Scaling Laws of Decoder-Only Models on the Multilingual Machine Translation Task

Gaëtan Caillaut and **Raheel Qader** and **Mariam Nakhlé** and **Jingshu Liu** and **Jean-Gabriel Barthélemy**

Lingua Custodia, Paris, France

`firstname.name@linguacustodia.com`

Abstract

Recent studies have showcased remarkable capabilities of decoder-only models in many NLP tasks, including translation. Yet, the machine translation field has been largely dominated by encoder-decoder models based on the Transformer architecture. As a consequence, scaling laws of encoder-decoder models for neural machine translation have already been well studied, but decoder-only models have received less attention. This work explores the scaling laws of decoder-only models on the multilingual and multidomain translation task. We trained a collection of six decoder-only models, ranging from 70M to 7B parameters, on a sentence-level, multilingual (8 languages) and multidomain (9 domains) dataset. We conducted a series of experiments showing that the loss of decoder-only models can be estimated using a scaling law similar to the one discovered for large language models, but we also show that this scaling law has difficulties to generalize to *too large models* or to a different data distribution. We also study different scaling methods and show that scaling the depth and the width of a model lead to similar test loss improvements, but with different impact on the model's efficiency.

1 Introduction

Most modern machine translation systems are based on Transformers (Vaswani et al., 2017), with an encoder-decoder architecture. Despite the tremendous advances made possible with the release of open-source decoder-only Large Language Models (LLMs) (Jiang et al., 2023; Biderman et al., 2023; Touvron et al., 2023), most NLP tasks still rely on encoder-decoder models. Based on the statistics obtained from the WMT23 shared task on general machine translation (Kocmi et al., 2023), 16 out of the 17 participants submitted a system based on an encoder-decoder model. Yet, recent studies show that decoder-only models can achieve comparable results (Gao et al., 2022; Fu et al.,

2023), or even surpass state-of-the-art encoder-decoder systems, when properly finetuned (Xu et al., 2023). Moreover, the decoder-only architecture is easier to train on massive amounts of data as one can simply concatenate documents and feed as much relevant data as possible into the model during training ; while encoder-decoder models requires either to pad the inputs or rely on complex masking strategies (Raffel et al., 2020) to combine multiple inputs in the same sample.

Furthermore, the decoder architecture is much more flexible than the encoder-decoder architecture as decoders treat all tokens similarly, while encoder-decoders make a distinction between input (source) tokens and output (target) tokens, which are processed, respectively, by the encoder and the decoder. As a consequence, it is more tedious to apply complex *self-reasoning* mechanisms, such as chain-of-thought (Wei et al., 2022), or to interface it with external tools (Schick et al., 2024), because the outputs of such method (the *reasoning* process) should, preferably, be treated as inputs of the model. For the same reasons, it is much more computationally expensive to rely on an encoder-decoder for conversational purposes, making this architecture less efficient for modern workflows such as iterative translation. Indeed, at each round (the user's query and the system's answer) should be appended to the input side, and reprocessed by the encoder for the next round. Decoder-only models support it by design, without needing to recompute the representation of the ever-growing inputs. While we do not explore these directions in this work, we do leverage the flexibility of the decoder architecture to include input-or-output parameters. As we are tackling the multilingual and multidomain machine translation task, the model needs input tokens to represent the language direction and the domain. We propose to train the model to predict the source language and the domain so that, during inference, they can be seamlessly predicted or provided by

the user.

Generally speaking, decoder-only models simply expect the input to be the whole discussion and process it in a single forward step. Causal masking enable efficient caching of already computed keys and values so inference is much cheaper. The main downside of decoder-only over encoder-decoder model is the potential inferior quality of the input representation, as input tokens attend only on past tokens. But it should not be a major issue, as generated tokens attend to the whole past sequence, they do have access to the same quantity of information as with an encoder-decoder model. In addition, previous work propose to update the attention mask so that input tokens can attend to all input tokens while generated tokens can attend only on past tokens (Tay et al., 2022; Raffel et al., 2020).

For all these reasons, we would like to embrace the decoder architecture for machine translation, even if it seems to be the exclusive preserve of encoder-decoder models. The flexibility and the simpler training setup of decoders should make them both more suitable and efficient for most real world applications, and the decoder architecture is more appropriate to answer the ever-growing demand for iterative, interactive and machine assisted translation workflow. To this aim, we study the scaling laws of neural machine translation models under different settings. Our contributions are as follow:

- We show that decoder-only models for translation follow the same scaling law as LLM

- Scaling laws do not scale uniformly across directions and domains and do not generalize well to other directions or domains

- Scaling width-wise and depth-wise yield similar improvements, but the former is more efficient

- We discovered a critical issue related to the packing of training samples in batches and propose a solution to fix it

2 Background

As the size, data requirement, and training costs of language models rise, it quickly becomes critical to estimate the *right* training configuration for a given training budget — expressed in number of floating point operations (FLOP) — required to train the model. Kaplan et al. (2020) discovered a power law relationship between the loss of a language model and its number of parameters, and that larger models perform better given the same amount of data. Even though most work in this area show that larger models tend to be more powerful, recent studies show that other parameters must be taken into account as well. For instance, the Chinchilla scaling law (Hoffmann et al., 2022) shows that model and dataset sizes are loosely tied and need to be scaled equally. In other words, even if increasing only the model size will most likely improve its performances, the compute-optimal solution often requires to also increase the quantity of training data, while preserving the same training cost. These findings had a great impact on LLM research, as researchers stopped increasing blindly the size of their models, in favor of more data, when it was necessary. For instance, the 176B BLOOM (Le Scao et al., 2022) model would probably have been trained very differently (or not at all) if this study was released sooner. As stated in the paper, "in light of revised scaling laws published during training, we decided to train the large models for an additional 25 billion tokens on repeated data", the authors discovered that training such a big model was sub-optimal given the quantity of data they had. As a consequence, many researchers started to work on the collection of large, high quality datasets (Nguyen et al., 2024; Penedo et al., 2023) or on means to enhance existing datasets (Sorscher et al., 2022; Tirumala et al., 2024).

Most of these scaling laws studies focus exclusively on causal generative language models. While it's likely that many of these findings could apply to translation models, the differences between the two tasks cannot be taken for granted. Translation is a lot stricter than causal language modeling since the model has to take into account each information in the source and precisely generate the target sentence without adding or omitting any information. Hence, many studies have naturally emerged to observe the scaling behavior of translation models (Gordon et al., 2021; Fernandes et al., 2023; Ghorbani et al., 2021). Yet, these works focus on encoder-decoder models. For instance, Gordon et al. (2021); Fernandes et al. (2023) showed that, when the encoder and the decoder are scaled proportionally, the model's loss follow a power-law similar to the observation made on language models. Ghorbani et al. (2021) tackle the

problem in a different setup, and propose to scale the encoder and the decoder individually. They show that encoder-scaling and decoder-scaling affect the model's performances differently, and they propose a new formula describing the scaling behavior of the cross-entropy loss as a bivariate function of encoder and decoder size. They found out that scaling decoder is, according to their experiments, always more beneficial, in terms of cross-entropy loss performance, than scaling the encoder.

Recently, Alves et al. (2024) introduced the TowerInstruct, an LLM based on a decoder architecture (LLama 2 (Touvron et al., 2023)) finetuned to handle several translation tasks. They show that a properly finetuned LLM can perform translation better than state-of-the-art models on high-resource languages. But the most promising aspect of this work is the inherent capacity of LLM to handle different tasks. They finetuned TowerInstruct so it can, for instance, clean source sentences before translating them, follow terminological constraints or respect a given level of language. However, this work is still empirical and we do not know, yet, the limits of such models. Inspired by the performances of TowerInstruct, an LLM finetuned for machine translation tasks, we study, in the following, the scaling behavior of decoder-based machine translation models trained from scratch. To this aim, we fit multiple scaling laws to see if translation models follow the same scaling laws as language modeling models (such as the Chinchilla law) or if they follow their own task-specific law.

3   Training methodology

We present in this section all details related to the training of our six models.

3.1   Data

To conduct our experiments, we collected many bilingual data from public repositories (CCMatrix (Schwenk et al., 2021b), WikiMatrix (Schwenk et al., 2021a), UN Parallel Corpus (Ziemski et al., 2016), Paracrawl (Bañón et al., 2020) and Europarl (Koehn, 2005)). We also included a subset of an in-house proprietary dataset collected over time, as well as a small portion of financial documents in order to observe the scaling behavior on domain-specific data. An overview of the dataset distribution is given in Table 1. The financial data is divided into 8 sub-domains, which are described in Appendix A. The data is made of bilingual texts

with one sample being one sentence pair.

Pair	Domain	Sentences	Tokens
en–de	general	46.53 M	2694.16 M
	finance	1.29 M	65.93 M
en–es	general	51.88 M	3525.21 M
	finance	1.34 M	71.48 M
en–fr	general	81.39 M	5430.77 M
	finance	8.29 M	494.47 M
en–it	general	26.21 M	1657.58 M
	finance	0.73 M	36.17 M
en–nl	general	42.74 M	2057.81 M
	finance	1.36 M	63.96 M
en–pt	general	42.02 M	2086.62 M
	finance	0.61 M	22.55 M
en–sv	general	46.35 M	2180.64 M
	finance	0.24 M	9.68 M
fr–de	general	23.60 M	1470.68 M
	finance	1.46 M	72.92 M
fr–es	general	32.90 M	2731.79 M
	finance	0.48 M	23.39 M
fr–it	general	28.02 M	1845.84 M
	finance	1.10 M	61.63 M
fr–nl	general	31.94 M	2034.74 M
	finance	0.62 M	29.18 M
Total:	general	453.58 M	27 715.84 M
	finance	17.53 M	951.36 M
	all	471.11 M	28 667.20 M

Table 1: Distribution of the training dataset. It covers 8 languages over 11 language pairs and 9 domains (general + 8 financial sub-domains).

We applied temperature sampling ($t = 5$) in order to increase the visibility of under represented pairs. Given a collection $\mathcal{D}$ of datasets, the probability of choosing a sample from a dataset $D_i \in \mathcal{D}$ after temperature sampling is given by $P_t(D_i)$ and is calculated from the original dataset statistical distribution $P(D_i)$.

$$P(D_i) = \frac{N_i}{\sum_{j=0}^{|\mathcal{D}|} N_j}$$

$$T(D_i, t) = P(D_i)^{1.0/t}$$

where N_i is the size of dataset D_i and $T(D_i, t)$ is the factor by which the dataset D_i should be oversampled. The new size k_i of the oversampled dataset D_i is given by:

$$k_i = \left\lfloor \frac{T(D_i, t) \cdot \max_{j=0}^{|\mathcal{D}|}(N_j)}{\max_{j=0}^{|\mathcal{D}|}(T(D_j, t))} \right\rfloor$$

Finally, the probability of picking a sample from dataset D_i after temperature sampling is given by

$$P_t(D_i) = \frac{k_i}{\sum_{j=0}^{|D|} k_j}$$

Since the balance between general and financial is also extremely skewed, we applied the temperature sampling separately on the general and financial domains.

3.2 Tokenizer

As we planned to train a multilingual model, we trained a Byte-Level BPE tokenizer (Wei et al., 2021) from scratch because, according to the authors, it is expected to better share the tokens among the multiple languages, resulting in less rare tokens and, hence, better embeddings. The tokenizer has been trained on the whole, non-oversampled, dataset, and we set the vocabulary size to 100 000.

We also reserved a small set of special tokens representing the supported languages and domains. They are inserted inside the input sequence so the model knows this information while generating a translation. For instance, the *English language token* is <lang_en> and the *general domain token* is <dom_general>.

3.3 Data format

Each sample of the datasets has two categories of features: inputs and outputs. Input features are data that will be given during inference, and output features are data that should be predicted by the model. Hence, inputs are the source sentence and the target language (because the model needs to know the desired target language); and outputs are the source language, the domain and the translated sentence.

Predicting the source language is not required, but we decided to include it to give to the model the ability to automatically detect the source language, as it is a very common and handy feature of most commercial translation tools. One could argue that this should be an input parameter, but we decided that the model should be able to classify by itself the language of the source sentence. Yet, the source language token can still be given as input at inference time to force a particular language. This also apply to the domain token.

Since we plan to train a decoder-only model, training samples have been formatted such that the input tokens are first seen by the model, so the model has access to the whole input when generating the first output token. This is why we chose to encode the sentence pairs in the following format:

```
SOURCE </src> <target lang> <source
    lang> <domain> TARGET <eos>
```

where </src> and <eos> are special tokens used to indicate, respectively, the end of the source and target sequences.

This data format gives the possibility to either provide the source language if required, or let the model predict it automatically. For instance, in the real example below, the green part represents the mandatory input (source sentence and target language), the blue part the optional input (source language) and the gray part is the output generated by the model.

```
The buyer pays at an ATM. </src>
<lang_fr> <lang_en> <dom_general>
L'acheteur effectue le paiement sur les
    bornes automatiques. <eos>
```

3.3.1 The <eos> token issue

All the models were trained in the same way LLM are trained. Sentence pairs were packed until the training batch was completely filled. These samples were separated by the usual end-of-sentence token <eos>. Ideally, one should also apply proper masking so tokens cannot attend to tokens from past sentence pairs. However, this features is not implemented in flash-attention 2 (Dao et al., 2022), so we trained the models without masks (except the causal mask). We expect the training task to be slightly more complex to solve, as the model now needs to learn to ignore every token before an <eos> token, but we decided that the gain in training speed is worthwhile.

Model	without <eos>	with <eos>
70M	30.80	41.11
160M	39.12	45.13
410M	40.85	46.82

Table 2: BLEU scores of the same models when sources are prefixed with and without the <eos> token.

Our experiments showed that the quality of translations generated by the models were far below our expectations. We found that the absence of the <eos> token before the source sentence was confusing the model, explaining the drop in translation

quality shown in Table 2. The <eos> token, which was meant to signal the end of the translation, is actually also interpreted as a "start of translation" token. Indeed, during training, all sentence pairs (except the first one) are prefixed with the <eos> token. This phenomenon is clear in the example below, in which three sentence pairs are packed in the same training sample.

```
x₁₋₁ x₁₋₂ x₁₋₃ </src> <tgtlang₁> <srclang₁>
y₁₋₁ y₁₋₂ <eos> x₂₋₁ x₂₋₂ </src> <tgtlang₂>
 <srclang₂> y₂₋₁ y₂₋₂ <eos> x₃₋₁ x₃₋₂ x₃₋₃
</src> <tgtlang₃> <srclang₃> y₃₋₁ y₃₋₂ y₃₋₃
             y₃₋₄ <eos> x₄₋₁ . . .
```

The impact of <eos> absence on the test loss can be seen in Figure 1. The model clearly outputs better translation when the source sentence is prefixed with an <eos> token. This is particularly blatant when comparing the 160M and 410M models, respectively with and without the <eos> token prefix. The 410M model, albeit being more than two times bigger than the 160M model, cannot generate better translations without the <eos> prefix.
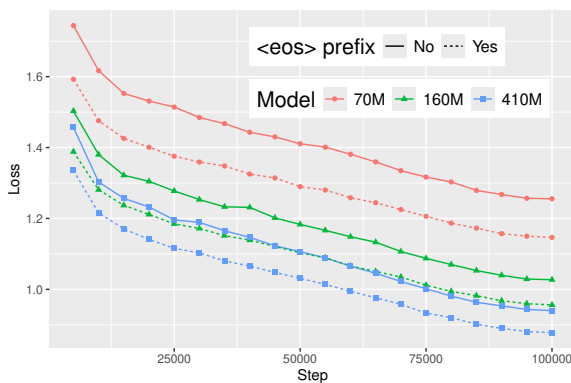

Figure 1: Test loss of our three smallest models (70M, 160M and 410M) with and without the <eos> prefix.

This problem should be negligible when training LLM, as documents are usually longer than sentence pairs, so <eos> tokens are scarcer. However, its impact will increase as batch size grows, since more sentence pairs can be packed into the same batch, making even more *obvious* that sentence pairs *should* start with an <eos> token. We experimented with a relatively small input length (512 tokens) and the absence of the <eos> token during inference already lead to significant drop in performance. Generally speaking, this issue should not be ignored when more than one sequence are packed in a single training sample. When possible, one should properly mask previous training sam-

ples. As it is not possible, currently, to leverage the state-of-the-art self-attention algorithms, we recommend to always prefix all source sentences with the same prefix token(s), both during training and inference. An alternative solution might be to prefix all sequences with a <bos> (begin of sentence) token, but we do not think it will solve this particular issue since the model will likely see that most sentence pairs start with the <eos><bos> sequence, which is still not the intended behavior. In the remaining of this paper, we will only consider translations generated with an <eos> prefix.

3.4 Training strategy

As we aim to train models dedicated to the translation task, we computed the loss only on target tokens, so the model learns to generate only text given a source sentence. This is different from pre-trained language models as there is no notion of source and target sentence. The *target-only* strategy has proven to be effective for training text-to-text models (Touvron et al., 2023), and is also similar to the way loss of encoder-decoder models is calculated, which are commonly used for machine translation (Costa-jussà et al., 2022). Finally, we packed as many sentence pairs that we could in a single batch, in order to increase the training efficiency.

3.5 Model architectures

We used almost the same model architectures used in the Pythia suite (Biderman et al., 2023), the only difference being the number of attention head of the 160M model, as flash-attention expects a multiple of 8. We trained the models using the GPT-NeoX library (Andonian et al., 2023). We made a few changes to the data processing scripts in order to ignore source tokens during the loss computation. An overview of the different models we trained is given in Table 3.

All models are trained with a fixed batch size of 262 144 tokens (512 sequences of length 512 tokens) per GPU, on 8 Nvidia A100 GPUs. The models are trained in bfloat16 precision using the Adam optimizer with weight decay set to 0.1, 100 warmup steps and cosine learning rate decay. The maximum learning rate of sub-1B models is set to 1×10^{-3}, and 1×10^{-4} for larger model because of loss instabilities during the training.

The models are trained for 100 000 steps on approximately 210B tokens, although only half of them were actually used to train the model as we

Model	Non-embedding	Embedding	Layers	Dim	Heads	Max lr
70M	70 295 552	51 380 224	6	512	8	$1e^{-3}$
160M	162 126 336	77 070 336	12	768	16	$1e^{-3}$
410M	405 071 872	102 760 448	24	1024	16	$1e^{-3}$
610M	607 448 064	154 140 672	16	1536	16	$1e^{-3}$
1B	1 011 257 344	205 520 896	16	2048	8	$1e^{-4}$
6.9B	6 855 204 864	411 041 792	32	4096	32	$1e^{-4}$

Table 3: Architectures of the trained models. All models are trained with the very same setup (data, random seed, batch size, number of GPU, . . .). They closely follow the Pythia models but parameters counts do no match because of the bigger vocabulary size, which increases the size of both the embedding and classification layer.

do not take into account source tokens when calculating the loss.

4 Experiments and results

In this section, we will study the impact of variations in training data size and parameters count on the test loss, for all our models. We will also verify if these changes correlate with their real translation performances using standard metrics such as BLEU and COMET. We finally explore two different model scaling strategies.

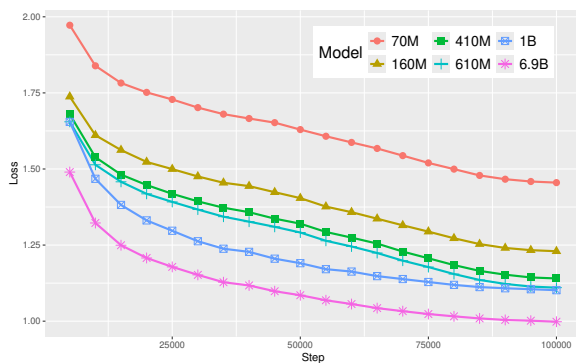### 4.1 Applying machine translation scaling law

Figure 2: Test loss of all model checkpoints. Each step represents 512 training samples. Larger models always converge faster given the same amount of training data.

All existing scaling-laws studies show that larger models exhibit better generalization capabilities (Gordon et al., 2021; Fernandes et al., 2023; Ghorbani et al., 2021; Rae et al., 2021; Kaplan et al., 2020; Biderman et al., 2023). This study is no exception, as can be seen in Figure 2, larger decoder models always converge faster and require less training data to reach the same loss value.

We first fitted multiple curves following the setting of Ghorbani et al. (2021); Fernandes et al.

(2023), who studied scaling laws for machine translation. The form of the law is given below:

$$L(N) = \alpha N^{-p} + \beta \qquad (1)$$

where N is the number of trainable parameters, and the other variables are fitted by minimizing the huber loss (with a delta value of 0.01) using the BFGS algorithm from SciPy (Virtanen et al., 2020).

As shown in Figure 3, the test losses of our translation models can be realistically described by the power law fitted on observations made on all our models (the purple dotted line). This suggests that, indeed, performances of translation models follow a scaling law, that can be expressed by the formula above. We also fitted curves on less data points in order to verify if we could estimate the loss of the 6.9B model. Unfortunately, the fitted curves become deviate from the real observations as soon as we remove the data points from the largest model (the 6.9B model). This is extremely problematic, as the main goal of scaling laws is to estimate the performances of not-yet-trained larger models. Yet, we show that it is difficult to find a good estimation of the 6.9B model's performance without actually training it. For instance, the law fitted on the observations made on the subset 70M-160M-410M-610M-1B (in green) cannot give a good approximation of the *unseen* 6.9B model's performance, and the others are even worse. Therefore, we think one might be particularly cautious when applying such scaling laws to estimate larger models behaviors. Even if our law fitted on all data points seems to be a good estimator of the test loss, we think it will deviate from real observations as the model grows in size.

We also fitted scaling laws on a per-domain and per-direction basis, on all available data points. This is particularly interesting as it highlights discrepancies between domains and directions. As
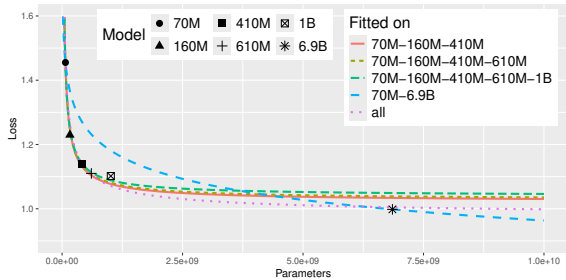
Figure 3: Test losses estimated by power law fitted on different subset of models. Laws fitted on all models and 70M-160M-410M-1B models subset match our observations.
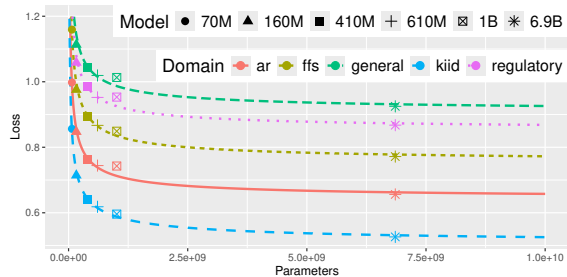

Figure 4: Scaling law fitted on the general domain and some financial subdomains. The law are fitted on the English-French direction only.

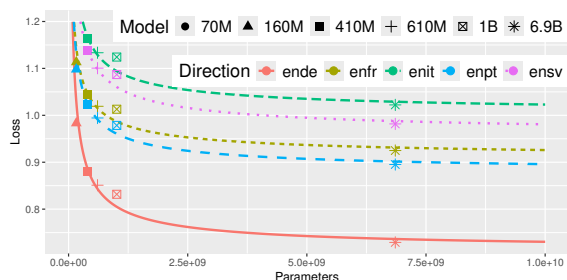Figure 5: Scaling law fitted on the general domain for English-X direction.

shown in Figure 4, it seems to be significantly easier to translate sentences from the *kiid* (Key Investor Information Document) financial domain, but translating general domain sentences is the most difficult, even though the huge majority of our training set is from the general domain. We suspect this curve are, somehow, indicators of the diversity inside each domain. Indeed, *kiid* documents are, by law, all following the same structure and must contain a specific set of information, written in a certain way. On the contrary, *general* domain documents do not follow any rule, making this domain the most heterogeneous one, and thus the most difficult to translate. Other phenomena might explain the differences between these curves. For instance, we also think the presence of many very specific and rare words in the *regulatory* domain explains partly the lower translation quality in this domain.

We also fitted one curve per direction and observed similar phenomena, as shown in Figure 5. For example, our models seem to be better at translating from English to German than from English to French, although our training dataset contains twice as many English-French pairs (before oversampling).

These observations show that the scaling behavior of translation models depends on the training data distribution, and thus scaling laws estimated on a given dataset will not match the real scaling behavior on another one, although they might have the same general shape. For instance, it is not realistic to rely on a scaling law fitted on the en–fr direction to estimate the performances on the en–de direction.

4.2 Applying language modeling scaling law

So far, we experimented with a scaling law formula based on the model size only, ignoring the

training dataset size. Even if we just showed that lower perplexity/loss can be obtained with fewer data samples (in the case of the en–fr and en–de directions), larger training datasets still tend to increase the overall models' quality. But, it's also a waste of computing resources to train a model on more data than required, this is why modern language modeling scaling formula take into account both the number of trainable parameter and the training dataset size. Hence, we fitted multiple Chinchilla laws following the setting of Hoffmann et al. (2022), whose form is given below, on various combinations of input data to see if it can be used to reliably predict model performances.

$$L(N, D) = E + \frac{a}{N^\alpha} + \frac{b}{D^\beta} \qquad (2)$$

E, a, α, b and β are variables fitted by minimizing the huber loss (with a delta value of 0.01) using the `BFGS` algorithm from `SciPy` (Virtanen et al., 2020) ; N and D are, respectively, the number of non-embedding parameters of the model and the number of training samples. More details are given in the original paper.

As shown in Figure 6, the test loss of our translation models can be realistically described by the power law fitted on observations made on all our models (the purple dotted line). Furthermore, the
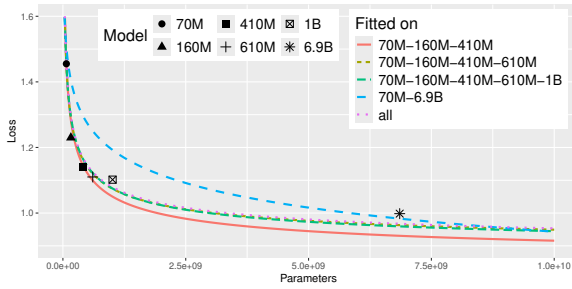
Figure 6: Test losses estimated by the Chinchilla law fitted on different model subsets. Curves deviate from the real observations when we remove too many data points to fit the curve.

general shape of the fitted curves is more stable, and thus more trustworthy. Indeed, the curve fitted on all models is very close to the one fitted without the 6.9B model, indicating that behaviors of larger models can be better estimated with this form of scaling law. However, as with the previous scaling law, the curve deviate from real observations when it is fitted on less data points. While it is not a surprising finding, it shows that scaling laws should not be trusted beyond a certain model size. However, we cannot provide a reasonable window in which the estimated loss is realistic.
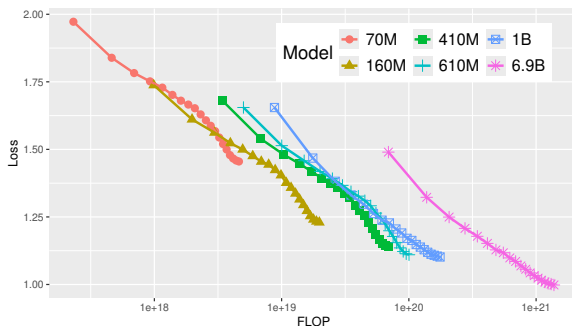

Figure 7: Test loss of all models, each data point represents 5k training steps, or 2.5M samples. Given a fixed FLOP, it's often more beneficial to increase the dataset size when possible.

These experiments shows two things. First, the test loss of decoder-based translation models follows a scaling law similar to language modeling models, as the curves fitted on all data points match the real observations. The form of the law (a power law) indicates that larger models will always generalize better, until a certain point where the curve will stay mostly flat. The second thing we show is that finding a good and universal estimation for the model's loss is very difficult, as fitted curves do not generalize well beyond an unknown model size.
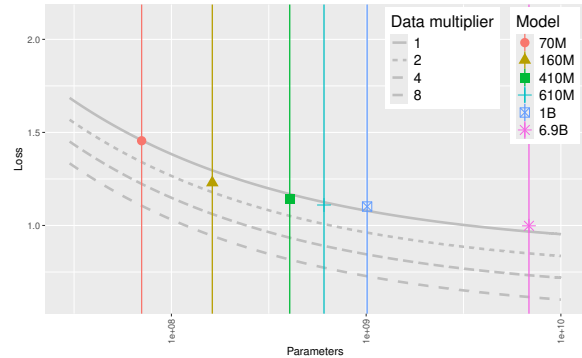

Figure 8: Estimation of models' test losses if they were trained on more data. According to the Chinchilla law fitted on all available observations, the 70M model should be on-par with the 410M performances with four times more data, and the 610M model should match the 6.9B model with only two times more data.

4.3 Correlating scaling law with real translation quality

Let us suppose we know the function modeling the real loss given a model size and an amount of training data. We still do not know if targeting lower loss values will actually improve the quality of the translations generated by the model. We provide in the following an empirical study showing the correlation between the model's loss and its translation performance. We computed BLEU (Papineni et al., 2002), COMET (Rei et al., 2022a) and CometKiwi (Rei et al., 2022b) scores for all six models, and we observed that, indeed, a lower loss does correlate with a performance increase, as shown in Table 4. This trend can be observed on the general domain for all directions, as shown in Appendix C. However, on the financial domain, CometKiwi does not always increase, it reaches a peak on the 610M model, then decreases. We conjecture that CometKiwi cannot correctly evaluate domain specific translations, as it is a reference-free model trained mainly on generalist sentences. We show in Appendix C that BLEU and COMET always increase with models' size, while CometKiwi often decreases at some point.

We also compare our models to well established LLM, and we show that smaller but specialized models clearly outperforms large and generalist LLM, as shown by our 410M model performing on par with Llama 8B. Our largest models are also real competitors to Tower 7B, even though it has been trained on much more data and specialized for machine translation. Tower 7B has the highest CometKiwi score, but as we just showed, it might

Model	BLEU	COMET	CometKiwi
General domain			
70M	29.62	81.31	80.72
160M	32.43	84.00	83.45
410M	33.60	84.81	84.14
610M	34.08	85.10	84.35
1B	34.42	85.10	84.33
6.9B	36.07†	85.88	84.82
Llama3.1 8B	30.43	84.82	84.47
Mistral 7B	23.26	80.08	82.29
Tower 7B	33.50	85.91†	85.02†
Tower 7B*	34.38	86.22	85.23
Financial domain			
70M	44.63	86.95	80.88
160M	49.02	88.27	81.80
410M	50.85	88.64	81.73
610M	52.00	88.85	81.71
1B	53.28	89.98‡	81.61
6.9B	58.34‡	89.62	81.35
Llama3.1 8B	34.99	84.42	81.75
Mistral 7B	38.93	76.52	76.17
Tower 7B	38.93	86.49	82.66‡
Tower 7B*	39.08	86.52	82.74

Table 4: Evaluation of the six models trained during this study on our in-house evaluation dataset. We reports both the scores on the general (G) domain and average over all financial (F) subdomains. We also include best performing LLM. As Tower has not been trained on Swedish, we also evaluate it after removing directions including Swedish (the Tower 7B* rows). Best scores on the general and financial domains are indicated by † and ‡ respectively.

not be reliable for specialized domains. Our models are obviously performing better on the financial domain, because only our models were finetuned on financial data. We also remark that Mistral's scores are quite low on the general domain, a quick manual inspection revealed that the model often give details and explanations about the produced translation, even when asked not to. As a consequence, we think that Mistral lower score is mostly caused by the model not following rigorously the instructions (see Appendix B).

So, while it certainly boost performances, increasing the model size is often not the optimal solution to improve the model's performance. The training dataset is also extremely important. Indeed, as can be observed in Figure 7, given a fixed FLOP budget, it is often preferable to increase the number of training samples. For instance, the 160M model appears to always be better than the 410M, 610M and 1B models given the same FLOP budget, as indicated by the 160M's curve being below other

models' curves. This observation is also validated by the fitted law, as indicated in Figure 8. Most of the time, and according to the fitted Chinchilla law, it would have been better to just train our models on more data, instead of training larger models. For instance, we estimate that the 160M model would be on-par with the 410M model if trained on approximately twice as many data, which would not exceed the total number of FLOP of our current 410M model.

To conclude with, we find that scaling laws are a powerful tool to have a glimpse of what we can expect from a *relatively larger* model trained on the same dataset, but it will probably fail to predict the performances of *much larger* models, even if trained on a similar data distribution. It has to be kept in mind when using such scaling laws to plan a training budget: **at some point, the fitted law will fail**. Planning a training budget based on observations made on a 10B model might be fine to train a 70B model, but completely wrong for a 500B one. Furthermore, a given scaling law can only estimate the end performances of a model trained on the same data distribution used to fit the scaling law. For instance, we show in Figures 4 and 5 that laws fitted on different language directions or domains are very different, and thus should not be applied to estimate the performances of the model on another direction.
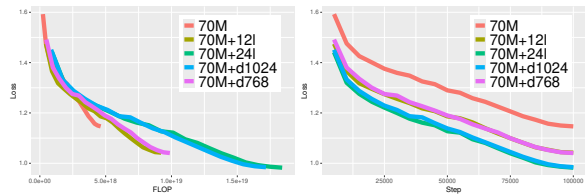
4.4 Scaling strategies

Figure 9: In our experiments, we increased the width and the depth of the 70M model so the additional cost in terms of FLOP is similar (left). Scaling the depth or the width can lead to similar performance gains (right). The two figures are similar, except that the loss decrease can be observed either through the FLOP budget prism (left) or throughout training time / size of dataset (right).

We also studied whether one should favor scaling the depth (increasing the number of layers) or the width (increasing the hidden size) of a decoder model. We took the smallest model as a baseline and scaled it depth-wise and width-wise so that the increase in parameters increased the total training FLOP by a similar amount, as illustrated in

Model	Layers	Dim	Non-embedding	Embedding	FLOP per s.	Samples per s.
70M	6	512	70 295 552	51 380 224	1.06×10^{14}	1170
70M+d768	6	768	119 599 104	77 070 336	1.74×10^{14}	900
70M+12l	12	512	89 209 856	51 380 224	1.37×10^{14}	760
70M+d1024	6	1024	178 339 840	102 760 448	2.43×10^{14}	725
70M+24l	24	512	127 038 464	51 380 224	1.6×10^{14}	445

Table 5: Sizes and architecture of models scaled in depth (70M+12l and 70M+24l) and models scaled in width (70M+d768 and 70M+d1024) compared to the base 70M model. Increasing the depth of the model has limited impact on the total parameters count, but decreases significantly the efficiency (higher FLOP per second but less training samples per second). Scaling the width of the model takes advantage of modern GPU architectures, but adds many trainable parameters.

Figure 9. An overview of the scaled model architectures can be seen in Table 5. Interestingly, we observed that both scaling methods yield the same performance improvement. As shown in Figure 9, given a similar FLOP cost, scaling the depth or the width seems to have the very same impact on the test loss.

Generally speaking, scaling depth-wise lead to smaller, but less efficient models. Indeed, modern hardware architecture can handle more efficiently large matrix products than many smaller matrix products. As shown in Table 5, width-scaled models are faster than depth-scaled models because the GPU can do more FLOP per second.

5 Conclusion

This work describes the behavior of decoder-only models on the multilingual multidomain machine translation task. We trained six models whose number of parameters range from 70M to 6.9B on sentence pairs in eight European languages. We found that scaling laws for machine translation cannot describe the general behavior of translation models, but they can still provide good estimation in a given domain, language-pair, and range of model sizes'. Indeed, We show that decoder-only models for translation tend to scale similarly as language models, as the Chinchilla law can also be applied to our models. As such, we recommend to train machine translation models using the same training recipes as large language models. While we think it is true for most, if not all, NLP tasks, more work need to be carried out to validate this hypothesis. However, we also highlight a critical limitation of scaling laws: they cannot generalize well beyond an unknown model and/or training dataset size. As models tend to be larger through time, it will be extremely important to find ways to detect early unreasonable deviations of the "reference" scaling laws on which larger models are build.

We also show that models scaled width-wise appear to be more FLOP efficient than models scaled depth-wise, while reaching almost the same loss. Our experiments need to be continued in order to see when increasing the depth of the model starts to be more valuable than increasing its width. But, generally speaking, increasing the linearly both the depth and the width seems to be a good trade-off between efficiency and parameters count.

Efficient training requires packing as many sentence pairs as possible in a training batch. We discovered that unexpected biases can be introduced if proper masking is not applied, that is to say, if sequences can attend to previous ones. Since it is not possible with current state-of-the-art optimization methods, one must carefully format the training input data. We suggest dropping the *end-of-sentence* token, commonly used to signal the end of text generation, in favor of a *start-of-translation* token signaling the start of a new source sentence and, therefore, the end of the generated target sentence.

This study has been conducted on sentence-level pairs only. While this setup is a bit outdated, it is still the first time a comprehensive study has been made on multilingual machine translation using decoder-only architectures. Nevertheless, we expect decoder models to be easy to adapt to the document-level translation task, as one can simply finetune a sentence-level decoder with non-shuffled sentence pairs from a corpus of parallel documents.

6 Acknowledgment

References

Duarte M. Alves, José Pombal, Nuno M. Guerreiro, Pedro H. Martins, João Alves, Amin Farajian, Ben Peters, Ricardo Rei, Patrick Fernandes, Sweta Agrawal, Pierre Colombo, José G. C. de Souza, and André F. T. Martins. 2024. Tower: An open multilingual large language model for translation-related tasks.

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Jason Phang, Shivanshu Purohit, Hailey Schoelkopf, Dashiell Stander, Tri Songz, Curt Tigges, Benjamin Thérien, Phil Wang, and Samuel Weinbach. 2023. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch.

Marta Bañón, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, Mikel L. Forcada, Amir Kamran, Faheem Kirefu, Philipp Koehn, Sergio Ortiz Rojas, Leopoldo Pla Sempere, Gema Ramírez-Sánchez, Elsa Sarrías, Marek Strelec, Brian Thompson, William Waites, Dion Wiggins, and Jaume Zaragoza. 2020. ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4555–4567, Online. Association for Computational Linguistics.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Patrick Fernandes, Behrooz Ghorbani, Xavier Garcia, Markus Freitag, and Orhan Firat. 2023. Scaling laws for multilingual neural machine translation. In *International Conference on Machine Learning*, pages 10053–10071. PMLR.

Zihao Fu, Wai Lam, Qian Yu, Anthony Man-Cho So, Shengding Hu, Zhiyuan Liu, and Nigel Collier. 2023.

Decoder-only or encoder-decoder? interpreting language model as a regularized encoder-decoder. *arXiv preprint arXiv:2304.04052*.

Yingbo Gao, Christian Herold, Zijian Yang, and Hermann Ney. 2022. Is encoder-decoder redundant for neural machine translation? In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 562–574.

Behrooz Ghorbani, Orhan Firat, Markus Freitag, Ankur Bapna, Maxim Krikun, Xavier Garcia, Ciprian Chelba, and Colin Cherry. 2021. Scaling laws for neural machine translation. *arXiv preprint arXiv:2109.07740*.

Mitchell A Gordon, Kevin Duh, and Jared Kaplan. 2021. Data and parameter scaling laws for neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5915–5922.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 30016–30030.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, et al. 2023. Findings of the 2023 conference on machine translation (wmt23): Llms are here but not quite there yet. In *Proceedings of the Eighth Conference on Machine Translation*, pages 1–42.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of machine translation summit x: papers*, pages 79–86.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model.

Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. 2024. CulturaX: A cleaned, enormous, and multilingual dataset for large language models in 167 languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4226–4237, Torino, Italia. ELRA and ICCL.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Ricardo Rei, José GC De Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André FT Martins. 2022a. Comet-22: Unbabel-ist 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585.

Ricardo Rei, Marcos Treviso, Nuno M Guerreiro, Chrysoula Zerva, Ana C Farinha, Christine Maroti, José GC De Souza, Taisiya Glushkova, Duarte M Alves, Alon Lavie, et al. 2022b. Cometkiwi: Istunbabel 2022 submission for the quality estimation shared task. *arXiv preprint arXiv:2209.06243*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.

Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong, and Francisco Guzmán. 2021a. Wikimatrix: Mining 135m parallel sentences in 1620 language pairs from wikipedia. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1351–1361.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Édouard Grave, Armand Joulin, and Angela Fan. 2021b. Ccmatrix: Mining billions of high-quality parallel sentences on the web. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6490–6500.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. 2022. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, et al. 2022. Ul2: Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos. 2024. D4: Improving llm pretraining via document de-duplication and diversification. *Advances in Neural Information Processing Systems*, 36.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Junqiu Wei, Qun Liu, Yinpeng Guo, and Xin Jiang. 2021. Training multilingual pre-trained language model with byte-level subwords. *arXiv preprint arXiv:2101.09469*.

Haoran Xu, Young Jin Kim, Amr Sharaf, and Hany Hassan Awadalla. 2023. A paradigm shift in machine translation: Boosting translation performance of large language models. *arXiv preprint arXiv:2309.11674*.

Michał Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1. 0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3530–3534.

A Full data distribution

Our models were trained on 11 language directions and 9 domains (8 are financial subdomains + general domain). The list 8 financial subdomains are given below:

am Asset Management

ar Annual Report

corporateAction Corporate Action Document

equi Equity Research

ffs Fund Fact Sheet

kiid Key Investor Information Document

lifeInsurance Life Insurance Document

regulatory Regulatory Document

B Prompt templates

We used the following system prompt to generate translation with Llama, Tower and Mistral:

```
You are an expert translator.
    The user will ask you to
    produce translations,
    generate only the asked
    translation, do no justify
    nor explain anything.
```

We used the following instruction template to query the models:

```
Translate from {SOURCE_LANGE}
    to {TARGET_LANG} the text
    below.
{SOURCE_TEXT}
```

C Models' performances per direction

Performances of all models increase as parameters counts increase, regardless of the scoring method, as shown in Figures 10 and 11.
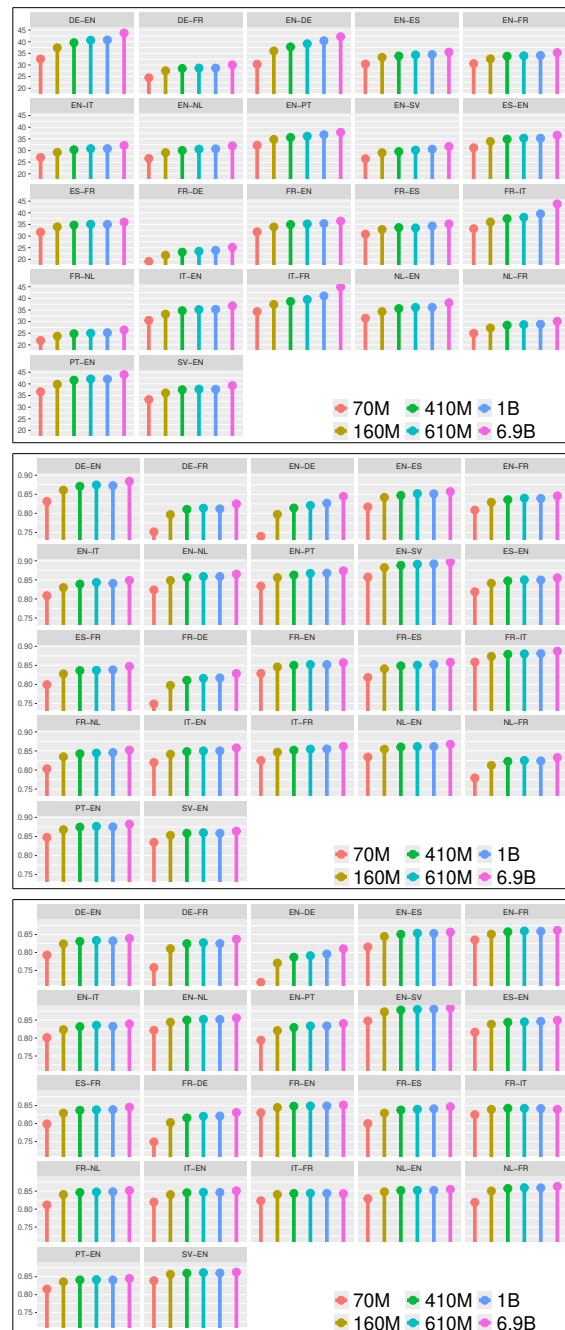

Figure 10: From top to bottom, BLEU, COMET and CometKiwi scores computed on the test dataset for all models and directions, on the general domain.
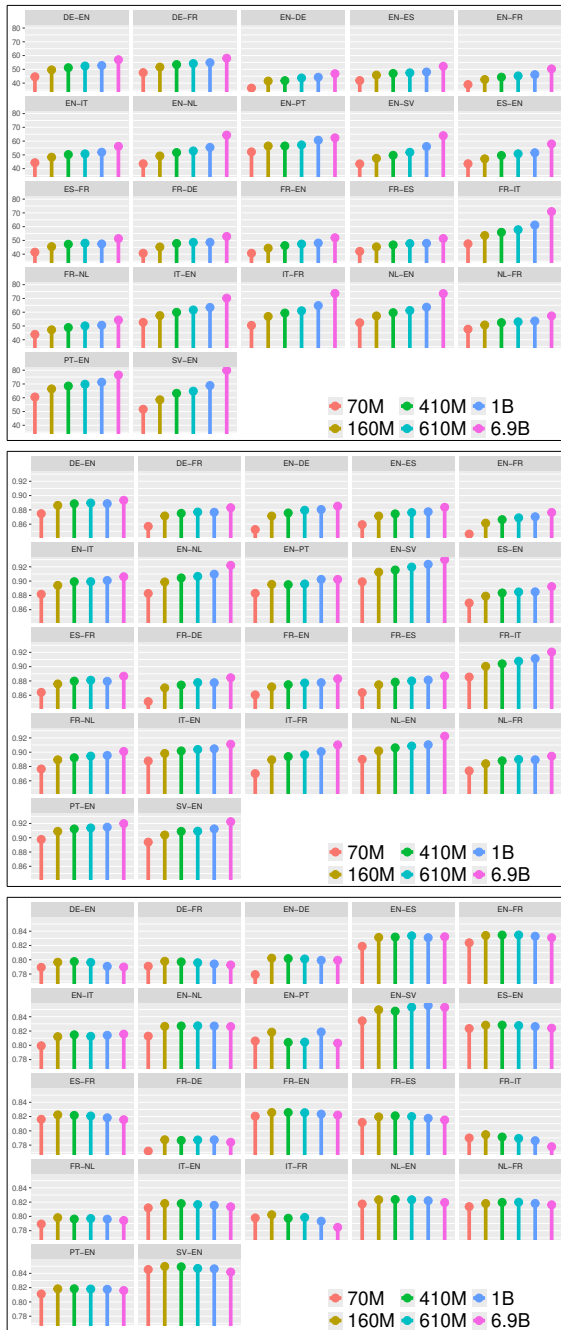
Figure 11: From top to bottom, BLEU, COMET and CometKiwi scores computed on the test dataset for all models and directions, averaged over all financial sub-domains.