligning with PwC's interests, primarily in the field of NLP. This constraint posed no issue for us, given our broad interests and the abundance of engaging projects within this scope.

We ended up choosing a project with as goal to produce a Large Language Model (LLM) tailored to the Italian language. Our reasons for choosing this project are twofold.

Firstly we have a more personal reason. As an international student from the Netherlands ourselves, we have been committed with learning Italian from the day we arrived. In a time-span of two years we have learned Italian without any prior knowledge up to about a C1 level. We are then also performing this internship in Italian. So building an LLM fit for the Italian language seemed the poetic cherry on top to conclude our academic journey.

Secondly, our motivation is driven by technical considerations. Developing LLMs is a complex undertaking, laden with technical challenges owing to their resource-intensive nature. At the project's outset, we anticipated the need to implement certain papers aimed at reducing this resource intensity. This expectation drew us towards the prospect of working at a low level with deep neural network code, a prospect that we found utmost appealing.

In the course of our thesis, we engaged in extensive readings and practical experiments. We commenced with a comprehensive literature review, delving into subjects such as the Transformer architecture and exploring recent research papers, including "Low Rank Adaptation (LoRA)" and "Attention with Linear Biases (ALiBi)." Subsequently, we initiated fine-tuning trials on

our commercially licensed models, MPT and LLaMA 2, leveraging programming libraries such as PEFT from Huggingface. Finally, we subjected our fine-tuned and base models, alongside the current state-of-the-art model, GPT 3.5, to rigorous testing against a real-world PwC use case. This evaluation process led to the analysis of 3,240 model responses.

This extensive endeavor yielded compelling results and findings.

Chapter 2

Problem description & Thesis Aim

Generative artificial intelligence has had a global impact, including within the consultancy sector, also here in Italy. PricewaterhouseCoopers (PwC) has identified numerous use-cases and opportunities that can be efficiently automated through the application of generative AI, both internally and externally. In our role as intern, we have been exposed to various such use-cases, thus gaining an immense appreciation for the implications of this technology.

Given PwC's extensive operations, the majority of our use-cases originate from various departments, such as Tax, Law, and Audit. As a result, nearly all our use-cases are text-centric, making our AI-team predominantly focused on text-generative AI.

Currently, we have two options to leverage generative AI. Firstly, we can utilize service providers like OpenAI, which offer state of the art models. As far as our knowledge goes, ChatGPT represents the current state of the art (SOTA) in text generation. The alternative approach involves the in-house development and -maintenance of large language models.

Both approaches have their respective advantages and disadvantages.

The advantage of maintaining our own LLM instance is our independence from external providers. However, a notable disadvantage is the considerable challenge of achieving performance on par with, or superior to, the current state of the art. Our practical experience has shown that the existing commercially licensed pre-trained models do not exhibit the same performance quality with Italian input as ChatGPT, thus leading to suboptimal results. Additionally, the availability of models under commercial licenses is limited. Then, perhaps most importantly, training an LLM from scratch can be prohibitively expensive. Frequently the training of LLMs costs up to hundreds of thousands of dollars. MPT, an LLM relevant for our thesis, incurred costs of approximately \$200,000 [23] for its pretraining.

Opting for a service provider offers the advantage of constant access to state-of-the-art models with minimal maintenance requirements. On the flip side, reliance on a service provider carries the risk of codebase entanglement with their API, potentially leading to difficulties in case of price hikes.

PwC has chosen to primarily utilize the services of OpenAI while investigating the possibilities of maintaining in-house LLMs as a risk management strategy. According to the AI department's management, it is one thing to sever ties with a service provider and face the consequences of disabling dependent services, and quite another to have in-house LLMs as a fallback option, albeit with (slightly) diminished performance.

Presented with the costs of pretraining, the current idea of building an Italian LLM is that of finetuning an already pretrained- and commercially licensed LLM on an Italian instruction dataset, hoping to achieve comparable performance.

This brings us to the objective of our thesis: "*Fine-tuning commercially licensed LLMs for application in PwC's Italian language-based use-cases, aiming for performance that does not significantly lag behind ChatGPT or other service providers*".

This thesis, therefore, serves as the first steps of development of a 'safety net' for all of PwC's AI applications.

Chapter 3

Related papers and Theory

This section provides an in-depth exploration of the theoretical foundations of our model’s architecture and discusses relevant research papers individually. It serves as supplementary material to aid newcomers in comprehending the current state of the art, addressing knowledge gaps, and contextualizing the discussed papers ([33, 19, 17, 12]).

3.1 Transformer Pipeline

In this subsection, we offer a comprehensive explanation of transformers, originally introduced in the ”Attention Is All You Need” paper [33]. We will cover the topic from a high-level abstraction down to intricate details. In essence, a transformer’s core function is to predict the next most likely word given a sequence of words.

To provide more precision, it’s crucial to note that transformers operate on tokens rather than full words. A token can be a character (’a’, ’b’, ’!’, etc), a part of a word (’ba’, ’wal’, ’rol’, etc) or even an entire word (’cat’, ’bag’, ’wall’, etc). Then, a transformer does not predict the next token, instead for all tokens it knows it calculates the probability that that specific token is in fact the next token. Rephrasing our sentence from before it becomes: given a

sequence of tokens a transformer calculates for each token known to the tokenizer that that token is in fact the next token.

A noteworthy observation is that transformers do not produce complete sentences. When employed in a pipeline, they iteratively predict the next token, extending the original input sequence by appending the predicted token. This process continues until the desired sequence length is achieved or until the model produces an end-of-sequence (EOS) token.

Transformers, therefore, use their previous predictions to make subsequent predictions.

3.1.1 Tokenizer

The first component to dissect in the transformer pipeline is the tokenizer. This crucial element converts textual input into numerical representations known as tokens. For the sake of conceptual clarity, we can envision the tokenizer as maintaining a mapping of character sequences to corresponding numerical values, as exemplified in Table 3.1.

The general approach of a tokenizer to tokenize a string is to look up the first character that does not belong to a token yet and see if it exists in the vocabulary. If so save the corresponding token and continue looking if the current character(s) plus the next new one exist in the vocabulary, if so save new token and repeat. Else output the current token and start a new search process with the current character. Now having the vocabulary shown in table 1 and the string "abd" to tokenize, lets see the tokenizer would handle it.

The character 'a' exists in our vocabulary, so for now we would output the token 2. Now let's check if 'ab' exists in the vocabulary, it does! For now our

Characters	Token
...	...
a	2
b	3
c	4
d	5
...	...
z	28
A	29
B	30
...	...
Z	54
0	55
...	...
9	64
ab	65
abc	66

Table 3.1: An example of a tokenizer vocabulary.

output would be token 65. Now we check if 'abd' exists in the vocabulary, it does not. Now we actually output token 65. Then we look up character 'd' separately, it is known as token 5. So for now our output would be token 5. Then our string ends. We now output token 5. Hence the output of the tokenizer is 65 and 5.

3.1.2 The Transformer Itself

This section delves into the architecture of the transformer. We commence by defining the relevant parameters, as presented in Table 3.2.

A transformer accepts input as a collection of b vectors, each containing n tokens. Here, b can be set to 1 or more, hence transformers can process one or more sequences of tokenized text in parallel.

Parameter	Description
b	How many sequences a transformer treats in parallel
n	The length of a sequence in number of tokens
vocab_size	How many tokens the tokenizer contains. A typical value is 50432.
model_dimension	The size of the embedding vector within the transformer that represents a token. A typical value is 2048.
no_blocks	A transformer consist of pieces of repeating architecture, blocks. This parameter denotes how many, a typical value is 32.
no_attention_heads	Blocks are among others divided into attention heads. This parameter denotes how many, a typical value is 32.
attention_head_length	The size of the embedding vector within an attention head. A typical value is 64.
expansion_ratio	This parameter denotes how many times the embedding vector grows in shape in the MLP. A typical value for this parameter is 4.

Table 3.2: An overview of the parameters relevant for the transformer.

3.1.2.1 The Embedding Layer

Then the first thing that happens is that the this collection of vectors passes through an embedding layer which transforms every token into a vector, resulting in a matrix of $b \times n \times model_dimension$. If $model_dimension$ were equal to the vocabulary size ($vocab_size$) of the tokenizer, these vectors could be one-hot vectors. But often the $model_dimension$ is much smaller. In this matrix the n dimension represents the time-steps or different tokens, the $model_dimension$ dimension represents the model-ready representation of the tokens and lastly the b dimension represents the parallelisation of multiple input vectors passing through the model at the same time. Keep in mind that in order for inputs to be processed in parallel they should be of the same length, being padded for example.

Now that we have concisely defined the input, we can also specify the exact output the transformer provides. A transformer does not just provide one prediction for the entire input. Instead it produces a prediction for every subsequence starting at index 0, so for an input with a length of n tokens we

get n predictions, one for each subsequence starting at index 0. Now we show an example of how a transformer transforms its input to its output: $[token_0 \ token_1 \ token_2 \ token_3] \rightarrow [token_1 \ token_2 \ token_3 \ token_4]$. The output here are predictions and are prone to errors. For illustration purposes we wrote tokens here, but actually the model produces probability vectors, having a value for each token in the vocabulary. At production time all these predictions serve no use, as we are just interested in the last predicted token, but that is just how transformers function.

3.1.2.2 Positional Encoding

Since the attention mechanism, which we will discuss later, does not capture information about the distance between tokens, it is necessary to introduce a "positional encoding" after the data has passed through the embedding layer. Positional encoding involves adding numerical values to the data based on a mathematical formula, which takes as input the position of the tokens. The original paper [33] uses sines and cosines for this function, while other papers, such as [17], present alternative, more effective approaches.

The input data then passes through blocks that are repeated a fixed number of times, typically around 32 repetitions. These blocks consist of specific components: Multi-Head Attention, Masked Multi-Head Attention, Feed Forward, and Add & Normalization, as illustrated in Figure 3.1. In this figure, the left half is referred to as the encoder and the right half is referred to as the decoder.

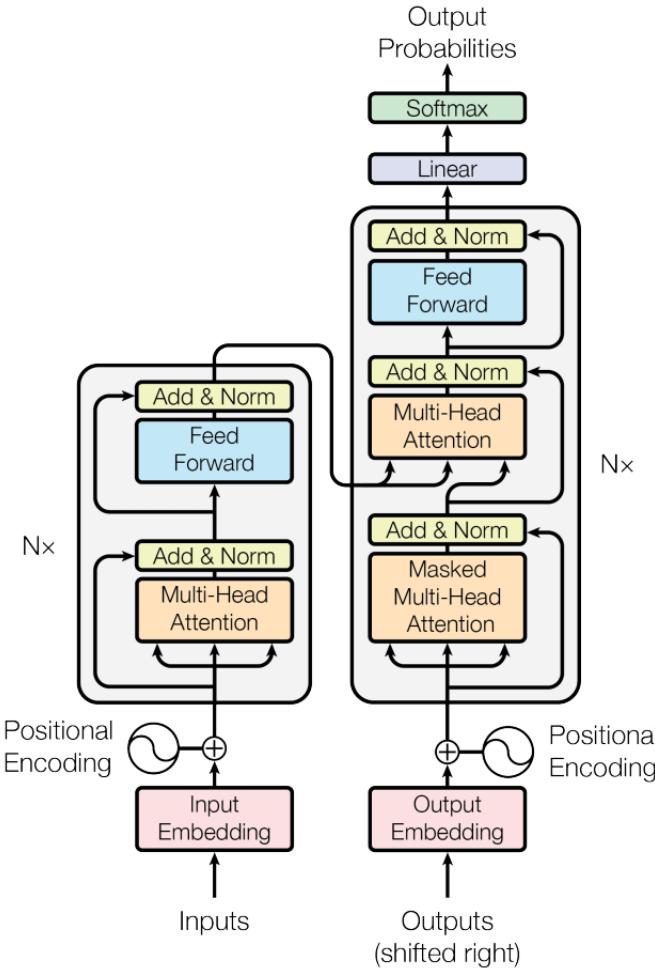


Figure 3.1: The original transformer architecture from the 'attention is all you need' [33] paper.

3.1.2.3 Multi-Head Attention

A Multi-Head Attention module takes an input vector of dimensions $b \times n \times model_dimension$ and directs it through $no_attention_heads$ attention heads, where $no_attention_heads$ is a user-defined parameter that determines the number of attention heads. Each attention head returns a vector of dimensions $b \times n \times attention_head_length$, where $attention_head_length = model_dimension/no_attention_heads$. These vectors are then concatenated to produce an output vector with the original dimensions: $b \times n \times model_dimension$.

3.1.2.4 Attention-Heads

Mathematically, the size of n has no bearing on the functioning of the attention head. In an attention head, the input (of dimensions $b \times n \times \text{model_dimension}$) is replicated to create three equal instances, each of which is processed through its dedicated fully connected layer. These layers reduce the *model_dimension* to *attention_head_length*. The three resulting vectors are named K , Q , and V , each with dimensions $b \times n \times \text{attention_head_length}$.

Next, a dot product is computed between K and Q , yielding a matrix of dimensions $b \times n \times n$. This matrix reveals the relevance between pairs of tokens. For example, in the sentence "Peter walks, and the dog stinks," it will indicate high relevance between "Peter" and "walks" and low relevance between "Peter" and "stinks." An example matrix is shown in Table 3.3.

	Peter	walks	and	the	dog	stinks
Peter	0.56	0.3	0.07	0.02	0.04	0.01
walks	0.37	0.61	0.01	0.01	0	0
and	0.24	0.02	0.44	0.05	0.23	0.02
the	0.02	0.01	0.01	0.59	0.31	0.06
dog	0.01	0.01	0.03	0.06	0.64	0.25
stinks	0.03	0.01	0.02	0.04	0.37	0.53

Table 3.3: An example of the $b \times n \times n$ matrix in the non masked attention head.

This matrix is data-dependent because the fully connected layers for K and Q are trained. This is where a lot of the power of a transformer lies. The matrix is subsequently multiplied (dot product) by V , producing a vector of dimensions $b \times n \times \text{attention_head_length}$. This vector then passes through a fully connected layer before leaving the attention head.

For a more intuitive understanding of the key, value and query vectors we would like to share an analogy we found on an online forum [1]: "*The key-/value/query concept is analogous to retrieval systems. For example, when you search for videos on Youtube, the search engine will map your query (text in the search bar) against a set of keys (video title, description, etc.) associated with candidate videos in their database, then present you the best matched videos (values)*".

The primary goal of the attention mechanism is to share information between tokens. For example in the sentence "Peter is born in Groningen, he ...", we want that in some way there is an information flow such that it becomes clear that 'he' refers to 'Peter'.

Looking at figure 3.1 an important remark we want to make regarding the data flow within the transformer is that in almost every step all the data that flows into a component comes from its previous component, but there is one exception. In the non-masked attention head in the decoder, the key- and value vectors always come from the exit point of the encoder. We include figure 3.2 as additional information for understanding the flow of data within transformers.

3.1.2.5 Masked Attention-Heads

In the decoder section of the architecture, we employ a masked attention module. Unlike the regular attention head, the masked version restricts the flow of information from the future to the present. Specifically, token at index i can only access tokens at indices smaller than i , accomplished by masking the upper triangle of the dot product. For instance, the example in Table 3.3

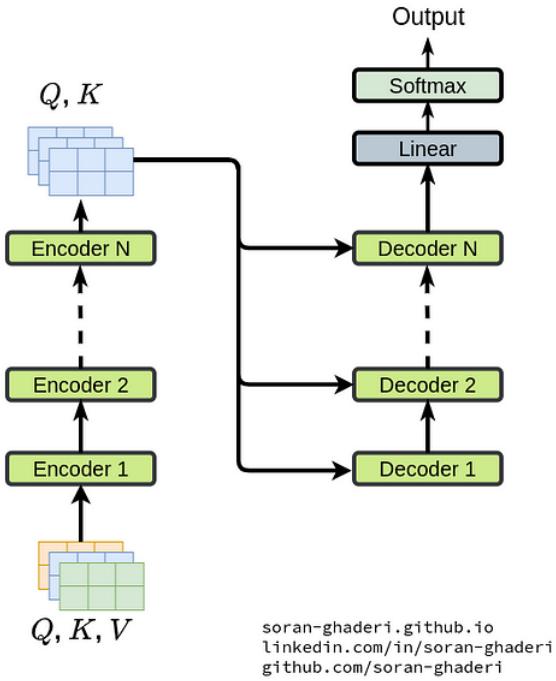


Figure 3.2: An alternate depiction of the dataflow within a transformer.

	Peter	walks	and	the	dog	stinks
Peter	1	0	0	0	0	0
walks	0.38	0.62	0	0	0	0
and	0.34	0.12	0.54	0	0	0
the	0.05	0.03	0.02	0.90	0	0
dog	0.01	0.01	0.03	0.11	0.84	0
stinks	0.03	0.01	0.02	0.04	0.37	0.53

Table 3.4: An example of the $b \times n \times n$ matrix in the masked attention head.

transforms into the masked version as shown in Table 3.4. This design enforces learning by preventing the model from merely copying future tokens from nearby entries and forcing it to make predictions and to learn.

For additional clarity we included Figure 3.3 from the original paper as illustration of the attention mechanism.

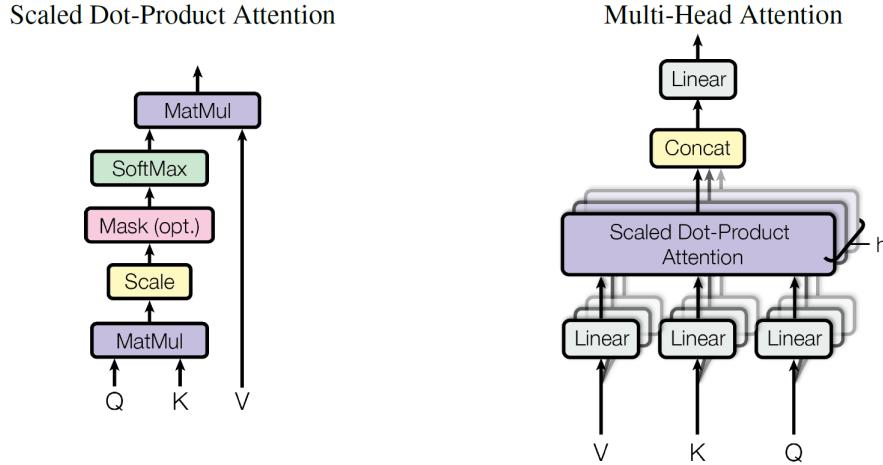


Figure 3.3: The original attention mechanism illustration from the 'attention is all you need' [33] paper.

3.1.2.6 Feed Forward

While the attention head's primary function is to establish information flow between tokens, the feed-forward component provides the space for the model to process and 'think' about this information. It employs a simple fully connected layer, transforming the data from dimensions $b \times n \times \text{model_dim}$ to dimensions $b \times n \times (\text{expansion_ratio} * \text{model_dim})$ before returning it to the original shape.

3.1.2.7 Add & Normalization

The "Add & Norm" component serves two purposes. Firstly, it combines two streams of the model, creating a skip or residual connection. Secondly, it applies layer normalization, specifically standardization, to the resulting values. These layers play a crucial role in addressing issues like vanishing and exploding gradients commonly encountered in deep neural networks. In-depth exploration of these topics and their solutions can be found in the paper that introduced the skip connection [11] and the paper on layer normalization [6].

3.1.2.8 Cleanup

At this stage of the transformer, two final steps remain. First, the data is transformed from dimensions $b \times n \times model_dimension$ to dimensions $b \times n \times vocab_size$. For each of the n predictions at each time step, this results in a probability distribution over the tokens in the vocabulary. A softmax function is then applied to ensure that these probabilities sum to one for each time step.

This concludes the description of the data flow within a transformer. For further reading we recommend [13], a valuable resource for low level understanding of transformers.

3.1.3 Loss Functions for Transformers

An often overlooked aspect of transformers is their loss function, which, though not particularly exciting, is necessary to mention.

The loss function used in transformers is the cross-entropy loss function, defined as:

$$H(p, m) = - \sum_{token \in vocab} p(x) \cdot \log_2(m(x))$$

Here, x represents the input to the model, $p(x)$ is the ground truth of the next token, and $m(x)$ is the model's prediction of the next token. Both $p(x)$ and $m(x)$ are vectors of length $vocab_size$. The summation is carried out over each element of these vectors.

We note that the structure of $p(x)$ is such that every element is zero, except

for the actual next token. Indicating that the entire equation depends solely on the model’s predicted value for the actual next token. We can now attribute meaning to our loss values. See the table 3.5 below.

Loss	Confidence
∞	0%
3.32	10%
2.32	20%
1.74	30%
1.32	40%
1.00	50%
0.74	60%
0.51	70%
0.32	80%
0.15	90%
0.00	100%

Table 3.5: Loss values and their corresponding confidence levels.

For example, a loss value of 1.32 indicates that the model is 40% confident that the ground truth token is indeed the next token.

In the field of NLP it is impossible for a model to be 100% sure. For example, for the following phrase *The lions eat the freshly caught {?}.*, both *zebra* and *antilope* are correct.

The final loss is computed as the average of all sequence losses, which is, in turn, defined as the average of the cross-entropy losses for all tokens in a sequence.

3.1.4 Temperature

When generating responses with our models using libraries from HuggingFace, we are required to specify the ‘temperature’. Although it is also possible that the temperature is already set behind the screens for some functions. The

temperature is a parameter passed to the softmax function at the final layer of the transformer. It determines how flat or peaky the distribution over the predicted tokens becomes. The higher we set this temperature the more we flatten the distribution. The lower we set the temperature the more we make the distribution peaky. The formula for the transformation of a single element q_i in the prediction vector q after the softmax is applied, is the following:

$$q_i = \frac{q_i / \tau}{\sum_{j \in q} q_j / \tau}$$

When the temperature is set closer to zero, the element with the highest probability approaches one. As the temperature approaches infinity, the distribution becomes completely flat, making it entirely random. Setting the temperature to one maintains the model's predictions as they are.

The choice of which tokens to select from the probability distributions is often determined by sampling methods such as top-k or top-p, but we won't delve into the details of these methods in this theoretical background as we consider them out of scope.

3.2 Decoder-Only Transformer

The model we use follows a decoder-only structure, as introduced in [14], which is identical to the encoder-decoder structure but omits the encoder part. See to Figure 3.4 for an example of a decoder-only model.

Currently, there is no consensus on whether a decoder-only transformer architecture is superior to an encoder-decoder architecture. The choice between the two architectures depends on the task, with empirical evidence showing that decoder-only architectures tend to perform better in chat and instruction-related tasks [22].

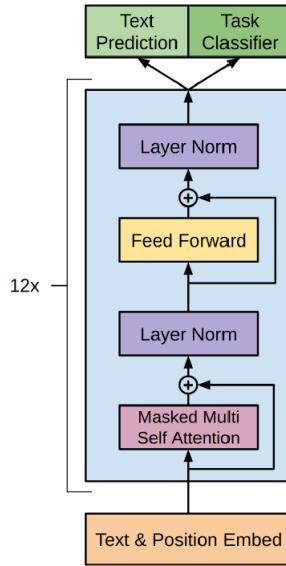


Figure 3.4: The decoder-only transformer architecture from the 'Improving Language Understanding by Generative Pre-Training' [19] paper.

3.3 LoRA - low rank adaption

In the world of Artificial Intelligence it has become more and more common to 'finetune' a deep learning model. With finetuning, or more specifically, full finetuning, is meant training the model for a second time on a different, and often smaller, dataset. In this full finetuning all original model weights can, and will be, modified. Finetuning is done to teach the model domain specific knowledge or to increase its performance on a specific task.

An example of finetuning for a specific task is preparing an LLM for question answering. Examples for teaching a model domain specific knowledge is teaching a model which classifies pictures of cats and dogs the difference between a Golden Retriever and a German Shepard. Or in our case, teaching Italian to predominantly English trained LLM.

However with the advent of bigger and more potent models (and all its possibilities) has also come a problem. Their incredible size of hundreds of billions of parameters makes them incredibly resource intensive to finetune. So expensive in fact that full finetuning of such models is only at the disposal of particularly wealthy parties such as big tech.

This where the authors of [12] come in. They tackle the resource problem described above. Furthermore their technique allows their model checkpoints to be stored at a minuscule fraction of current checkpoint sizes. Then, contrary to the current state of the art, their method does not introduce any latency at inference time. And, if that wasn't enough, their method also outperforms full finetuning in terms of loss.

In their proposed solution they freeze all the pretrained weights, i.e. the weights from the first training, and then add new weights themselves. The number of new weights the authors add is orders of magnitudes smaller than the number of pretrained weights. These new weights get added side by side to the already existing (fully connected) layers. Then in the finetuning stage only the new weights are updated.

A fully connected layer can be represented by a matrix multiplication of shape d, k , being the input and output dimension respectively. Their idea is that this multiplication can be approximated very accurately by two matrices of a lower rank B of shape d, r and A of shape r, k . Here r is the 'LoRA dimension', a parameter defined by the authors, with $r \ll \min(d, k)$. The authors found that this already worked well for r as small as 1 or 2. These matrices then get added to already existing flow of the model. See figure 3.5 for an illustration.

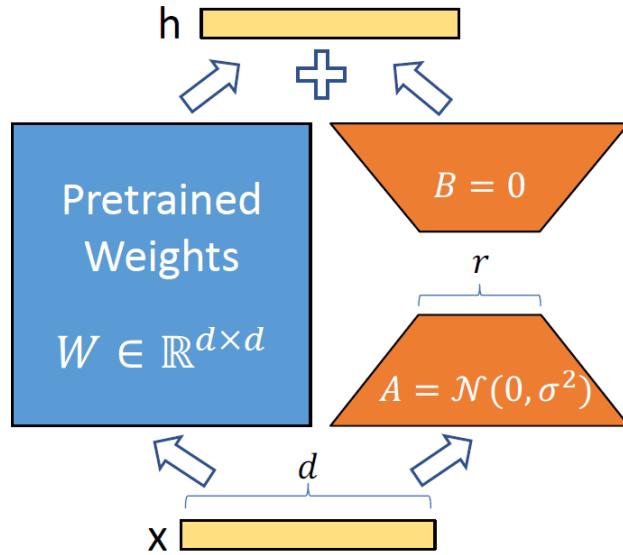


Figure 3.5: The LoRA mechanism illustrated.

To put it formally, the full finetuning formula as shown below:

$$\max_{\Phi} \sum_{(x,y) \in Z} \sum_{t=1}^{|y|} \log(P_{\Phi}(y_t|x, y_{<t}))$$

With LoRA becomes:

$$\max_{\Theta} \sum_{(x,y) \in Z} \sum_{t=1}^{|y|} \log(P_{\Phi+\Delta\Phi(\Theta)}(y_t|x, y_{<t}))$$

In these formulas Z is the dataset, with x being the input and y the corresponding output of one arbitrary entry. Then Φ denotes the parameters of the model of the pretraining/first training and Θ denotes the parameters of the LoRA method.

Looking at these equations we can clearly see how LoRA leaves the weights of the first training intact and only optimizes over the LoRA parameters.

The current state of the art methods for resolving the problem of too much

memory usage when finetuning consist of Prefix-embedding tuning, Prefix-layer tuning or Adapter tuning. These methods either inject new layers in such a manner that causes increase in runtime during inference or reduce the context length of the model as a side product. Inherent to how LoRA works, it does not cause these issues. LoRA updates the model weights directly, leaving the model architecture as it is, and not having any influence on the context length.

As mentioned before the LoRA method requires a parameter r as well as which modules to apply LoRA to, to be defined. The question remains what actually are good values for these parameters.

For the first question we look at table 6 from their paper. The authors apply finetuning with LoRA to various modules and various values of r (1,2,4,8 and 64), and we see that increasing r does not seem to have a correlation with model performance. In fact low values of r already perform competitively. Also figure 2 of their paper supports this, where performance does not improve as r increases.

For the second question also look at table 6, we see that performance almost always improves as we add more modules to LoRA. we look at table 5 from their paper. In this table they keep the amount of LoRA parameters constant. Then table 5 of their paper is interesting too. Here they test whether or not it is better to keep a high LoRA dimension applied to a few modules, or whether it is better to keep a low LoRA dimension applied to many modules. The answer is the later. This is then also what we applied to our own finetuning.

Then somethings else, thanks to the small LoRA dimension the checkpoints we have to save are very small, the checkpoint sizes are reduced about 10000 times. For example for GPT the new checkpoint size is reduced from 350GB

Parameter	Description
r	In how many dimensions modules should be approximated. Typically values such as 1 and 2 suffice
Alpha	The LoRA matrix is scaled by alpha, which is a parameter constant in r. When optimizing with Adam, tuning alpha is roughly the same as tuning the learning-rate. Hence during training this parameter is to be kept constant.
Modules	Which modules within the model to apply LoRA to
Dropout	LoRA can be trained with dropout, here said ratio can be specified

Table 3.6: An overview of the LoRA parameters.

to 35MB. Then because we freeze the model weights we also do not have to calculate the gradients or maintain the optimizer states for most parameters. This reduces the RAM usage up to 2/3 times.

Something very interesting to see is that LoRA finetuning actually often outperforms full finetuning, see e.g. table 4 of their paper. Why this actually is the case is unknown, as why and how finetuning works in generally is currently unknown too. A theory as to why LoRA outperforms full finetuning is that freezing the weights from the pretraining prevents the model from forgetting what it has learned in its pretraining. In their future works sections the authors actually mention that they hope that their work will contribute to an enhanced understanding of finetuning in general.

We conclude our section with table 3.6 explaining all parameters relevant to LoRA and its HuggingFace implementation.

3.4 ALiBi - Attention with Linear Biases

One of the inherent limitations when working with LLMs is the restricted context length. Expanding the model's capacity to handle longer sequences usually requires increasing the model's dimension, which, in turn, increases the

number of parameters and overall cost. Moreover, longer sequences result in extended training periods, further elevating costs.

However, as we discussed in a previous section on the transformer architecture, context length is not dependent on the model’s architecture. This observation gives rise to the concept of extrapolation, where models can predict sequences longer than those seen during training. The ALiBi paper [17] delves into this very concept. Based on perplexity measurements, it concludes that models trained with the original sine and cosine positional embeddings from the ”attention is all you need” paper [33] do not extrapolate well. Instead, the authors propose their own method, which exhibits excellent extrapolation capabilities. It allows for effective extrapolation on sequences 2-10 times longer than those encountered during training, all while speeding up model training.

The authors make their claims based on perplexity measurements. Perplexity is above all a measure of how confident a model is. Intuitively it is a measure of how likely a model deems a certain sequence X with length n . The lower the perplexity, the more the model deems a sequence likely. When comparing models such a sequence is often an entire dataset. Mathematically perplexity is defined as follows:

$$\text{perplexity}(X) = \exp\left(-\frac{1}{n} \sum_{i=1}^n \log_2(P_m(x_i|x_{<i}))\right)$$

Here, P_m represents the probability assigned by the model m .

Perplexity provides a computationally efficient way to gauge a model’s performance and to identify potential overfitting, as overfitted models often exhibit excessively high confidence. However we should note no measurement is perfect, perplexity (i.e. confidence) does not tell us how accurate a model is, or

more metaphorically, a model can confidently make mistakes.

ALiBi achieves its goals by eliminating positional embeddings from transformers. To encode positional information into transformers, they modify the masking in the attention mechanism. They replace the conventional binary attention mask of ones and zeros, as seen in Table 3.7, with their own as shown in Table 3.4.

1
1 1
1 1 1
1 1 1 1
1 1 1 1 1

Table 3.7: Standard attention mask.

0
-1 0
-2 -1 0
-3 -2 -1 0
-4 -3 -2 -1 0

Table 3.8: ALiBi attention mask.

In the attention mechanism, this is multiplied by a scalar m (slope), where each attention head has a unique slope. However, the set of slopes remains the same in each block. This set of slopes, of size n , is defined as:

$$\{2^{\frac{-8}{n}}, 2^{\frac{-8}{n} \cdot 2}, 2^{\frac{-8}{n} \cdot 3}, \dots, 2^{\frac{-8}{n} \cdot n}\}$$

ALiBi operates on this simple premise but yields impressive results, as shown in Figure 3.6. In this figure, the authors compare various extrapolation methods, including the one from the original transformer paper (sinusoidal embeddings), to their method. These methods are trained on 512- and 1024-token

sequences, and their perplexity on significantly longer sequences is measured. The original method demonstrates poor extrapolation with perplexity shooting up immediately. Other methods perform better, yet ALiBi outperforms both of them. Remarkably, ALiBi leads to an initial decrease in perplexity, followed by consistently low values, a remarkable outcome.

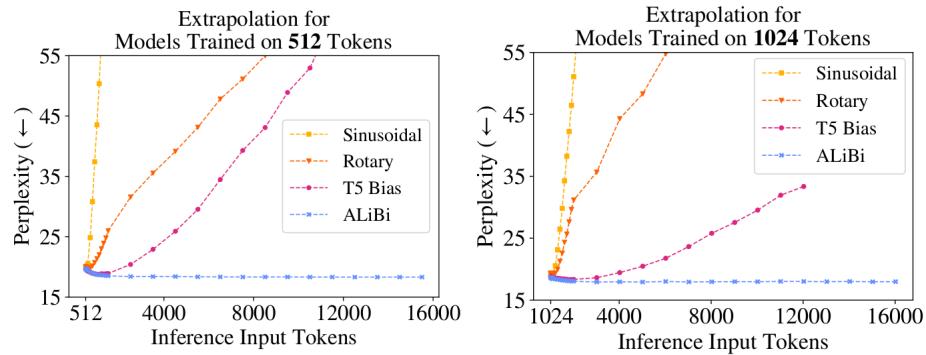


Figure 3.6: A comparison of various extrapolation methods.

Chapter 4

Models & Data used

To develop a commercially licensed language model that performs well in Italian, two crucial components are required: an open-source, commercially licensed model for fine-tuning and an appropriate dataset for the fine-tuning process.

4.1 Models

At the beginning of this project, only one commercially LLM was available: MPT-7B-Instruct [2]. However, around halfway through our project, another commercially licensed LLM, LLaMA 2, was released [3]. These two models serve as the focal points for our fine-tuning endeavors.

4.1.1 MPT-7B-Instruct

MPT, short for Mosaic Pretrained Transformer, is developed by Mosaic AI. The company's core business model revolves around providing software and

tools that expedite the development of AI solutions for businesses. Consequently, they released an open-source, commercially licensed LLM. Mosaic introduced several models simultaneously, encompassing a 30-billion-parameter family and a 7-billion-parameter family. In our research, cost-efficiency is a primary concern, so we exclusively focus on the 7-billion-parameter family. This family includes various models: MPT-7B, MPT-7B-Instruct, MPT-7B-Chat, and MPT-7B-StoryWriter-65k+. The latter three are fine-tuned variants of the MPT-7B model. Our primary interest lies in MPT-7B-Instruct, as it is fine-tuned on an instruction dataset. While MPT-7B-Chat might have been useful too, it lacks commercial licensing.

MPT-7B is a decoder-style transformer model, akin to OpenAI’s GPT, with 6.7 billion parameters and a context length of 2048, as well as its finetuned variants, except for MPT-7B-StoryWriter-65k+, which boasts a context length of 65,000. It was trained on 1 trillion tokens of text and code curated by Mosaic’s data team. The training data encompasses diverse datasets, including mc4, C4, Semantic Scholar ORC, and others. Mosaic specifically opted to utilize only the English subsets of these datasets. Another dataset they have included is called ‘The Stack’, this dataset concerns itself with coding questions and code itself and is most likely completely in English. Then lastly Mosaic also uses the CommonCrawl, arXiv, Wikipedia, Books, and Stack-Exchange subsets from the RedPajama dataset [32], which is an attempt to replicate LLaMA’s training data. We know that this Wikipedia subset contains some Italian data. Furthermore it is also possible that the Books subset contains some Italian data. Concluding: it is hence possible and certainly not excluded that MPT-7B is trained, if not only a little, on Italian data. We cannot be sure as Mosaic makes no statements on this topic rather than *“This dataset emphasizes English natural language text...”* [23].

In general, LLMs trained on diverse text corpora tend to underperform in

question-answering tasks. Therefore, after initial pretraining on a large text corpus, these models undergo instruction training using a dataset designed to teach them how to respond to questions. MPT-7B-Instruct is the result of fine-tuning MPT-7B on a combination of the Dolly dataset [9] and a subset of Anthropic’s Helpful & Harmless dataset [7].

After evaluation, Mosaic determined that MPT-7B-Instruct performs comparably to LLaMA 1. In a comparative analysis across 12 benchmarks, MPT-7B-Instruct outperformed other models on six benchmarks, while LLaMA 1 excelled on the remaining six, indicating similar performance between these models.

4.1.2 LLaMA 2

LLaMA 2 (and LLaMA 1) are Large Language Models developed by Meta, a company widely known as the parent company of social media platforms such as Facebook, Instagram, and WhatsApp. The majority of Meta’s revenue is generated through advertising on these platforms. However, Meta also maintains an AI division responsible for creating the LLaMA models.

The LLaMA 2 model family offers various variants, including models with 7 billion parameters, 13 billion parameters, and 70 billion parameters, all with a context length of 4096 and trained on 2 trillion tokens. This training dataset size is 40% larger than that of LLaMA 1. In our research, we focus on experimentation while keeping costs in check, which leads us to concentrate solely on the 7-billion-parameter model.

On Meta’s website, a comparative analysis is presented, encompassing various LLaMA 2 variants and models like MPT-7B and MPT-30B on 11 distinct

benchmarks. LLaMA 2-7B outperforms MPT-7B on all benchmarks except one, with the difference in performance ranging from minor to significant. The benchmark where MPT-7B performs better is "HumanEval," which evaluates the ability to produce functional computer code. Interestingly, LLaMA 2-13B also outperforms MPT-30B similarly, excelling across most benchmarks, except in "HumanEval," where MPT-30B performs substantially better.

It's crucial to clarify that while we occasionally mention LLaMA 1, our primary focus is on LLaMA 2, specifically the 7-billion-parameter variant. All our experiments are executed on LLaMA 2. This choice is guided by two key reasons: LLaMA 2 is the only commercially licensed model among the two, and it consistently outperforms LLaMA 1.

Lastly we would like to know what languages LLaMA 2 is trained on, but unfortunately very little information is released on this topic. We do know that LLaMA 1 has seen some Italian in its pretraining, then given that at least on an architectural level LLaMA 1 and 2 are not too different and given the fact that the training set of LLaMA 2 is 40% bigger than that of the first version, makes it not improbable that LLaMA 2 has seen Italian in its pretraining. However we can not be sure, furthermore on Meta's FAQ we have seen that "*the LLaMA models thus far have been mainly focused on the English language*" [4].

4.2 Data

The initial step in training a language model involves extensive training on a diverse text corpus, with the model tasked with predicting the next word in the text. This training dataset comprises hundreds of millions of words gathered from various sources, including books, internet content, music lyrics, and more.

Following this initial phase, models often exhibit limited performance in question-answering tasks. To improve their question-answering capabilities, a subsequent training phase is employed, known as instruction training. During this phase, models are trained on a dataset containing thousands to tens of thousands of question-and-answer pairs, teaching them how to answer questions effectively. It's important to note that both questions and answers in this dataset can be quite lengthy.

Both MPT-7B-Instruct and LLaMA-2-7B-Chat have undergone these two training phases. However, in the first phase, both models had minimal exposure to Italian text, if any at all. This exposure was most likely completely lacking in the second training phase.

In our attempts to teach these models a sufficient Italian language proficiency, we aimed to train these models on an instruction dataset exclusively in Italian. The dataset we selected for this purpose is known as Stamblecco [15], the only freely available Italian instruction dataset at the time of writing.

4.2.1 Exploring Stamblecco

The Stamblecco dataset comprises 51,713 entries, each containing three fields: instruction, input, and output. Notably, the instruction and output fields are always populated, while the input field may remain empty. The instruction field provides the model with specific tasks, such as "think of a name for my kitten", "write an essay about penguins", "correct the following sentence", or "put the following list in alphabetical order". The input field supplements the task with contextual information, including phrases to be corrected, such as "Mi name are Peter and I have 23 years," or lists to be sorted, such as "[Kiwi,

Strawberry, Mango, Grape].” The output field represents the expected model response, including corrected sentences, essays, or sorted lists.

Examples from the dataset, both with and without contextual input, can be found in Table 4.1.

Id	Instruction	Input	Output
0	Migliora la struttura grammaticale della frase data.	Mi sorprendo per la buona notizia	Sono rimasto sorpreso dalla buona notizia.
1	Componi una poesia selezionando a caso tre parole.		Un fiume piange per sogni dimenticati in luoghi lontani.
2	Nomina gli eventi storici che portarono alla Rivoluzione francese.		Le cause principali della Rivoluzione francese furono la disegualanza tra la nobiltà e il clero da un lato e la gente comune dall’altro, l’onere economico sulla popolazione causato dalle tasse, i raccolti scarsi e l’aumento dei prezzi del cibo e l’influenza dell’Illuminismo e la Rivoluzione americana. Altri fattori includevano la cattiva gestione finanziaria del re Luigi XVI, il crollo del commercio mediterraneo a causa della guerra con la Gran Bretagna e i Paesi Bassi e il numero crescente di contadini, artigiani e mercanti scontenti.
4	Modifica la frase data e assicurati che sia nella voce passiva.	Il cane ha inseguito il gatto.	Il gatto è stato inseguito dal cane.
9	Elenca le fasi lunari in ordine.		Luna nuova, Mezzaluna crescente, Primo quarto, Gibbosa crescente, Luna piena, Gibbosa calante, Terzo quarto, Mezzaluna calante.
10	Vediamo quale carattere dall’input dato è il più vecchio.	Max, Antonio, Gina	Max è il più vecchio dei tre personaggi.
12	Metti i seguenti elementi nell’ordine appropriato	Taccuino, penna, righello	Penna, righello, taccuino
15	Analizza la seguente frase nelle sue singole parole e rimuovi tutta la punteggiatura.	La comunicazione è fondamentale!	comunicazione, è, chiave!
18	Quale paese ha il maggior numero di laghi?		Il paese con il maggior numero di laghi è il Canada, con oltre 31.752 laghi.

Table 4.1: Entries from the Stambecco dataset.

The entry with the longest instruction has an instruction length of 1135 characters. This entry starts with a question of how to call a lazy and authority ignoring person, and then continues rambling on about how annoying a certain coworker is. The entry with the shortest instruction has an instruction length of 9 characters. The entry is: ”6 + 3 = ?”. The average instruction length is 67.6 characters and the median instruction length is 62 characters.

The entry with the longest input has an input length of 2925 characters. The entry was a news article from the CNN. The instruction to the model was to find a title for said article. The entry with the shortest input is the empty entry. The average input length is 26.2 characters and the median input length is 0 characters. Our dataset contains 32667 entries with empty input, which corresponds to 61.5% of the dataset. The average length of the inputs which

are not empty is 71.1 and the median length is 42.

The entry with the longest output has an output length of 4522 characters. The entry is the code for red-black tree implementation in C++. The entry with the shortest output has an output length of 1 character. The output in question is "7". The average output length is 312.6 characters and the median output length is 220 characters.

4.2.2 Quality remarks on Stambecco

The Stambecco dataset contains useful entries from which we think a model can gain a good comprehension of the Italian Language. Examples of such entries are 'improve the sentence' questions like entries 0 and 4 shown in table 4.1. But also just general language understanding questions like entries 1,2 and 9. From what we have seen all entries are grammatically and syntactically correct Italian.

However there are also entries that we think will throw the model off, like 10, 12, 15 and 18. Entry 10 asks specific information that cannot be assumed generally known without providing relevant context, basically asking the model to hallucinate. Entry 12 asks the model the model to put items in the appropriate order without specifying what appropriate is, it can only be assumed that alphabetically is meant here. Entry 15 asks to remove punctuation from the input, only to provide more punctuation in the output than originally present in the input. Entry 18 gives a technically correct answer, but it would have been better if they specified the actual number of lakes in Canada: 879800.

A future work could be to sift through the data and keep only a subset of quality prompts, as we have seen in [35], also just a subset of 1000 prompts can improve the model a lot. This paper also strongly suggest that almost all

knowledge in large language models is learned during pretraining.

Concluding, just looking at the collection of random samples of table 4.1, we see that Stambecco certainly, on a semantic level is not always a dataset of utmost quality, but currently there are no alternatives. We do want to emphasize that the quality of the Italian language in these entries are grammatically and syntactically correct.

4.2.3 The Conception of Stambecco

To understand the Stambecco dataset, it is essential to delve into its origins. The dataset's inception can be traced back to the paper "SELF-INSTRUCT: Aligning Language Models with Self-Generated Instructions" [34]. This paper introduced a novel technique that employed bootstrapping, using a small set of human-created instruction and answer pairs (approximately 175), and a pretrained model to generate a new instruction dataset. In the initial experiment, the "Davinci" model, part of the GPT-3 series, was used as the pretrained model. This technique proved effective in generating instruction datasets with substantial volume, approximately 52,000 entries, and of reasonable quality within the context of machine-generated data standards.

The Alpaca project [21], an initiative by Stanford aimed at creating an instruction-following LLaMA model, further refined this technique, making significant enhancements. They upgraded the model used for generation to "Davinci003," part of the GPT-3.5 series. Although they largely retained the pipeline from [34], they introduced simplifications and technical modifications that resulted in a dataset of higher quality and at reduced cost.

The dataset underwent another transformation when the authors of [16] sought

to enhance the answers within the Alpaca dataset. In this process, they duplicated all instruction and input tuples and generated new outputs using GPT-4.

The dataset’s final phase involved cleaning and translation. The cleaning process was undertaken by [5], who identified and removed entries with incorrect outputs, missing outputs, or hallucinations attributed to external references. The dataset’s translation was carried out by the authors of [15], leading to the creation of Stambecco, with translation encompassing all instruction, input, and output fields facilitated by GPT-3.5.

It’s important to note that Stambecco is a synthetic dataset, meaning that it is generated by machines rather than humans. While synthetic datasets are generally considered to be of lower quality than human-generated datasets, in the current AI landscape, using entirely human-generated datasets may not always be economically feasible.

Lastly it should be said that OpenAI does not allow their models (GPT etc) to be used to create models that compete with GPT. This could include our use of the Alpaca and Stambecco in our finetuning. However we are currently doing research and so our models are not competing with OpenAI’s models.

4.3 Camoscio

During the course of our thesis research, we also became aware of the Camoscio model [20]. Camoscio essentially represents LLaMA 1, fine-tuned on Alpaca translated into Italian using GPT3.5, a configuration akin to Stambecco. However, LLaMA 1 is not commercially licensed, so we still had to train models ourselves.

While Camoscio could have been used for testing and comparison, we chose

not to do so due to time constraints. Our research focuses on fine-tuning LLaMA 2, which outperforms LLaMA 1, making it more suitable for our research goal.

Chapter 5

Pipeline & Practical insights

The transition from pre-trained language models to practical applications often necessitates resource-intensive fine-tuning processes. In this section, we delve into how our project tackled these challenges, with a primary focus on harnessing the HuggingFace ecosystem, the "PEFT" library, and the Low Rank Adaptation (LoRA) technique.

5.1 HuggingFace and the PEFT Library

HuggingFace, a prominent entity within the AI community, serves multiple roles. It acts as a repository for a vast collection of pre-trained machine learning models, thereby making these models accessible on a global scale. But HuggingFace also provides an extensive suite of tools and programming libraries that facilitate the training and inference of these models, effectively establishing itself as a central hub for the AI community.

Within this ecosystem, the "PEFT" library, or Parameter-Efficient Fine-Tuning, emerges as a pivotal resource. It is tailored specifically for the efficient adaptation of pre-trained language models to a diverse range of downstream applications. What distinguishes PEFT is its ability to achieve this adaptation without necessitating the fine-tuning of the entire model's parameters. Instead, PEFT

concentrates on fine-tuning a relatively small number of additional model parameters, thus leading to a significant reduction in computational and storage costs. A substantial portion of these cost savings is achieved through the implementation of LoRA, as elaborated in Section 3.3.

For our project we initially believed we would need to implement LoRA ourselves. This would have involved making profound low-level modifications to model architectures and would have required a profound understanding of PyTorch’s functions. However, our journey took a fortuitous turn when we later discovered that Hugging Face’s PEFT library had already integrated LoRA. This alleviated the need for us to delve into the intricate low-level coding ourselves.

5.2 MPT Architecture

Despite the assistance provided by PEFT, a deep understanding of the model architectures designated for fine-tuning was imperative. At this stage, our primary focus was on the MPT model. The PEFT library required us to specify the layers within MPT where the LoRA matrices would be injected.

MPT, adheres to the common transformer decoder-only architecture, as illustrated in Figure 3.4. However, one notable deviation is that MPT does not use sinusoidal positional embeddings. Instead, it implements ALiBi, as elaborated in Section 3.4. MPT comprises 32 blocks, following the structure as depicted in Figure 5.1.

The attention blocks within these repeating blocks adhere to the same attention mechanism, as illustrated in Figure 3.3. The feedforward block within the repeating blocks consists of two layers following the shape transformation $d_{\text{model}} \rightarrow d_{\text{model}} \times \text{expansion ratio} \rightarrow d_{\text{model}}$, where $d_{\text{model}} = 4096$, and the

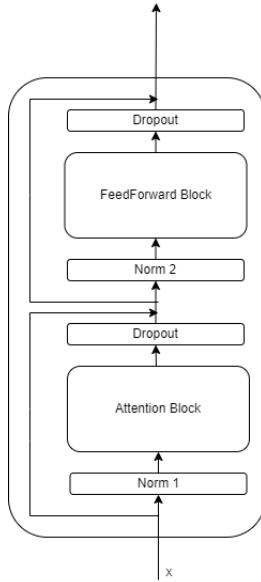


Figure 5.1: The architecture of the repeating blocks of the MPT model.

expansion ratio is 4.

To facilitate fine-tuning across as many model components as possible, we needed to pinpoint the instantiation of the Q, K, V, and the last linear layer of the attention model within the MPT codebase, as well as the layers related to the feedforward block.

In the MPT codebase, the q, k, and v layers are defined within the "Multi-headAttention" class in the "attention.py" file. The instantiation is denoted as follows:

```

1 self.Wqkv = nn.Linear(self.d_model, 3 * self.d_model, device=
    device)
  
```

The outgoing linear projection layer for the attention block is implemented in the same class, as shown below:

```

1 self.out_proj = nn.Linear(self.d_model, self.d_model, device=
    device)
  
```

These definitions are situated in the same file within the "layers" directory,

which resides in the "models" directory of the "llm_foundry" directory in the MPT codebase.

For the feedforward block, the structure is defined by two lines within the "MPTMLP" class in the "blocks.py" file, found in the same directory as the "MultiheadAttention" class:

```
1 self.up_proj = nn.Linear(d_model, expansion_ratio * d_model,
2                           device=device)
3 self.down_proj = nn.Linear(expansion_ratio * d_model, d_model
4                           , device=device)
```

By identifying these layer names, we enable PEFT to conduct fine-tuning on our MPT model.

Additionally, our exploration extended to the fine-tuning of the LLaMA 2 model. Interestingly, unlike MPT, LLaMA 2 did not necessitate specifying the layers for injecting Low Rank Adaptation. In fact, any attempt to do so resulted in errors. Unfortunately, there is also limited publicly available information regarding the architecture of LLaMA 2.

In summary, our approach to efficiently fine-tuning Large Language Models was empowered by the HuggingFace ecosystem, PEFT, and LoRA. These components facilitated our experiments while optimizing resource utilization.

5.3 Practical Insights

Beyond the presentation of experimental results, our thesis offers valuable practical insights and information pertinent to the development of the fine-tuning pipeline.

5.3.1 Infinite Generation Issue

An issue we encountered during fine-tuning, both for MPT-Instruct and LLama2, is that of infinite generation. During inference, a model should halt text generation when it reaches either the maximum token limit (a user-defined parameter) or an end-of-sequence (eos) token. The infinite generation bug occurs when the model fails to produce the eos token and continues generating text indefinitely, producing nonsense answers like this:

```
1 Gli eventi storici che portarono alla Rivoluzione francese
   includono la caduta del primo re di Francia, il re Carlo
   IX, nel 1792; la caduta del re Carlo X nel 1804; la caduta
   del re Carlo XIV nel 1815; la caduta del re Carlo XVI nel
   1815; la caduta del re Carlo XVI nel 1815; la caduta del
   re Carlo XVI nel 1815; la caduta del re Carlo XVI nel
   1815; la caduta del re Carlo XVI nel 1815; la caduta del
   re Carlo XVI nel 1815; la caduta del re Carlo XVI nel
   1815; la caduta del re Carlo XVI nel 1815; la caduta del
   re Carlo XVI nel 1815; la caduta del re Carlo XVI nel
   1815; la caduta del re Carlo XVI nel 1815; la caduta del
   re Carlo XVI nel 1815; la caduta del re Carlo XVI nel
   1815; la caduta del re Carlo XVI nel 1815; la caduta del
```

This is not an uncommon issue and we found that a three-step approach is effective:

- 1. Explicitly Setting the Eos and Pad Token:**

Ensure that the tokenizer has the eos and pad tokens explicitly set. The padding token should be initialized with its value, text, and ID, as specified in the tokenizer configuration. Additionally, ensure that the eos token is distinct from the padding token.

- 2. Adding the Eos Token to Training Data:**

Append the eos token to each sentence in the training data. This helps the model retain its ability to produce the eos token.

- 3. Strictly Following the Model's Prompt Template:**

To avoid incoherent predictions, ensure that the training data adheres to

the prompt template used during pre-training. Any deviation from this template structure can result in erratic model behavior.

Listing 5.1: The MPT prompt template

```

1  ### Instruction: {{prompt}}
2  ### Input: {{input}}
3  ### Response:{{gen}}
4

```

Listing 5.2: The LLaMA 2 prompt template

```

1 <s>[INST] <<SYS>>
2 {{system_prompt}}
3 <</SYS>>
4 {{user_msg_1}} [/INST] {{model_answer_1}} </s><s>[INST]
5   {{user_msg_2}} [/INST]

```

5.3.2 Memory Usage Considerations

During our training processes, we encountered recurrent issues with GPUs running out of memory, even when employing LoRA. Although LoRA contributes to substantial memory savings (2-3 times) by eliminating the need to save optimizer states of the Large Language Models (LLMs), memory management remained a challenge.

An approximate formula for memory usage during full training of an LLM can be described as:

$$M = M_{\text{model}} + M_{\text{activations}} + M_{\text{gradients}}$$

Where:

$$M_{\text{activations}} \approx \frac{BT^2}{4ND^2}$$

In this equation:

```
1 M = memory
2 B = batch size
3 T = sequence length
4 N = # of attention heads
5 D = dimension per head
```

In the context of transformers, activations consume the majority of memory resources.

To address memory challenges, we explored various techniques highlighted in the HuggingFace documentation [10]. These techniques include:

- **Model Quantization**

Reducing M_{model} by storing the model in lower precision, such as 16, 8, or 4 bits. However, this approach may substantially increase runtime, which we observed in our experiments. Further investigation is needed to determine whether this issue is inherent to quantization or specific to our pipeline.

- **Gradient Accumulation**

Focuses on limiting memory usage related to gradients.

- **Gradient Checkpointing**

This technique aims to reduce the number of activations saved during training. Given that activations are a significant source of memory usage in transformers, this approach proves highly beneficial. The article by its creators is an interesting read and can be found here [8].

- **Mixed Precision Training**

While primarily used to accelerate training, mixed precision training increases memory usage by approximately 50 percent.

- **Batch Sizes**

The batch size significantly influences memory usage. To mitigate out-of-memory errors, it is advisable to keep the batch size low.

- **Sequence Lengths**

Explicitly specifying the sequence length is crucial. Failure to do so may lead to increased memory usage, as models may rely on default values or values from unidentified configuration files.

5.3.3 Background Processes

We conducted our experiments on Google Cloud Vertex AI, which offers an environment similar to Google Colab but with more powerful GPUs, improved file management, and access to a terminal. While training our models, we initially employed a Jupyter notebook within the Jupyter environment. However, this method posed a risk of disconnection from Jupyter due to minor network issues, potentially disrupting the training process.

To address this concern, we transitioned to running our experiments as background processes through the terminal using the following command:

```
1 nohup python your_python_filename.py > your_output_filename.  
txt &
```

In this command:

- The ampersand (&) places the process in the background.
- The "nohup" command prevents the process from terminating when the terminal is closed.
- The ">" sign directs the terminal output to a specified file.

This approach ensured the continuity of experiments, even in the face of network interruptions.

In summary, this section not only presents experimental findings but also offers practical insights relevant to the fine-tuning pipeline, including solutions for issues like infinite generation, memory management, and the use of background processes in GPU-intensive experiments.

Chapter 6

Real business usecase: The Bandi project

In this section, we introduce a significant project undertaken by the AI team within PwC, in which we actively participated. The project, known internally as the "Bandi Project," holds substantial importance as it serves not only as an internal initiative but also has direct exposure to PwC's clientele.

The term "Bandi" originates from Italian, translating to "tenders" in English. Tenders are formal documents issued by government entities, outlining the eligibility criteria, purpose, and application conditions for accessing subsidies from specific government funds.

For more comprehensive information about the project, readers are encouraged to visit the official website at [18].

6.1 Project Overview

The core challenge addressed by PwC within this project revolves around the centralization and standardization of tender documents. These documents are

dispersed across various government websites, including municipalities, regional authorities, and national bodies. Furthermore, they often lack a consistent structure, occasionally even requiring the aggregation of information from multiple sources.

PwC's role is to assist companies by comprehending their unique needs and identifying potential business opportunities within the scope of available tenders. This approach allows PwC to connect companies with tenders that align with their specific needs.

Having provided an overview of the problem at a high level, we will now delve into the technical intricacies and elucidate how the AI team contributes value to the "Bandi Project."

6.2 Pre-AI Team Process

Prior to the involvement of the AI team, the process of monitoring tender documents encompassed a daily task. A group of five employees diligently sifted through a collection of approximately 160 websites to check for newly uploaded tender documents.

Upon discovering new documents, these employees were tasked with the extraction of specific data points, which were subsequently entered into a web platform. Examples of such data points include the opening date of the tender, expenses covered by the tender, size(s) of eligible companies, and region(s) the tender is active in. It is essential to note that these documents were composed in bureaucratic Italian, characterized by sophisticated language and frequent use of jargon. Consequently, these tasks were already considered challenging for human operators.

6.3 The AI Team’s Contribution

The AI team stepped in to alleviate the burden of data extraction, with the primary aim of significantly reducing the workload for the aforementioned five employees. While the technical architecture of our solution is outside the scope of our thesis, the essence of our approach involves the application of Large Language Models (LLMs) to snippets of text extracted from tender documents.

Specifically, we employ LLMs to extract specific pieces of information, such as the opening date of a tender. The effectiveness of our LLMs in responding to these prompts serves as the metric for evaluating their performance.

6.4 Examples

In this section, we present a selection of text snippets (input) along with their corresponding required outputs. To obtain the desired output, we utilize prompts, as further elaborated in Section 7. It is noteworthy that the formatting of the text snippets may appear irregular, featuring occasional newlines, misplaced numbers, and other discrepancies. This is a testament to the challenges inherent in extracting text from PDF documents and reflects the intricacies encountered in our pipeline and use case.

The examples provided pertain to the following topics: "the opening date of a tender", "the eligible expenses for a tender", and "the sizes of the companies eligible for the tender".

- 1 Sviluppo temporale del progetto
- 2 La durata degli impegni è di 5 anni dalla approvazione degli Accordi.
- 3 -> Indicazioni delle esigenze di consulenza
- 4 Descrivere dettagliatamente le esigenze di consulenza delle aziende aderenti nel territorio di competenza, con la dimostrazione della coerenza con le finalità dell'accordo. Sulla base di tale descrizione sarà valutata la coerenza dei progetti proposti dai richiedenti, rispetto all' Accordo Agroambientale d'Area a cui si riferiscono.
- 5 6.1.2 Termini per la presentazione della domanda
- 6 La domanda di sostegno può essere presentata a partire dal 12/09/2023 e fino al 12/10/2023 ore 13:00, termine perentorio. La domanda deve essere corredata di tutta la documentazione richiesta dal presente bando al paragrafo 6.1.3.
- 7 Saranno dichiarate immediatamente inammissibili:
 - 8 - le domande presentate oltre il termine;
 - 9 - le domande sottoscritte da persona diversa dal legale rappresentante o da soggetto delegato, o prive di sottoscrizione.
- 10 La verifica viene effettuata entro 10 giorni decorrenti dal giorno successivo alla scadenza di presentazione delle domande

Listing 6.1: An example of the context for extracting the opening date

1 12/09/2023

Listing 6.2: The output corresponding to the opening date example

- 1 ARTICOLO 4
- 2 DANNI AMMESSI A CONTRIBUTO
- 3 1. Costituisce requisito essenziale di ammissibilità l' esistenza del nesso di causalità del danno subito con gli eventi calamitosi di cui al presente bando.

⁴ 2. Sono ammessi a contributo i danni relativi a macchinari, strutture e arredi, veicoli aziendali, attrezzature, scorte di magazzino, spese per la predisposizione di perizie per la quantificazione dell'entità dei danni.

⁷ Non sono ammessi danni da lucro cessante (es. mancato profitto, etc.).

⁸ 2 Alfonsine, Bagnacavallo, Bagnara di Romagna, Brisighella, Casola, Valsenio, Castelbolognese, Cervia, Conselice, Cotignola, Faenza, Fusignano, Lavezzola, Lugo, Massa Lombarda, Ravenna, Riolo Terme, Russi, Sant'Agata, Solarolo

⁹ 3 Argenta limitatamente alla frazione Campotto

¹⁰ 3

¹¹ 3. Sono ammessi a contributo esclusivamente i danni effettivamente subiti e quantificati entro la data di scadenza del presente bando. I danni subiti dovranno essere indicati al netto dell'IVA e/o di altre imposte e tasse.

¹² ARTICOLO 5

¹³ REGIME DI AIUTO

¹⁴ 1. Gli aiuti di cui al presente bando sono concessi ai sensi del Regolamento CE n. 1407/2013 del 18 dicembre 2013, relativo all'applicazione degli articoli 107 e 108 del trattato sul funzionamento dell'Unione europea agli aiuti <de minimis>. Il Regolamento comporta che l'importo complessivo degli aiuti in de minimis concessi ad una medesima impresa, congiuntamente con altre imprese ad essa eventualmente collegate nell'ambito del concetto di "impresa unica", non debba superare euro 200.000 nell'arco di tre esercizi finanziari. Sudetto limite massimo è ridotto a euro 100.000 per le imprese appartenenti al settore dei trasporti su strada

Listing 6.3: An example of the context for extracting the eligible expenses

1 I danni relativi a macchinari, strutture e arredi, veicoli aziendali, attrezzature, scorte di magazzino, spese per la predisposizione di perizie per la quantificazione dell' entità dei danni.

Listing 6.4: The output corresponding to the eligible expenses example

1 BANDO INNOVAZIONE DIGITALE 4.0

2 ANNO 2023

3 Approvato con Deliberazione della Giunta camerale del 30 maggio 2023

4 Art. 1 - OGGETTO E FINALITA'

5 La Camera di commercio della Maremma e del Tirreno intende promuovere la diffusione della cultura e della pratica digitale nelle Micro, Piccole e Medie imprese di tutti i settori economici attraverso il sostegno economico alle iniziative di digitalizzazione dei processi aziendali.

Nello specifico, con questa iniziativa, si propone di promuovere l'utilizzo, da parte delle MPMI della circoscrizione territoriale camerale, di servizi o soluzioni focalizzati sulle competenze e tecnologie digitali nell'ambito delle attività previste dal Piano Transizione 4.01 a seguito del decreto del Ministro dello Sviluppo economico del 12 marzo 2020 che ha approvato il progetto "Punto Impresa Digitale" (PID).

6 Art. 2 - DOTAZIONE FINANZIARIA

7 La dotazione finanziaria iniziale a disposizione dei soggetti beneficiari ammonta a euro 90.000,00.

8 Nel rispetto dell'art. 5 ter del D.L. 24 gennaio 2012, n. 1 modificato dal D.L. 24 marzo 2012, N. 29 e convertito, con modificazioni, dalla L. 18 maggio 2012, n. 62 e tenuto conto del D.M. 20 febbraio 2014, n. 57 (MEF-MiSE), viene stabilita una riserva del 2% delle risorse finanziarie a favore delle imprese in possesso del rating di legalità.

Listing 6.5: An example of the context for extracting the sizes of the eligible companies

```
| Micro, Piccole e Medie imprese
```

Listing 6.6: The output corresponding to the sizes of the eligible companies example

Chapter 7

Experiments run

7.1 Training Models

As explained in the preceding sections, our primary objective is the development of an LLM optimized for the Italian language for application on PwC’s Italian use cases. To accomplish this goal, we employ a finetuning approach on the Stambecco dataset, followed by rigorous testing of our models against the Bandi use case. It is imperative to emphasize that our finetuning is conducted on the Stambecco dataset and not on the Bandi project.

The table of our most relevant trials is presented here, and we delve into a comprehensive analysis of these trials in the subsequent sections.

In addition to the variable hyperparameters outlined in Table 7.1, it is essential to highlight the constants within our experimental setup:

- **Hyperparameters**

The LoRA alpha value, a hyperparameter that is roughly equivalent to the learning rate, is held constant throughout all training phases with a value of 32. For further information on the hyperparameters, please refer to Sections 3.3 and 5.3.2.

- **Hardware Specifications**

Model	Trail No.	No. epochs	Training time (hrs)	Batch size	Gradient acc. steps	Gradient check-pointing	LoRA Finetune moduels	Lora dim. (r)	Lora dropout rate	No. LORA Params
MPT	1	15	17.5hrs	2	4	No	Wqkv, up_proj, down_proj	4	0.1	7340032
MPT	2	15	17.5hrs	2	4	No	Wqkv, up_proj, down_proj	2	0.2	3670016
MPT	3	15	17.5hrs	2	4	No	Wqkv, up_proj, down_proj, out_proj	1	0.2	2097152
LLaMA	1	15	22.5hrs	4	2	Yes	N.A.	2	0.2	1048576
LLaMA	2	25	37.5hrs	4	2	Yes	N.A.	4	0.2	2097152
LLaMA	3	15	22.5hrs	4	2	Yes	N.A.	8	0.1	4194304

Table 7.1: The hyperparameter differences between trials.

All training procedures are executed on a computational setup comprising two A100 GPUs, each equipped with 40 gigabytes of RAM. The market price of an A100 GPU is approximately 20,000 dollars.

- **Dataset and Split**

All models are consistently trained on the exact same dataset, namely the Stambecco dataset as described in Section 4.2. The train/test split has also been the exact same for each trial: 90% training data and 10% test data split, generated with a fixed seed of '42'.

Variations in Hyperparameters Between LLaMA 2 and MPT

In our attempt to finetune LLaMA 2 we used the same pipeline as we used for finetuning MPT. However, this caused memory issues. We found gradient checkpointing as our solution, as already explained in section 5.3.2. Gradient checkpointing in theory would increase the training time by 20%, but in practice it slowed down our training a bit more. However it did allow us to run the experiments on the same hardware still. Then the extra memory that technique freed up we could use to double the batch size and half the gradient accumulation steps to speed up the training a bit again.

7.1.1 Finetuning MPT

In our pursuit to finetune the MPT-Instruct-7B model, we conducted three trials, each comprising 15 epochs. The training duration for each of these 15-epoch trials was approximately 17.5 hours, with an hourly cost of \$5.50 and hence a total cost of \$96.25 per trial.

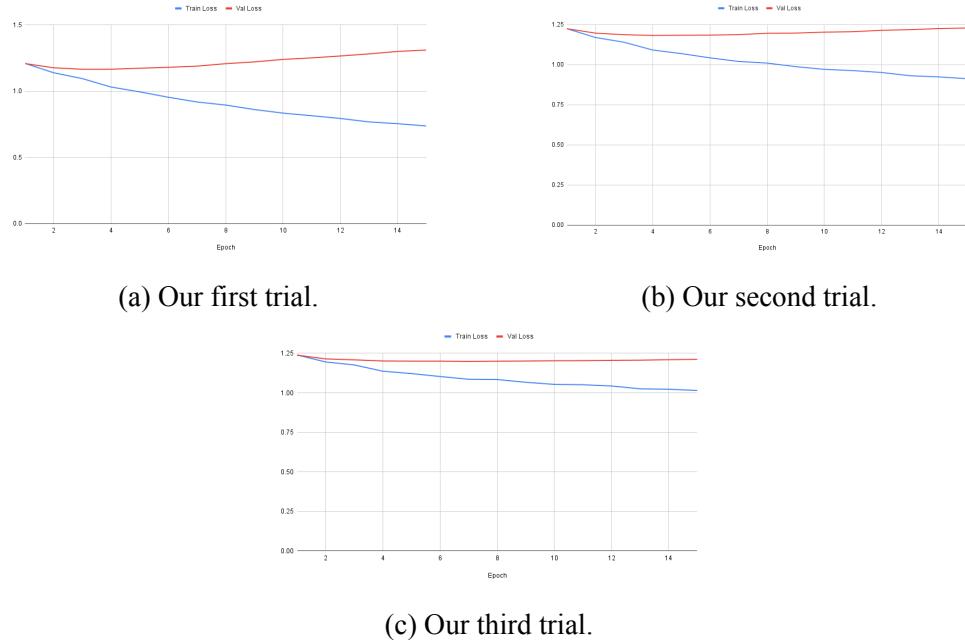


Figure 7.1: Both the training- and validation loss from our 3 MPT Finetuning trials.

Before starting our training, the initial loss on the MPT model’s validation set was measured at 2.0552. The performance of these three trials is summarized as follows:

In Figure 7.1a, the results of the first finetuning trial are presented. The loss after the first epoch was 1.2090, with the lowest validation loss occurring at epoch 3, registering at 1.1658, and the final loss concluding at 1.3108.

Figure 7.1b illustrates the outcomes of the second finetuning trial. The initial

loss after the first epoch was 1.2237, the lowest validation loss was observed at epoch 4, with a value of 1.1816, and the final loss was recorded at 1.2281.

Figure 7.1c presents the results of the third finetuning trial. The initial loss after the first epoch was 1.2383, the lowest validation loss occurred at epoch 7, registering at 1.1986, and the final loss was documented at 1.2115.

Additional tables with detailed information are available in Appendix A.

These figures unveil evident signs of overfitting, particularly noticeable in the first two trials, and to a lesser extent in the third trial. The potential causes of overfitting and its implications will be discussed in greater detail in the discussion section.

When training MPT we aimed to insert LoRA in every possible module. 'Wqkv' represents the linear layers after the streams of V, K and Q in the attention module as seen in the rightmost illustration in figure 3.3. 'out_proj' represents the final linear layer after the concat connection in the same illustration. 'up_proj' and 'down_proj' represent the entire feed forward module as illustrated in figure 3.1.

Then note that in our first two trials the 'out_proj' layer is not included (see table 7.1), this is our mistake as we originally missed this while analysing the MPT code. Adding this layer to the previous selection of LoRA layers increases the number of trainable parameters by about 14%. It would have been more scientifically correct to rerun trials 1 and 2 with these layers included within the finetuning, as this way our models would be more equally comparable. But given the cost of training we refrained from this. Also adding more parameters to an already overfitted model would most likely increase overfitting even more, so from this perspective it would also not have made much

sense.

To mitigate overfitting, we implemented two critical changes: increasing the dropout rate of the LoRA layers from 0.1 to 0.2 and reducing the LoRA dimension from 4 to 1. While the third trial still exhibited some overfitting, further model simplification was avoided due to diminishing returns and time constraints. Additionally, the release of LLaMA 2 prompted our interest in exploring finetuning on this model. For future research, potential adjustments, such as increasing dropout rates or reducing LoRA modules, can be considered.

Ultimately, the final checkpoint from the third trial was chosen for subsequent testing due to its stable training process with minimal overfitting.

7.1.2 Finetuning LLaMA 2

Finetuning LLaMA 2 comprised three trials: two 15-epoch trials and one 25-epoch trial. The 15-epoch trials each took 22.5 hours, while the 25-epoch trial required 37.5 hours. The hourly cost was 5.50, resulting in a cost of \$123.75 for the 15-epoch trials and \$206.25 for the 25-epoch trial.

Then we would like to point out that the modules used for LLaMA 2 in table 7.1 are specified as 'N.A.'. This is not because LoRA is not applied to LLaMA, it is, but this specific model did not allow us to specify which specific modules to inject LoRA into. Attempting to specify these modules resulted in error messages.

Before commencing training, the initial loss on the LLaMA 2 model's validation set was measured at 3.4196. The performance of the three trials is summarized as follows:

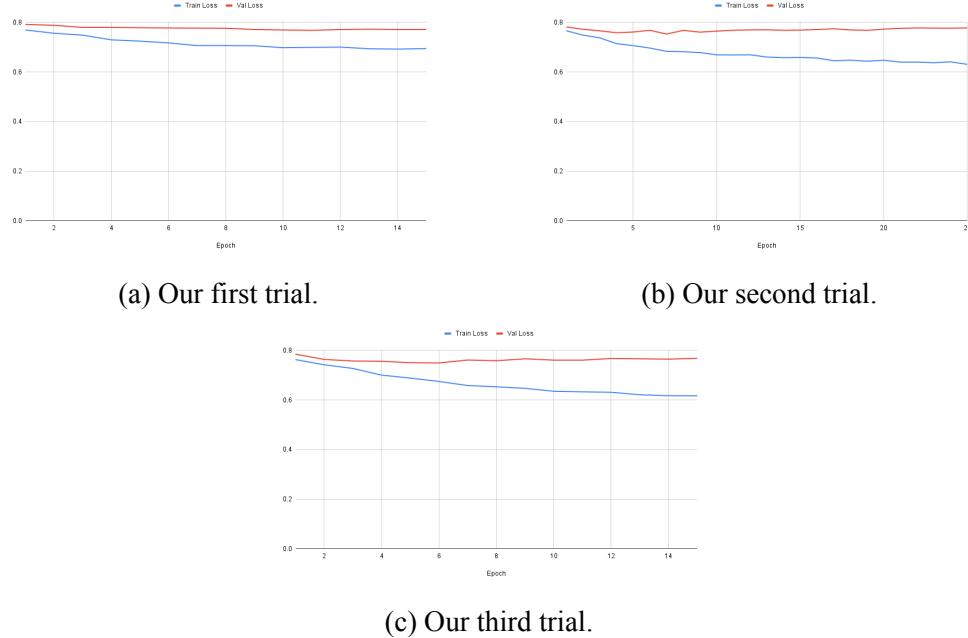


Figure 7.2: Both the training- and validation loss from our 3 LLaMA 2 Fine-tuning trials.

In Figure 7.2a, the results of the first finetuning trial are presented. The loss after the first epoch was 0.7916, with the lowest validation loss occurring at epoch 11, registering at 0.7678, and the final loss concluding at 0.7713.

Figure 7.2b displays the outcomes of the second finetuning trial. The initial loss after the first epoch was 0.7816, with the lowest validation loss observed at epoch 7, with a value of 0.7530, and the final loss documented at 0.7777.

In Figure 7.2c, the results of the third finetuning trial are presented. The initial loss after the first epoch was 0.7847, with the lowest validation loss occurring at epoch 6, registering at 0.7493, and the final loss concluding at 0.7681.

For more detailed information, please refer to the corresponding tables in Appendix A.

Having seen the issues with overfitting in our previous 3 trials, we started with a lower LoRA dimension and dropout than our first MPT trial. This resulted in a very stable decreasing loss curve that is not overfitted. Having seen these great results we increased the LoRA dimension as well as decreased the LoRA dropout in search of better results. We did not find better results and instead encountered overfitting. We then decided to bring the final checkpoint of the first LLaMA trial into real use case testing.

During the course of our experiments, we conducted a longer 25-epoch training run to assess potential performance improvements. However, no significant reduction in the loss was observed, leading us to revert to 15-epoch trials.

7.1.3 Summary

Upon reviewing these trials, we note that the loss curves of the MPT trials demonstrate smoother progression compared to the LLaMA trials. Despite these smoother curves, LLaMA models exhibit substantially lower losses. LLaMA 2 achieved a final loss of 0.768, while MPT recorded a loss of 1.212, marking a 37% difference.

We acknowledge that further research and resources could lead to models better fitted on the data. Nevertheless, our current trials provide models that sufficiently allow us to assess the effectiveness of our proposed methodology towards achieving our research objectives.

The cumulative training cost includes three 15-epoch MPT trials, two 15-epoch LLaMA trials, and one 25-epoch LLaMA trial, amounting to $3 \times 96.25 + 2 \times 123.75 + 206.25 = \742.5 .

7.2 Evaluating Language Comprehension

We possess two datasets that are nearly identical in semantic content. One is the Italian-translated variant, Stambecco, and the other is the original version in English, Alpaca. As these datasets are equal, the difference in losses a model m has on these datasets tells us something about difference of capability of m in Italian and English. There are two variants of this experiment.

The less resource-intensive approach involves evaluating LLaMA 2 and MPT on both the Alpaca and Stambecco datasets without prior finetuning on these datasets. When testing these models on the entire datasets, LLaMA 2 exhibits a loss of 2.602 on Stambecco and 1.822 on Alpaca. Conversely, MPT demonstrates a loss of 2.048 on Stambecco and 1.688 on Alpaca. The notably lower losses on the Alpaca dataset suggest that these models are better suited for English, signaling potential limitations in their proficiency with the Italian language. A more detailed discussion on these findings can be found in the discussion section.

The second, perhaps more scientifically correct, but more resource intensive experiment is to besides finetuning the models on Stambecco like we are already doing, to also finetune the models on the Alpaca dataset. We can then compare the losses of the models finetuned on Alpaca to the losses of the models finetuned on Stambecco. Then these models having had the chance to get to 'know' the data, we can judge their aptitude for the Italian- and English language. However, due to time- and resource constraints we refrained from this experiment.

Note that if one might want to undertake this experiment in a future work, that the alpaca- and stambecco dataset do not have the same order of entries and that the alpaca dataset contains 47 entries more.

7.3 Evaluating Models on the Bandi Project

In this section we will explain how we evaluate our models at the hand of the Bandi project. We want to emphasize that the tasks within the Bandi project pose a considerable challenge, not only for deep learning models but also for humans, given the complex and jargon-heavy juristic language used. In this section, we will provide insights into the dataset we have curated, the variables under investigation, and the criteria for evaluating the model responses.

We also want to stress that perhaps causal language modelling might not be the optimal technique for the Bandi usecase, instead one could look at purely extractive models. However PwC has chosen to use the service of OpenAI and hence causal language modelling in its software pipelines, for e.g. ease of use. Given that we are experimenting on building a backup plan for these services, it would not make sense to train a purely extractive model as this could possibly be incompatible with other usecases. Hence it is not our goal to build the best performing model on the Bandi usecase, our goal is to test our finetuned models against a real usecase. Might one instead only aim to get the best possible performance on the Bandi usecase, purely extractive question answering definitely becomes a viable option.

Our dataset consists of 18 text-snippet and answer pairs extracted from the Bandi project. These pairs are derived from six distinct tender documents, each encompassing information related to three topics: the tender opening date, the expenses financed through the tender, and the dimensions a company should have to be eligible to participate in the tender.

The tender documents we have selected are as follows:

- Bandi di finanziamento - Agricoltura Sviluppo rurale e Pesca - Regione Marche [26].
This tender is used for the 'opening date' and 'eligible expenses' topics.
- Avviso per contributo a sostegno del collegamento marittimo Pescara - Croazia [25].
This tender is used for the 'opening date' and 'eligible company sizes' topics.
- BANDO DI CONTRIBUTO PER LA PARTECIPAZIONE A EVENTI FIERISTICI - Alessandria [24].
This tender is used for the 'opening date', 'eligible expenses' and 'eligible company sizes' topics.
- Regione del veneto - BANDO PER LA CONCESSIONE DI CONTRIBUTI A SOSTEGNO DELLE ASSOCIAZIONI ENOGASTRONOMICHE - ANNO 2023 [31].
This tender is used for the 'opening date', 'eligible expenses' and 'eligible company sizes' topics.
- Bollettino Ufficiale della Regione Puglia - n. 54 del 15-6-2023 37243 - DETERMINAZIONE DEL DIRIGENTE SEZIONE TURISMO E INTERNAZIONALIZZAZIONE 9 maggio 2023, n. 129 [29].
This tender is used for the 'opening date' and 'eligible expenses' topics.
- Piccole imprese per il territorio Camera di Commercio Parma [28].
This tender is used for the 'opening date', 'eligible expenses' and 'eligible company sizes' topics.
- Bando Innovazione digitale 4.0 - Camera di Commercio Maremma e Tirreno [27].
This tender is used for the 'eligible company sizes' topic.
- CONTRIBUTO STRAORDINARIO ALLE IMPRESE PER IL RISTORO DEI DANNI SUBITI DAGLI EVENTI ALLUVIONALI DEL MAGGIO 2023 [30].
This tender is used for the 'eligible expenses' and 'eligible company sizes' topics.

Our context and answer pairs constitute our dataset. To enable the models to provide accurate responses, we must formulate prompts or "instructions". This is where the 'variables' of our experiment come into play. We have learned through experience at PwC that the phrasing of instructions can significantly impact a model's performance. For each topic, we have chosen to create three distinct prompts to ensure robustness and statistical reliability in our assessments. This approach helps avoid unjustly favoring or disfavoring a model due to a specific prompt's idiosyncrasies.

We aim to prevent any form of "prompt engineering," a practice that involves designing prompts to produce favorable results for a specific model. This can lead to biased comparisons. However, one exception to this is that for each topic, we include one prompt derived from a formulation optimized for GPT. This is based on prior work done by our colleagues on the project.

Another consideration we address is whether the prompts should be written in English or Italian. Both languages offer advantages: English aligns with the primary training language of the models, while Italian matches the language of the contextual documents. To comprehensively evaluate these factors, we provide prompts in both English and Italian. This gives us 18 prompts in total: we have 3 prompts per topic, and for each topic we have 3 different formulations, both in Italian and English which gives us $2 * 3 * 3 = 18$. All the 18 prompts can be found in this document, 4 are listed below and the others can be found in appendix B. We have indicated as well if they are derived from a GPT optimized formulation or not.

- Prompt 1 - Italian (GPT formulation, original): Immagina di essere un estrattore di metadati da bandi di gara. Dal contesto fornito estrai la data di apertura del bando. La data di apertura del bando rappresenta il giorno da cui il bando è attivo ed è possibile fare richiesta per accedere alle agevolazioni previste dall'ente erogatore. La data di apertura del bando può essere scritta in diversi formati, ad esempio: dd month yyyy, dd/mm/yy oppure dd month. All'interno del contesto fornito potresti trovare anche altre date che non sono la data di apertura del bando: nessuna di queste deve essere riportata nella tua risposta. La data di apertura del bando è spesso preceduta da stringhe come: 'dal giorno' o 'dalle ore del giorno'. La data di apertura del bando non è mai preceduta da stringhe come: 'entro il giorno' o 'fino al'. Inserisci nella tua risposta SOLAMENTE la data che rappresenta la data di apertura del bando nel formato dd/mm/yyyy. Se quella data non è presente nel contesto fornito puoi rispondere 'ND'.
- Prompt 1 - English (GPT formulation, translated): Imagine you are a tender metadata extractor. From the context provided, extract the opening date of the call. The opening date of the tender represents the day from which the tender is active and it is possible to apply to access the facilities provided by the provider. The opening date of the call can be written in different formats, for example: dd month yyyy, dd/mm/yy or dd month. Within the context provided you may also find other dates that are not the opening date of the call: none of these should be reported in your answer. The opening date of the announcement is often preceded by strings such as: 'dal giorno' or 'dalle ore del giorno'. The opening date of the call is never preceded by strings such as: 'entro il giorno' o 'fino al'. Insert ONLY the date that represents the opening date of the call in the format dd/mm/yyyy. If such a date is not present in the context provided, you can respond 'ND'.
- Prompt 2 - Italian: Sei un assistente AI che aiuta a estrarre dati da contesti specifici. Il contesto in questo caso sono bandi: documenti forniti dal governo che spiegano chi può ricevere finanziamenti, per cosa e a quali condizioni. Dal contesto che ti forniremo, vorremmo sapere quando apre il bando, cioè la data a partire dalla quale gli enti (aziende o persone) possono presentare domanda per usufruire dei benefici offerti dal bando. Se tale data non è fornita nel contesto, puoi rispondere 'ND'. Si precisa infine che i bandi sono in lingua italiana.
- Prompt 2 - English: You are an AI assistant that helps extract data from specific contexts. The context in this case are tenders: documents provided by the government explaining who can receive funding for what and under which conditions. From the context we are going to provide to you, we would like to know when the tender opens, i.e. the date from when entities (companies or people) can submit applications to make use of the benefits offered by the tender. If such a date is not provided in the context, you can reply 'ND'. Lastly, note that the tenders are in Italian.
- ...

After crafting these prompts, we realize that the language in which we request model responses, either English or Italian, can also impact the outcomes. To account for this, we append "respond in English" or "respond in Italian" to our prompts in English and "rispondi in Inglese" or "rispondi in Italiano" to our Italian prompts. This results in the following permutations:

- "{Italian prompt} Rispondi in Inglese."
- "{Italian prompt} Rispondi in Italiano."
- "{English prompt} Respond in English."
- "{Italian prompt} Respond in Italian."

These four phrases which specify in which language the model should answer, we refer to as language addendums.

With the prompts ready to go, we face the question of where to position the prompt relative to the input/context. We have identified three possible arrangements: before the input, after the input, or both before and after the input. The permutations of these prompt positions are as follows:

- "{Prompt with language addendum} {Input}"
- "{Input} {Prompt with language addendum}"
- "{Prompt with language addendum} {Input} {Prompt with language addendum}"

This question has also been brought to our attention partly due to our colleagues working with prompt engineering, who have seen that this factor has influence of the quality of the responses. With our experiments we hope to draw conclusions on the influence of this factor sustained by statistically robust data, as we currently do not have any.

In summary, to evaluate the models effectively, we manipulate several key factors: prompt language (English and Italian), prompt formulation (three variations per topic), model response language (English and Italian), and prompt

position relative to the input (before, after, or both). We perform this evaluation for three topics across the six tender documents, resulting in $2 \times 3 \times 2 \times 3 \times 3 \times 6 = 648$ prompt permutations, and equally many responses to analyze.

We assess the performance of five models: GPT, MPT Base, MPT Finetuned, LLaMA 2 Base, and LLaMA Finetuned. This gives us a total of $5 \times 648 = 3240$ prompt permutations , and equally many responses to analyze.

7.3.1 Temperature

In the process of generating responses, some libraries require the specification of the temperature hyperparameter. For GPT, we explicitly set a temperature of 0.15 based on practical experience from our colleagues, which is found to be optimal for the Bandi project. MPT also necessitates the specification of the temperature, and we choose a temperature of 0. When it comes to generating responses, LLaMA abstracts away the temperature parameter, using a default value of 0.9, specified in the "generation_config.json" file in the relevant HuggingFace repository. In the end where required we opted low temperatures because in our use case we do extractive question answering, no creativity is needed, the answer is 'there' so to speak.

7.4 Analyzing Model Outputs

To quantify the performance of the models, we categorize every answer for the 3240 prompts into one of three categories: correct, semi-correct, or incorrect. Each category is defined as follows:

- Correct: The model returns the requested data accurately, devoid of errors, and with little to fuss/filler text around the answer.

- Semi-correct: The model provides the requested data with an acceptable amount of fuss/filler text around the answer. Any answer containing falsehoods is not accepted in this category.
- Incorrect: Answers that contain falsehoods, excessive fuss/filler, or otherwise rubbish.

By classifying the responses into these categories, we can quantitatively assess and compare the performance of the models. We acknowledge that this process is resource-intensive, but we believe it offers a deep understanding of the model’s functionality. We explored alternatives, such as calculating loss over the Bandi dataset or requiring models to respond in a standard format. However, these methods proved inadequate, as even the state-of-the-art model, GPT 3.5, faces challenges in returning standard formats for straightforward questions in our use case. Our chosen method provides a more comprehensive evaluation.

Chapter 8

Results

As explained in the previous section, with our queries to the models we would like to answer a variety of questions: how effective is finetuning for the Italian language? How does the language of the prompt influence the quality of the responses of the model? Does the model give better responses in English or in Italian? In extractive question answering, how does the location of the prompt w.r.t. to the question’s context influence the response’s quality? And also, can we see patterns in these responses for the different models? In this section we will analyse our results and look for answers.

8.1 Exploratory analysis

In this section we want to provide some comments on how the models function, that are not necessarily specific to a certain section. We want to provide a sense for how these models function, how they perform well, what are their weakpoints and what are there peculiarities, basically we want to sketch an image of what we are dealing with here.

We have seen that in general the models all have a tendency to generate a lot of text, also when it is not necessary. For example regarding the question

of the opening date of the call as answer just the date would suffice: "dd/mm/yyyy". However the models have much longer responses: "The opening date is dd/mm/yyyy as can be found in ..." or "The tender is open from dd/mm/yyyy at hh:mm until from dd/mm/yyyy at hh:mm". LLaMA 2, especially the base variant, is an extreme example of this, it is almost like your aunt Betty who does not stop talking.

Then we have seen that when models answer incorrectly they can do all kinds of strange things. They can return "N.A." even when there is an answer present in the input/context, they can completely fabricate an answer e.g. name a date that is not present in the input/context, they can return a completely irrelevant piece of input/context or return the inverse of what we want to know e.g. the expenses a tender does *not* cover instead of those that a tender covers.

Then for MPT we have seen that the infinite generation bug as described in section 5 persists. Not only in the finetuned version, but also in the base version. Could it be that the Bandi usecase is so confusing that the model loses the ability to produce the eos token?

When analysing all these prompts one really becomes conscious of how these models function and their unique quirks. For us it became really clear that for how new and flashy AI is, training LLMs at its core still is a craft.

Lastly, we wanted to share a particular humorous response of finetuned version of the MPT model. It was responding to a prompt to extract the opening date from a snippet of text from the Parma tender. The response was "*Il bando è in lingua italiana, quindi non è necessario rispondere.<|endoftext|>*". This translates to "The tender is in Italian, so there is no need to respond".

8.2 Finetuning's effectiveness

The most straightforward thing to investigate if finetuning actually has effect is to compare the number of (semi) correct answers of the finetuned models to their base version. For this exact information we refer to figure 8.1. We also included also our Baseline, GPT.

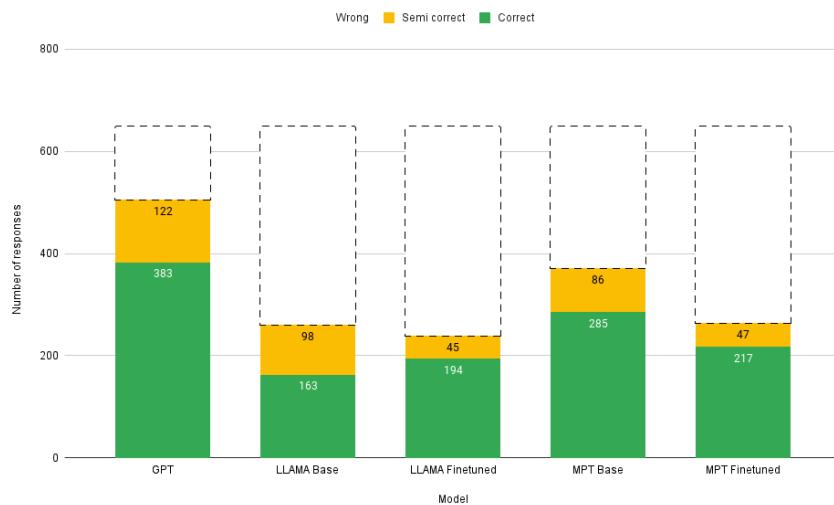


Figure 8.1: The number of correct, semi correct and wrong answers for each model. The total number of questions answered for each model is 648.

We see many things in this graph. Perhaps most importantly we see that GPT outperforms all models by far, responding to more than 3 quarters of all prompts (semi) correctly. The only model that perhaps comes close is the MPT Base model.

This then also brings us to our second observation: the MPT Finetuned model has both less correct- and semi-correct responses than the MPT Base version. We find this unexpected and quite frankly disappointing, in the discussion we will explore what might be the cause.

This leads us to another unpleasant surprise: LLaMA 2 Base performs significantly worse than MPT Base. This is noteworthy as LLaMA 2 is pretrained on twice as many tokens (2T) as MPT (1T). Although the LLaMA 2- and the MPT base model losses on the validation sets are 3.42 and 2.06 respectively, the losses of their finetuned versions are 0.75 and 1.20. We will again explore possible causes in the discussion. LLaMA 2 also comes with a context length (4096) twice as big as that of MPT (2048), but for our testing this is not relevant as our prompts remain smaller than 2048 tokens.

We do see that finetuning LLaMA 2 has had a positive effect, as the finetuned model sees an increased amount of correct responses of 19% at the cost of less semi-correct responses w.r.t. to its base model.

Then we would also like to investigate how often these models actually reply in the requested language. This data can be found in figure 8.2. The figure shows for each model how many times the response of the model is actually in the language that is requested. For each model we ask the model to answer in English 324 times and to answer in Italian 324 times as well. So from this figure we can conclude for example that GPT always responds in Italian when asked to respond in Italian, and that LLaMA always responds in English when asked to respond in English.

In figure 8.2 we again see that GPT outperforms all models, with LLaMA 2 finetuned perhaps being the runner up.

What is maybe the most interesting to see is that for both finetuning cases, LLaMA 2 and MPT, the finetuned models respond more often in Italian when the Italian language is required and less often in English when English is required w.r.t. to their base versions. We think this is proof that models do learn Italian in the finetuning, but more on that in the discussion.

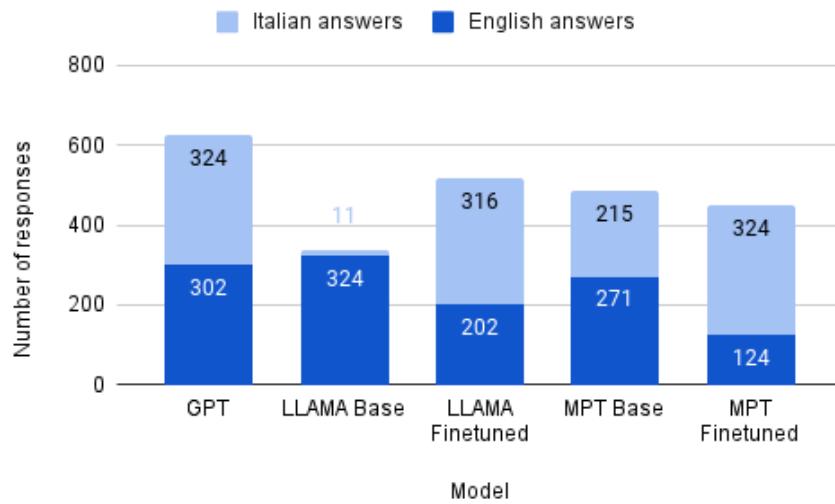


Figure 8.2: We show for each model the number of times it responds in the requested language, separated by the requested language itself. The total number of questions answered for each model is 648.

Then lastly we want to note something very peculiar, we see that the LLaMA 2 Base almost never responds in Italian when requested, while MPT does. This is peculiar because as elaborated on in section 4, we presume that both models have seen some Italian in their pretraining. We will come back to this in the discussion.

8.3 Prompt language and response quality

In order to see how the language performs on English prompts w.r.t. Italian prompts we present figure 8.3. Here we show the number of correct answers for each model separated by the language of the prompt.

What we see is that each model performs slightly better on English prompts than on Italian prompts. That is, the number of correct answers on English prompts is higher than the number of correct answers on the Italian prompts, for each model. This is to be expected as all of these model have primarily

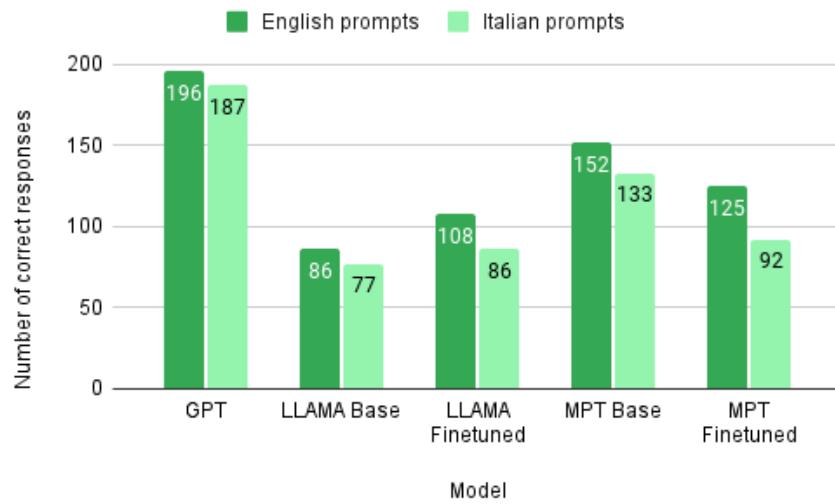


Figure 8.3: We show for each model the number of correct answers separated by the prompt language. The total number of questions answered for each language/model combination is 324.

been trained on the English language.

8.4 Answer language and response quality

In order to see if the model performs better answers in Italian or in English, we provide figure 8.4. This figure shows us the number of correct answers per model separated by the requested language.

The fact that we ask to model to reply in a specific language, does not guarantee that the model actually answers in that language. This separation is made by the language in which we ask the model to answer.

We define the influence of the requested language (the language in which ask the model to answer) by the difference in the number of correct answers between the requested languages.

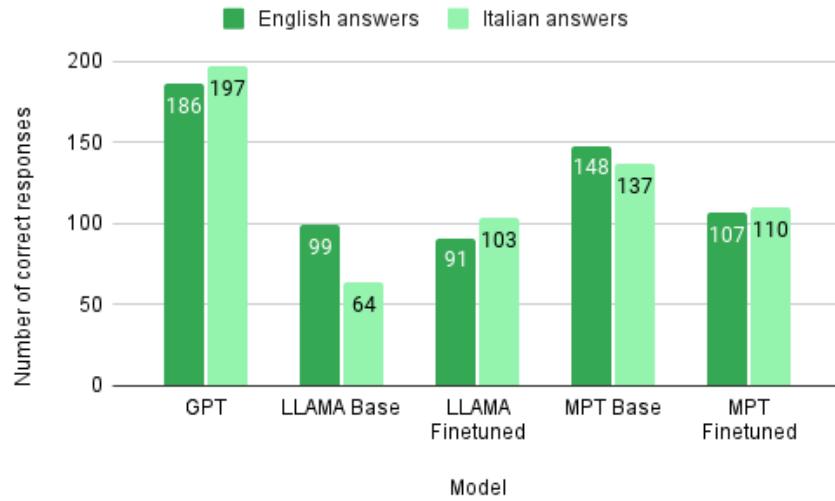


Figure 8.4: We show for each model the number of correct answers separated by the language in which we ask the model to respond. The total number of questions answered for each language/model combination is 324.

We see that this influence is minimal for every model except the LLaMA Base, which sees a 35% decline in number of correct answer when requested in Italian w.r.t. English.

The influence is also not the same between every model, for GPT, LLaMA Finetuned and MPT Finetuned the difference is positive. Meaning that the number of correct answers with the requested language Italian is higher w.r.t. the same model with the requested language English. This influence is negative for LLaMA Base and MPT Base. This could indicate that finetuning does enhance Italian language understanding, especially for LLaMA, but take note that the total number of correct answers for MPT has gone down after fine-tuning.

8.5 Instruction location and response quality

In order to judge how different prompt locations influence the quality of a models response, we provide figure 8.5. In this graph for every model we provide for each prompt location the number correct answers.

We see that what makes a prompt location good is model specific. We also see that the amount of influence that the prompt locations has, is different for each model. Here with influence we mean the difference in the number of correct answers between different prompt locations. GPT and MPT Finetuned work best when the prompt comes both before and after the context, while MPT Base, LLaMA Base and LLaMA Finetuned work best when the prompt is only inserted before the context.

Interesting to note is that with respect to the other models MPT Finetuned seems to be influenced little by the prompt location. Then we note as well that no model works best with the prompt being inserted after the context only.

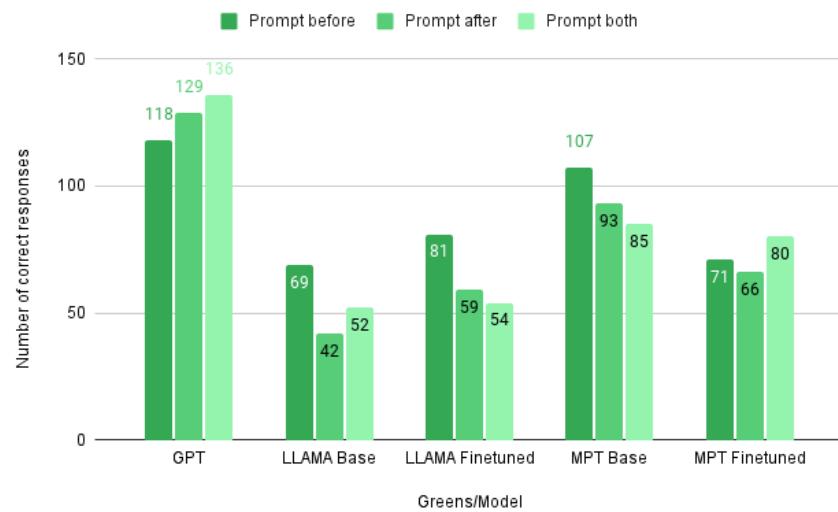


Figure 8.5: We show for each model the number of correct answers separated by the prompt location. The total number of questions answered for each location/model combination is 216.

8.6 Instruction formulation and response quality

In order to judge how different prompt formulations influence the quality of a models response, we provide figure 8.6. In these graphs we provide per question topic the number of correct responses for each model separated by prompt formulation.

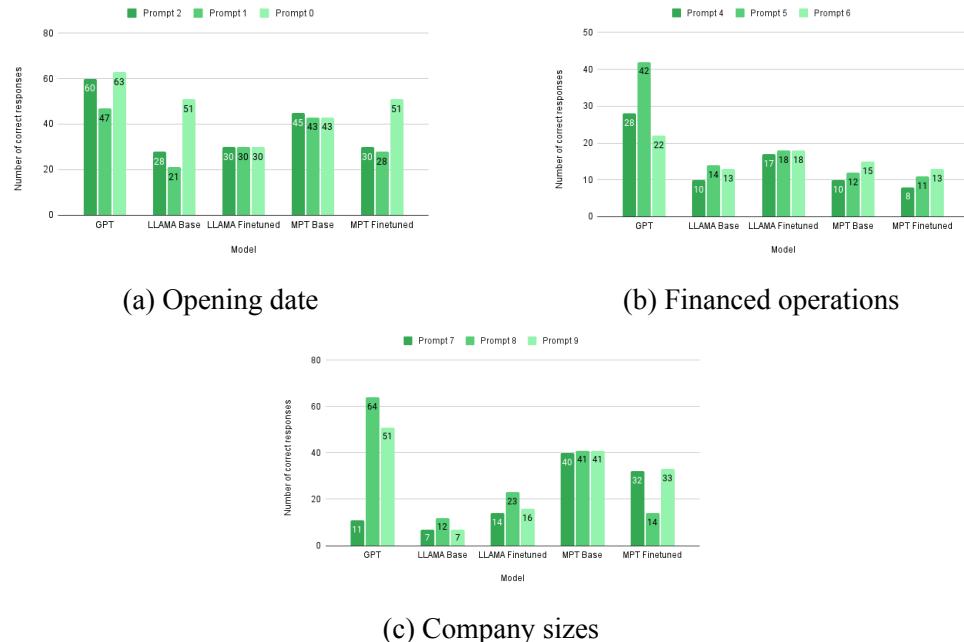


Figure 8.6: The number of correct responses for different prompt formulations, shown for each topic. The total number of questions asked for each prompt/model combination is 72

In this figure we can see that how much influence a prompt formulation has on the response quality is highly topic- and model specific.

We define the influence of the prompt formulation as the difference in the number of correct answers between different prompts. For example for the 'company sizes' topic we see that the difference of the correct number of

responses for the GPT model between prompt 7 (11 correct responses) and prompt 8 (64 correct responses) is very big. In this case we say that the influence of the prompt formulation is big. On the other hand if we look at the MPT Base model in the 'opening date' topic, we see that the difference of the number of correct responses between prompts 1 (45), prompt 2 (43), and prompt 3 (43) is minimal. Hence in this case we also say that the prompt formulation has minimal influence.

We should add a comment to this section however, as said before we use prompts derived from a GPT optimized formulation. These are, in figure 8.6a prompt 1, in figure 8.6b prompt 2 and in 8.6c prompt 2. These prompts somewhat skew the results, but it can be seen that even without these prompts that GPT still outperforms the other models overall.

Then, we see that for the 'opening date' topic, the prompt formulation has a big influence on the GPT, LLaMA Base and the MPT finetuned model, that for the 'financed operations' topic the prompt formulation has a big influence on the GPT model, and that for the 'company sizes' topic the prompt formulation has big influence on the GPT and the MPT Finetuned model.

We also see that for the 'opening date' topic, the prompt formulation has little influence on the LLaMA Finetuned and the MPT Base model, that for the 'financed operations' topic the prompt formulation has a little influence on the LLaMA Finetuned model, and that for the 'company sizes' topic the prompt formulation has big influence on the MPT Base model.

Furthermore, we have seen that for every model there is at least one topic for which different prompt formulations make a difference. Thus prompt engineering seems to be worthwhile.

Concluding, we see that prompt formulation has a big influence on especially the performance of GPT. Our models are not immune to this influence either but some of them, i.e. LLaMA finetuned and MPT Base, do seem more resistant to this influence.

8.7 Correct answers per tender/topic pair

In order to gain insight in whether or not there are any particular difficult tenders or topics we present figure 8.7. This figure shows for each model how many correct answers that model gives for said combinations. This takes away the influence of the prompt formulation and allows us to identify difficult test cases. The maximum number of correct answers for each tender and topic pair is 36.

Looking at these figures there is not something that particularly stands out. We see that generally the topic of opening dates does well. Some models have difficulties with a particular tender, but these cases are covered by another model that does do well on that specific tender/topic combination. Albeit that this 'saviour' is often, but not always, GPT.

The above does come with one exception, there is one tender and topic pair that does prove to be difficult for all models to answer: the financed operations of Marche (interv_finan - Marche). There are models that do answer correctly to this pair, but it does not occur too often.

Lastly we note that the distributions of LLaMA 2 and MPT did change after having been finetuned, but only by a little.

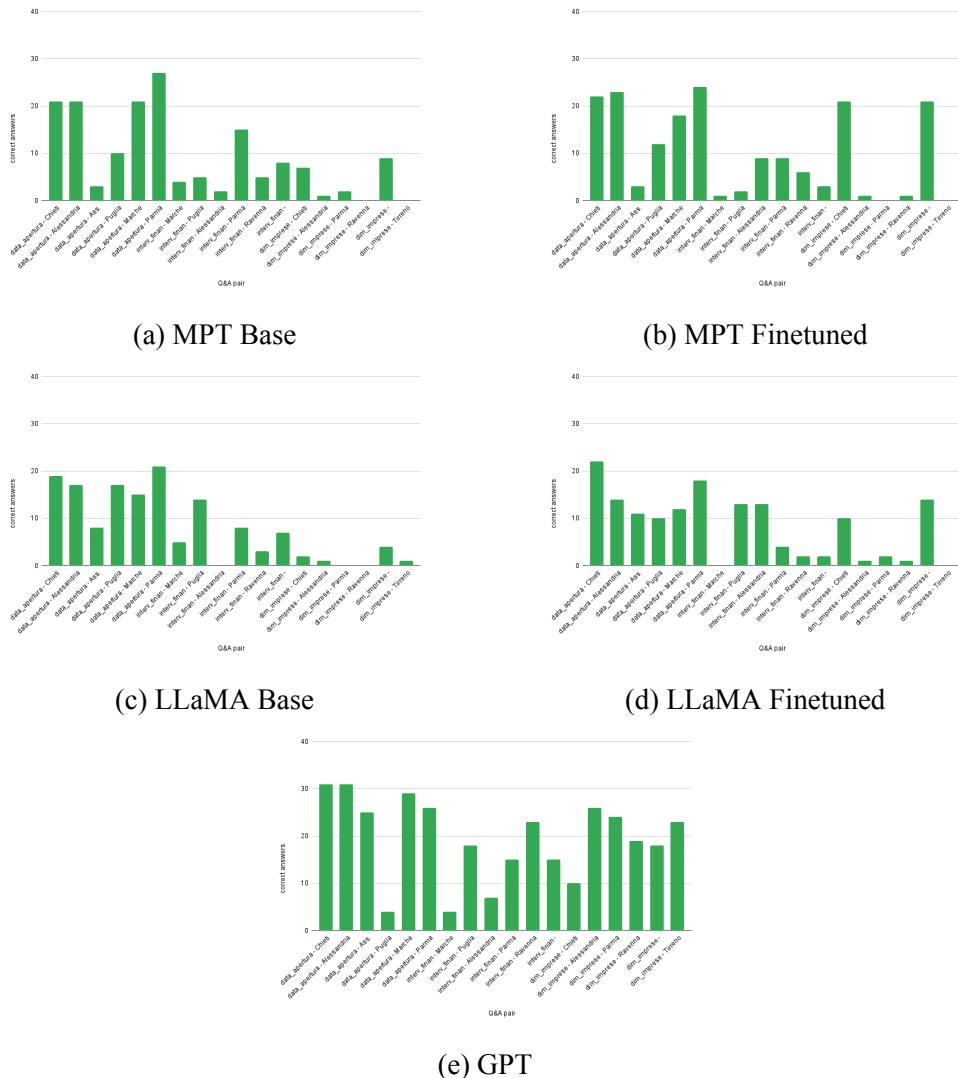


Figure 8.7: The number of correct responses for different instruction/bandi (input) pairs

8.8 Patterns

Our final inquiry is whether any patterns can be observed across the various topics and models. As discussed in the previous sections, some general trends emerge:

- First of all we have seen that all models seem to function better, that is give more correct answers when the prompt is given in English. This is logical as all models are primarily trained on the English language
- It is model and topic specific how much the prompt's location influences the quality of the model's response and which prompt location is best, but putting the question after the context only is the worst option in every case.
- We have seen that which prompts function well on a specific topic is model specific. We have seen that on some topics different prompts make no difference in the quality of the responses of a model. However we have seen that for every model there is at least one topic for which different prompt formulations make a difference. Hence we recommend to always include prompt engineering in development pipelines.

In conclusion, our systematic observations provide a foundation for understanding the behavior of language models and their performance characteristics. These insights can be invaluable to researchers and practitioners working with large language models.

Chapter 9

Discussion

In this section we would like to discuss the 5 questions that came up in the results section: How come that our two models suffer from overfitting? How come our MPT finetuned model performs worse than our MPT Base model? How come that MPT Base outperforms LLAMA 2 Base? After finetuning both models respond more often in Italian, is this proof of learning? LLaMA 2 Base never responds in Italian while MPT Base does, how come? We will find that the answer often comes back as lacking understanding of the Italian language.

9.1 Base model comparison

We have seen that LLaMA 2 Base performed worse than MPT Base. We certainly find this unexpected as LLaMA has been much more extensively trained. Also on the website of LLaMA 2 it is shown how it outperforms MPT on all but one benchmark. Here we will discuss if this should also be considered strange or odd.

Arguments to consider this strange are twofold. First we have seen that the LLaMA 2 Finetuned model achieved a loss 37% lower than MPT Finetuned. This hints at an aptitude of LLaMA 2 for capability in the Italian language.

Secondly we have the fact that LLaMA 2 has been pretrained on much more data, on twice as many tokens as MPT. One would expect more extensively trained model to be the more proficient one, also if the scope of their applications (a different language) fall partly outside the training data.

Arguments to consider this within the norms of the expected are also twofold. First we have seen that before training MPT scored a much lower loss on the validation set than LLaMA 2, having losses of 2.0552 and 3.4196 respectively. Indicating clearly a higher base capability of MPT. Secondly we have seen in our testing that the LLaMA 2 Base almost never responds in Italian when requested, while MPT Base does. This is somewhat peculiar as we presumed that both model have seen some Italian in their pretraining.

We conclude that this observation is within the norms of the expected. The reasoning being that, at least in proportion, LLaMA 2 must have seen less Italian in its pretraining than MPT, hence resulting in worse base performance on the Italian language and consequently also on the Bandi usecase.

9.2 Models learning

In testing we have seen for the finetuned versions of both LLaMA 2 and MPT that they respond more often in Italian when the Italian language is requested and less often in English when English is requested w.r.t. to their base versions. In other words, finetuned models respond more often in Italian, both when requested and when not requested. In some case it seems even to go as far as that finetuning changes the default language in which the model answers to Italian. Presented with this fact there is no doubt in our mind that during this finetuning that learning is happening, the question remains whether or not the model learns something worthwhile. This ties back in with our next question.

9.3 Overfitting

We have seen in our finetuning graphs, figures 7.1a, 7.1b, 7.2b and 7.2c especially, that we are suffering quite a bit from overfitting. We want to know why this is the case. Overfitting has three generally accepted causes: A too complex/potent model, noisy and/or incorrect data, or an insufficient amount of data. We will go over each of these motives and analyse which one might cause our overfitting.

9.3.1 Too potent/complex model

This is the first cause we looked at during the finetuning of the MPT model. We reached the point in which we were training with the minimal LoRA dimension ($r = 1$), but we were still suffering from overfitting. Dummung down the model further was still possible by removing modules from the LoRA injections, but looking at the loss curves we presumed that doing so would impare the capacity of the model to learn from the dataset.

For LLaMA 2 we had the same problem, but we arrived from the opposite side. We had a model with a relatively horizontal- and clean curve. Then we rendered our model more complex/potent and the same overfitting as seen when training MPT reappeared.

We therefore conclude that this finetuning showing overfitting is not the cause of adding too many parameters using LoRA.

9.3.2 Insufficient amount of training data

Whether or not the amount of training data is sufficient is difficult to make sustained claims about. We have seen that various instruction datasets are about the same order of magnitude in size as Stambecco. So perhaps for a pretrained model to learn to respond to the question/answer formula is enough.

But if a pretrained model also has to gain Italian language understanding it is imaginable that this dataset is too small. Especially if one compares the size of Stambecco to that of the size of the pretrain data: 50 thousand question/answer pairs versus 2 trillion tokens.

9.3.3 A noisy/inaccurate dataset

We have not tried to hide the fact that we are not too fond of the quality of the dataset. In section 4.2 we have illustrated diverse bad- but also good entries. Although we do think the quality of Stambecco contributes to the subpar performance of our models, we think it is too shortsighted to put the blame (entirely) here.

9.3.4 Summary

In the above we have been focusing on the finetuning, which is done with LoRA. But one should not forget that what a LoRA model returns, always has the base model as a base to its answer: $\text{Answer}(\text{input}) = \text{Base_Model}(\text{input}) + \text{LoRA}(\text{input})$. At this point we ask ourselves the question, how good is the base model?

In order to answer this question we refer to our experiment in section 7.2 in which we test our models on both the entire Alpaca and Stambecco dataset. The difference in losses between these datasets tell us something about these model's capabilities between these languages. In table 9.1 we have gathered all our loss data from different models, base- and finetuned versions, on Alpaca and Stambecco.

The first thing we note regarding our testing of LLaMA 2 on Stambecco is that the loss on the test set is much bigger than the loss of the entire training set. For MPT this is not the case.

Test set/ model	Stambecco	Alpaca
MPT Base	Entire set: 2.048 Test set: 2.055	Entire set: 1.688
MPT Finetuned on Stambecco	Train set: 1.015 Test set: 1.212	N.A.
MPT Finetuned on Alpaca	N.A.	?
LLaMA 2 Base	Entire set: 2.602 Test set: 3.420	Entire set: 1.822
LLaMA 2 Finetuned on Stambecco	Train set: 0.617 Test set: 0.768	N.A.
LLaMA 2 Finetuned on Alpaca	N.A.	?

Table 9.1: Loss of various models on the Stambecco- and Alpaca dataset.

Then as already stated in section 7.2, both base models score a lot better on the Alpaca dataset than on the Stambecco dataset. This indicates a much greater aptitude of both models for the English language w.r.t. the Italian language. Seeing this result we then also expect that if we were to finetune our models on Alpaca, that these models would outperform those finetuned on Stambecco. Simply because these models have seen a lot more English in their pretraining in comparison.

We then also arrive at our conclusion: both MPT and LLaMA 2 have not seen enough Italian in their pretraining to perform well on the Italian language. We do not think that finetuning with LoRA on Stambecco can resolve this: all our models suffer from overfitting sooner or later. The quality of the Stambecco dataset does not help, but it is certainly not the primary cause of the subpar performance. In our opinion the solution lies with doing more pretraining on the Italian language. The pretraining of both models has after all been focused on the English language. This conclusion is also in line with [35] that strongly suggest that almost all knowledge in LLMs is learned during pretraining.

9.4 Results finetuning

This then also brings us to our final observation: the MPT Finetuned model has both less correct- and semi-correct responses than the MPT Base version. We find this unexpected and quite frankly disappointing. It is difficult to say what could have been the cause of this problem, especially given the fact that we do not have this problem with LLaMA 2. Also MPT having smooth loss curves does not explain this, the contrary actually. For an answer we look to our previous conclusion, that the model is not pretrained sufficiently well on the Italian language. Although finetuning does improve the loss on the Stambecco dataset, apparently this leads the model to unlearn certain capabilities that helped with our Bandi usecase.

Chapter 10

Conclusion

In this thesis, our goal was to develop a commercially licensed LLM tailored for application on PwC’s Italian language-based use-cases, with performance comparable to ChatGPT and other service providers. While we did succeed in training such models, regrettably, we did not achieve the performance we had aspired to.

Our findings suggest that fine-tuning an LLM on a language with limited representation in the training set does not adequately equip the model to operate effectively in that language. As a result, we advocate for the necessity of a commercially licensed LLM that is pre-trained on the Italian language, or any other new language for that matter.

The decision to initiate this fine-tuning process was both logical and cost-effective. Training an entirely new language model requires a substantial financial investment, whereas our fine-tuning experiments were relatively low-cost, yet offered the potential for significant returns. Therefore, these preliminary experiments served as a prudent and necessary first step in our exploration.

Through these experiments, we derived several sustained conclusions that we

believe hold value for the broader AI community, particularly in the context of extractive question answering:

- When a model must respond to context-based questions in a new language, it tends to deliver superior results when the instruction is in its primary pretraining language rather than the new language.
- The influence of instruction location relative to the input on performance varies based on the model and topic, with no consistent best practice for instruction placement. However, placing the instruction after the input consistently yields the least favorable results.
- Prompt engineering, may not consistently impact the quality of model responses. Its effectiveness varies depending on the model and topic, but when it does have an impact, it is often substantial. Therefore, we endorse the inclusion of prompt engineering in development pipelines.

In summary, we have taken the initial steps towards building LLMs suitable for the Italian language. In doing so, we conducted an exploratory analysis of LLM performance in Italian, made novel discoveries, provided evidence for previously unproven best practices, and underscored the need for commercially licensed LLMs pre-trained on specific languages. This is particularly significant in the context of the predominantly English-focused NLP landscape and even more so within the open-source NLP domain.

While we acknowledge that our results did not meet our high expectations, these findings contribute to the growing body of knowledge in the field of natural language processing and guide the way toward more effective language model development. Our work serves as a foundation for future research and the continued pursuit of high-performance LLMs for diverse languages and applications.

Chapter 11

Future work

We conclude our paper by outlining potential future work. These encompass experiments we were unable to undertake due to time and resource constraints, ideas that can contribute to PwC’s business strategy, and new projects that could benefit Italian speakers in general.

- **Creation of a Language Model Pretrained on Italian**

We believe that the development of an open-source, commercially licensed Language Model pretrained primarily on the Italian language could greatly benefit the Italian-speaking community. Such a model is expected to surpass the currently available open-source models and may even compete with GPT. We strongly recommend the creation of such a Language Model.

- **Finetuning MPT and LLaMA 2 on the Alpaca Dataset**

As elaborated in our paper, we intended to finetune MPT and LLaMA 2 on the Alpaca dataset to compare these models’ capabilities on different languages. Regrettably, we were unable to conduct this experiment. While we do not anticipate extraordinary results, performing such a control experiment is recommended for thorough analysis.

- **Development of a High-Quality Instruction Dataset in Italian**

Presently, Stambecco is the only instruction dataset in Italian and, as

indicated in our thesis, we think there is room for improvement. We would then also encourage the creation of a high quality variant of such a dataset.

- **Finetuning on Specific Use Cases**

In pursuit of building Language Models that function well on PwC’s use cases, direct finetuning on these specific use cases could be worthwhile. However, this necessitates the creation of datasets and training tailored to each use case. Although it is a labor-intensive effort, it has the potential to yield promising results and can also serve as a basis for further pre-/finetuning an Italian Language Model.

- **Finetuning on Larger Models**

As detailed in Section 4, the models we have finetuned have larger counterparts with more parameters, which perform better across all benchmarks. Replicating our finetuning experiments on these larger models is an idea worth exploring. Nevertheless, we suspect that even these models may not have seen enough Italian in their pretraining, resulting in comparable performance.

In conclusion, artificial intelligence is a rapidly evolving field, and we have much to learn, improve, and build, referring to us as authors, but also referring to the scientific community as a whole.

Bibliography

- [1] URL: <https://stats.stackexchange.com/questions/421935/what-exactly-are-keys-queries-and-values-in-attention-mechanisms>.
- [2] URL: <https://huggingface.co/mosaicml/mpt-7b-instruct>.
- [3] URL: <https://ai.meta.com/llama/>.
- [4] URL: <https://github.com/facebookresearch/llama/blob/main/FAQ.md>.
- [5] URL: <https://github.com/gururise/AlpacaDataCleaned>.
- [6] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016. arXiv: 1607.06450 [stat.ML].
- [7] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. El-Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, and J. Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. arXiv: 2204.05862 [cs.CL].
- [8] Y. Bulatov and T. Salimans. Fitting larger networks into memory. URL: <https://medium.com/tensorflow/fitting-larger-networks-into-memory-583e3c758ff9>.

- [9] M. Conover, M. Hayes, A. Mathur, J. Xie, J. Wan, S. Shah, A. Ghodsi, P. Wendell, M. Zaharia, and R. Xin. Free dolly: introducing the world’s first truly open instruction-tuned llm. 2023. URL: <https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-l1m> (visited on 06/30/2023).
- [10] Efficient training. URL: https://huggingface.co/docs/transformers/v4.23.1/en/perf%7B%5C_%7Dtrain%7B%5C_%7Dgpu%7B%5C_%7Done.
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. arXiv: 1512.03385 [cs.CV].
- [12] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: low-rank adaptation of large language models, 2021. arXiv: 2106.09685 [cs.CL].
- [13] A. Karpathy. Let’s build gpt: from scratch, in code, spelled out. URL: <https://www.youtube.com/watch?v=kCc8FmEb1nY>.
- [14] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. Generating wikipedia by summarizing long sequences, 2018. arXiv: 1801.10198 [cs.CL].
- [15] Mchl-Labs. Mchl-labs/stambecco: the home of stambecco ☺: italian instruction-following llama model. URL: <https://github.com/mchl-labs/stambecco>.
- [16] B. Peng, C. Li, P. He, M. Galley, and J. Gao. Instruction tuning with gpt-4, 2023. arXiv: 2304.03277 [cs.CL].
- [17] O. Press, N. A. Smith, and M. Lewis. Train short, test long: attention with linear biases enables input length extrapolation, 2022. arXiv: 2108.12409 [cs.CL].
- [18] PwC. Bandi e incentivi. URL: <https://bandieincentivi.pwc-tls.it/>.

- [19] A. Radford and K. Narasimhan. Improving language understanding by generative pre-training. In 2018.
- [20] A. Santilli and E. Rodolà. Camoscio: an italian instruction-tuned llama, 2023. arXiv: 2307.16456 [cs.CL].
- [21] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: an instruction-following llama model. [https://github.com/tatsu-lab/stanford{ }alpaca](https://github.com/tatsu-lab/stanford_alpaca), 2023.
- [22] Y. Tay, M. Dehghani, V. Q. Tran, X. Garcia, J. Wei, X. Wang, H. W. Chung, S. Shakeri, D. Bahri, T. Schuster, H. S. Zheng, D. Zhou, N. Houlsby, and D. Metzler. UI2: unifying language learning paradigms, 2023. arXiv: 2205.05131 [cs.CL].
- [23] M. N. Team. Introducing mpt-7b: a new standard for open-source, commercially usable llms, May 2023. URL: <https://www.mosaicml.com/blog/mpt-7b>.
- [24] Tender: alessandria. URL: <https://www.ra.camcom.gov.it/attivita-promozionali/contributi/contributi-cciaa-ravenna/contributi-rivolti-alle-imprese/bandi-anno-2023/Contributi%7B%5C%7D20bando%7B%5C%7D20alluvione%7B%5C%7D202023>.
- [25] Tender: chieti pescara. URL: <https://www.chpe.camcom.it/pagina704%7B%5C%7Davviso-per-contributo-a-sostegno-del-collegamento-marittimo-pescara-croazia.html>.
- [26] Tender: marche. URL: <https://www.regione.marche.it/Regione-Utile/Agricoltura-Sviluppo-Rurale-e-Pesca/Bandi-di-finanziamento/id%7B%5C%7D8293/7058>.
- [27] Tender: maremma e tirreno. URL: <https://www.lg.camcom.it/bandi/bando-innovazione-digitale-40>.

- [28] Tender: parma. URL: <https://www.pr.camcom.it/news-eventi/Bpk>.
- [29] Tender: puglia. URL: https://burp.regione.puglia.it/web/guest/bollettini?p%7B%5C_%7Dp%7B%5C_%7Did=it%7B%5C_%7Dindra%7B%5C_%7Dregione%7B%5C_%7Dpuglia%7B%5C_%7Dburp%7B%5C_%7Dweb%7B%5C_%7DSearchPortlet&p%7B%5C_%7Dp%7B%5C_%7Dlifecycle=0&p%7B%5C_%7Dp%7B%5C_%7Dstate=normal.
- [30] Tender: ravenna. URL: <https://www.ra.camcom.gov.it/attivita-promozionali/contributi/contributi-cciaa-ravenna/contributi-rivolti-alle-imprese/bandi-anno-2023/Contributi%7B%5C%7D20bando%7B%5C%%7D20alluvione%7B%5C%%7D202023>.
- [31] Tender: veneto - enogastronomiche. URL: <https://bandi.regione.veneto.it/Public/Download?idAllegato=25696>.
- [32] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: open and efficient foundation language models, 2023. arXiv: 2302.13971 [cs.CL].
- [33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017. arXiv: 1706.03762 [cs.CL].
- [34] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-instruct: aligning language models with self-generated instructions, 2023. arXiv: 2212.10560 [cs.CL].
- [35] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer, and O. Levy. Lima: less is more for alignment, 2023. arXiv: 2305.11206 [cs.CL].

Appendix A

Tables of finetuning losses

Epoch	Train Loss	Val Loss
1	1.2102	1.209008694
2	1.1393	1.176738739
3	1.0951	1.165756583
4	1.0317	1.165969729
5	0.9951	1.173580408
6	0.9545	1.180489421
7	0.9186	1.189377546
8	0.8955	1.208033204
9	0.8616	1.221395493
10	0.8347	1.240166068
11	0.8148	1.25161159
12	0.7943	1.266140103
13	0.7682	1.281847715
14	0.7547	1.300261617
15	0.7376	1.310849309

Table A.1: The table of the training- and validation losses corresponding to the first MPT trial, figure 7.1a.

Epoch	Train Loss	Val Loss
1	1.2239	1.223728776
2	1.1688	1.195912719
3	1.1391	1.186287999
4	1.0909	1.181571603
5	1.0684	1.182962537
6	1.042	1.183759451
7	1.0195	1.186593294
8	1.0092	1.195128202
9	0.9871	1.19621563
10	0.9701	1.202170968
11	0.9624	1.20488596
12	0.9507	1.213654399
13	0.9303	1.217672586
14	0.9232	1.224497676
15	0.9118	1.228129387

Table A.2: The table of the training- and validation losses corresponding to the second MPT trial, figure 7.1b.

Epoch	Train Loss	Val Loss
1	1.239	1.238278389
2	1.1952	1.214365363
3	1.1762	1.208330274
4	1.1366	1.201288223
5	1.1216	1.200346112
6	1.1031	1.200184226
7	1.0856	1.198607326
8	1.084	1.19994092
9	1.0665	1.200973392
10	1.0533	1.202574372
11	1.051	1.203165054
12	1.0431	1.20519495
13	1.0252	1.206318259
14	1.0226	1.209497333
15	1.0149	1.211533308

Table A.3: The table of the training- and validation losses corresponding to the third MPT trial, figure 7.1c.

Epoch	Train Loss	Val Loss
1	0.7694	0.7916440368
2	0.7562	0.7881999612
3	0.7487	0.7799901962
4	0.7297	0.7799584866
5	0.7245	0.7784932852
6	0.7172	0.777428329
7	0.7063	0.7768800259
8	0.7064	0.7763211131
9	0.7057	0.7714284658
10	0.6981	0.769372642
11	0.6993	0.7678364515
12	0.7003	0.7712532878
13	0.6937	0.7726595998
14	0.6923	0.7714828849
15	0.6945	0.7713136673

Table A.4: The table of the training- and validation losses corresponding to the first LLaMA trial, figure 7.2a.

Epoch	Train Loss	Val Loss
1	0.7662	0.7815913558
2	0.7486	0.772269845
3	0.7377	0.7660132051
4	0.7143	0.7583094835
5	0.7063	0.7612097263
6	0.6965	0.7680630088
7	0.6833	0.7530135512
8	0.6818	0.7678575516
9	0.6782	0.7607021332
10	0.6691	0.7647323012
11	0.6687	0.7683621049
12	0.6692	0.7697759867
13	0.6603	0.7702064514
14	0.6577	0.7680479288
15	0.6586	0.7687986493
16	0.6564	0.7713423967
17	0.6454	0.7741293311
18	0.6476	0.7697276473
19	0.6436	0.7679331303
20	0.6473	0.7728589773
21	0.6398	0.7761081457
22	0.6399	0.7776685357
23	0.6373	0.7768996358
24	0.6409	0.7766670585
25	0.6309	0.7776957154

Table A.5: The table of the training- and validation losses corresponding to the second LLaMA trial, figure 7.2b.

Epoch	Train Loss	Val Loss
1	0.7628	0.7847151756
2	0.742	0.7636311054
3	0.7273	0.7571750283
4	0.7003	0.7562611699
5	0.6884	0.7503700852
6	0.6748	0.7493066788
7	0.6584	0.7612172365
8	0.6533	0.7583049536
9	0.647	0.7661053538
10	0.6352	0.7607308626
11	0.6329	0.7607308626
12	0.6312	0.7673183084
13	0.621	0.7660494447
14	0.6173	0.7646733522
15	0.6169	0.7680584788

Table A.6: The table of the training- and validation losses corresponding to the third LLaMA trial, figure 7.2c.

Appendix B

Remaining prompts used for testing

- Prompt 3 - Italian: Sei un assistente AI. Ti forniremo un testo estratto da un bando di gara. Un bando di gara è un documento fornito dal governo che specifica chi può presentare domanda per usufruire di un determinato fondo pubblico, a cosa e a quali condizioni. È tuo compito estrarre da tale contesto il momento di apertura del bando (la data di apertura), cioè la data a partire dalla quale enti, persone e organizzazioni, possono presentare domanda per usufruire dei benefici del bando. Tieni presente che nel contesto potrebbero essere fornite più date. In quel caso è fondamentale riportare solo la data di apertura del bando. La data di apertura dell'annuncio è spesso preceduta da stringhe del tipo: 'dal giorno' oppure 'dalle ore del giorno'. La data di apertura del bando non è mai preceduta da stringhe del tipo: 'entro il giorno' o 'fino al'. Se nel contesto non è presente una data di apertura, una cosa molto plausibile, si può rispondere con 'ND'.
- Prompt 3 - English: You are an AI assistant. We will provide you with a piece of text extracted from a tender document. A tender document is a document provided by the government specifying who can apply to make use of a certain public fund, what for and at which conditions. It is your job to extract from that context when the tender opens (the opening date), i.e. the date from which entities, people and organizations, can apply to make use of the tender's benefits. Note that in the context multiple dates could be provided. In that case it is crucial that you only report the date in which the tender opens. The opening date of the announcement is often preceded by strings such as: 'dal giorno' or 'dalle ore del giorno'. The opening date of the call is never preceded by strings such as: 'entro il giorno' or 'fino al'. Should no opening date be present in the context, you can respond with 'ND'.
- Prompt 4 - Italian: Sei un assistente AI che aiuta a estrarre i dati dai bandi: documenti forniti dal governo che spiegano chi può ricevere finanziamenti per cosa e a quali condizioni. Dal contesto che andiamo a fornirti, vorremmo sapere quali sono le spese coperte da quello specifico bando. Vorremmo che i risultati fossero presentati in un elenco. Se non ci sono spese coperte dal bando puoi rispondere 'ND'.
- Prompt 4 - English: You are an AI assistant that helps extract data from tenders: documents provided by the government explaining who can receive funding for what and under which conditions. From the context we are going to provide to you, we would like to know which are the expenses covered by that specific tender. We would like to have your results presented in a list. If there are no expenses covered by the tender you can reply 'ND'.
- Prompt 5 - Italian (GPT formulation, original): Immagina di essere un estrattore di metadati da bandi di gara. Estrai le spese ammissibili finanziate dal bando, se presenti nel contesto. Per spese Ammissibili si

intendono le spese che si possono sostenere grazie ai finanziamenti forniti dal bando. Nel contesto potresti trovare anche cifre o percentuali che non devono essere inserite all'interno dell'output. Fornisci come tua risposta una lista di stringhe aventi come valore le spese ammissibili estratte dal contesto fornito, se presenti. Un esempio: [‘spese di pubblicità e promozione’, ‘spese per la gestione di spazi’, ‘costi per il personale e compensi professionali’]. Se nel contesto fornito non è presente alcuna spesa ammissibile puoi rispondere ‘ND’.

- Prompt 5 - English (GPT formulation, translated): Imagine you are a tender metadata extractor. Extract the eligible expenses financed by the tender, if present in the context. By Eligible expenses we mean the expenses that can be supported due to the funding provided by the tender. In the context you may also find figures or percentages that should not be inserted into the output. Provide as your response a list of strings having as their value the eligible expenses extracted from the context provided, if any. An example: [‘spese di pubblicità e promozione’, ‘spese per la gestione di spazi’, ‘costi per il personale e compensi professionali’]. If there are no eligible expenses in the context provided you can reply ‘ND’.
- Prompt 6 - Italian: Sei un assistente AI che aiuta a estrarre dati da frammenti di testo. Riceverai frammenti di testo dei documenti di bandi italiani. I bandi sono documenti che descrivono come, chi e a quali condizioni si può accedere ai fondi pubblici. Descrivono in modo molto specifico per quali attività/spese e obiettivi i fondi possono essere utilizzati. È qui che entri in gioco tu. Vorremmo che tu indichi dal frammento di testo che forniamo quali attività/spese sono sovvenzionate dal rispettivo documento di gara. Se non riesci a trovare alcuna attività/spesa puoi rispondere ‘ND’.
- Prompt 6 - English: You are an AI assistant helping extract data from text snippets. You will receive snippets of text of Italian tender documents. Tenders are documents describing how, who and under what conditions one can access public funds. They describe very specifically for which activities/expenses and goals the funds can be used. This is where you come in. We would like you to tell us from the text snippet we provide which activities/expenses are subsidized by the respective tender document. If you cannot find any activities/expenses you can reply ‘ND’.
- Prompt 7 - Italian: Immagina di essere un estrattore di metadati dei bandi. Estrai dal contesto fornito le dimensioni che un’azienda deve avere per poter beneficiare dei fondi del bando. I valori validi per la dimensione aziendale sono i seguenti: ‘Micro impresa’, ‘Piccola impresa’, ‘Media impresa’, ‘Mid Cap’, ‘Grande impresa’. Se nel contesto fornito trovi uno o più valori di ‘dimensione’, forniscili tutti nella tua risposta. Un esempio di risposta valida è [‘Microimpresa’, ‘Grande impresa’]. A volte nel contesto fornito troverai l’acronimo ‘PMI’, in questo caso la risposta dovrebbe contenere almeno le seguenti dimensioni: [‘Microimpresa’; ‘Piccola impresa’; ‘Media impresa’]. Se nel contesto fornito non sono presenti i valori descritti sopra, puoi rispondere ‘ND’.
- Prompt 7 - English: Imagine you are a tender metadata extractor. Extract from the context provided the size(s) a company must have in order to be eligible to receive the funds from the tender. Valid values for the company size are the following: ‘Micro impresa’, ‘Piccola impresa’, ‘Media impresa’, ‘Mid Cap’, ‘Grande impresa’. If in the context provided you find one or more ‘size’ values, provide them all in your answer. An example of a valid answer is [‘Micro impresa’, ‘Grande impresa’]. Sometimes in the context provided you will find the acronym ‘PMI’, in this case the answer should contain at least the following sizes: [‘Micro impresa’; ‘Piccola impresa’; ‘Media impresa’]. If there are values as described above in the provided context, you can answer ‘ND’.
- Prompt 8 - Italian (GPT formulation, original): Immagina di essere un estrattore di metadati da bandi di gara. Estrai dal contesto fornito la dimensione dell’impresa richiedente, se presente. L’impresa può avere una o più di una tra le seguenti dimensioni: ‘Micro impresa’, ‘Piccola impresa’, ‘Media impresa’, ‘Mid Cap’, ‘Grande impresa’. Se la dimensione o le dimensioni dell’impresa sono esplicitamente presenti nel contesto fornito inserisci in una lista di stringhe aventi come valori la dimensione o le dimensioni dell’impresa. Talvolta nel contesto fornito è indicata la dimensione dell’impresa richiedente con la sigla ‘PMI’, in tal caso inserisci in la seguente lista [‘Micro impresa’; ‘Piccola impresa’; ‘Media impresa’]. Se non è presente alcun valore rappresentante la dimensione dell’impresa nel contesto fornito inserisci nella stringa di output il valore ‘ND’.

- Prompt 8 - English (GPT formulation, translated): Imagine you are a tender metadata extractor. Extract from the context provided the size of the requesting company, if any. The enterprise can only have one or more of the following dimensions: 'Micro impresa', 'Piccola impresa', 'Media impresa', 'Mid Cap', 'Grande impresa'. If the size or dimensions of the company are explicitly present in the context provided, insert in a list of strings having the size or dimensions of the company as values. Sometimes in the context provided the size of the requesting company is indicated with the acronym 'PMI', in this case enter in the following list ['Microimpresa'; 'Piccola impresa'; 'Media impresa']. If there is no value representing the size of the company in the context provided, enter only the value string 'ND'.
- Prompt 9 - Italian: Sei un assistente AI che aiuta a estrarre i dati dai bandi: documenti forniti dal governo che spiegano chi può ricevere finanziamenti per cosa e a quali condizioni. Dal contesto, un frammento di testo tratto da un bando di gara italiano, che ti forniamo, vorremmo sapere quali sono le dimensioni delle aziende idonee a ricevere i finanziamenti del bando. I valori validi sono uno o più dei seguenti: 'Micro business', 'Small business', 'Medium business', 'Mid Cap', 'Large business'. A volte nel contesto fornito troverete l'acronimo 'PMI', in questo caso la risposta dovrebbe contenere almeno le seguenti dimensioni: ['Microimpresa'; 'Piccola impresa'; 'Media impresa']. Se nel contesto fornito sono presenti i valori descritti sopra, puoi rispondere 'ND'.
- Prompt 9 - English: You are an AI assistant that helps extract data from tenders: documents provided by the government explaining who can receive funding for what and under which conditions. From the context, a snippet of text from an Italian tender document, we are going to provide to you, we would like to know what the sizes of the companies eligible to receive funding from the tender are. The valid values are one or more of the following: 'Micro impresa', 'Piccola impresa', 'Media impresa', 'Mid Cap', 'Grande impresa'. Sometimes in the context provided you will find the acronym 'PMI', in this case the answer should contain at least the following sizes: ['Micro impresa'; 'Piccola impresa'; 'Media impresa']. If there are values as described above in the provided context, you can answer 'ND'.

Acknowledgements

I want to thank both parties with whom I entered the internship agreement.

On the side of PwC I want to thank Anna Elisabetta Ziri for the pleasant work environment at the office in Bologna, as well as for granting me the possibility to develop myself in my first professional role.

I want to thank Dario Fioravante Alise for his constant support during the thesis and I want to voice my appreciation for his low level knowledge of everything in and around the project.

On the university's side I want to thank Paolo Torroni and Federico Ruggeri for their involvement and feedback on the thesis.