Arrows of Time for Large Language Models

Vassilis Papadopoulos [*1 2] **Jérémie Wenger** [3] **Clément Hongler** [*2]

Abstract

We study the probabilistic modeling performed by Autoregressive Large Language Models (LLMs) through the angle of time directionality, addressing a question first raised in (Shannon, 1951). For large enough models, we empirically find a time asymmetry in their ability to learn natural language: a difference in the average log-perplexity when trying to predict the next token versus when trying to predict the previous one. This difference is at the same time subtle and very consistent across various modalities (language, model size, training time, ...). Theoretically, this is surprising: from an information-theoretic point of view, there should be no such difference. We provide a theoretical framework to explain how such an asymmetry can appear from sparsity and computational complexity considerations, and outline a number of perspectives opened by our results.

1. Introduction

Generative Models have revolutionized modern AI, yielding a wide array of applications. Modern works have shown that such models can perform spectacularly (and somewhat mysteriously) well on various kinds of data. Text is perhaps the domain where progress has been the most drastic: in a few years, Large Language Models (LLMs) have gone from generating barely correct sentences to producing consistent stories, code, and performing countless new tasks; key milestones include the Transformer architecture (Vaswani et al., 2017), BERT (Devlin et al., 2019), and GPTs (Radford et al., 2018; 2019; Brown et al., 2020; OpenAI, 2023).

At the heart of these developments are probabilistic models trained in an unsupervised manner on vast amounts of data, for prediction or recovery tasks: this yields an estimation of the probability measure underlying the data. These probabilistic models appear to gain surprising abilities, such as reasoning, as their sizes increase (see (Wei et al., 2022; Schaeffer et al., 2023) among others).

In this work, we investigate the interplay between the probabilistic structure of autoregressive LLMs and the data they are trained on. More precisely, we investigate how time directionality influences their ability to model natural and synthetic languages.

1.1. Autoregressive LLMs

Famously, the pre-training of LLMs such as the GPTs consists in 'learning to predict the next token' knowing previous ones, in sequences extracted from large text corpora, using the natural time ordering of the data they are being trained on. A vocabulary $\mathcal{V}$ of V tokens is chosen; the dataset is then tokenized into a sequence of tokens in $\mathcal{V}$; at each step, the model reads a sequence of tokens and outputs a probability distribution on $\mathcal{V}$ predicting the next token.

Typically, as a probabilistic model, an autoregressive model will estimate the probability that n random consecutive tokens $(X_1, \cdots, X_n)$ are equal to $(x_1, \cdots, x_n) \in \mathcal{V}^n$ by taking the product of the (estimations of) the probabilities

$$\begin{aligned} &\mathbb{P}\{X_1 = x_1\} \\ &\mathbb{P}\{X_2 = x_2 | X_1 = x_1\} \\ &\vdots \\ &\mathbb{P}\{X_n = x_n | X_1 = x_1, \cdots, X_{n-1} = x_{n-1}\}, \end{aligned} \tag{1}$$

yielding an estimated probability measure $\mathbb{P}_n^{\rightarrow}$ on $\mathcal{V}^n$.

Autoregressive LLMs (GPTs, GRUs, LSTMs, ...) thus factorize (their estimates of) the joint probabilities in terms of conditional probabilities for each token knowing past ones. This brings a number of advantages: first, this leverages the fact that for each token sequence $x_1, \ldots, x_n$, each token x_k is used in to predict each token x_ℓ with $\ell > k$. In particular, GPT (compared to the earlier BERT) includes causality-aware attention, allowing for a parallelization of the training process: a sequence $x_1, \ldots, x_n$ generates $n-1$ fully parallelizable tasks (predict x_k from $(x_1, \ldots, x_{k-1})$ for $2 \leq k \leq n$). Also, this representation enables a natural

[*] Equal contribution [1] FSL/Institute of Physics, EPFL [2] CSFT/Institute of Mathematics, EPFL, Lausanne, Switzerland [3] Department of Computing, Goldsmiths/University of London, London, UK. Correspondence to: Clément Hongler <clement.hongler@epfl.ch>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

sampling from $\mathbb{P}_n^{\rightarrow}$ (token by token), as well as data compression: the factorization decomposes these processes into many smaller substeps (Graves et al., 2023).

Autoregressive LLMs such as GPTs have enabled a massive scaling up of the number of parameters and dataset sizes, yielding numerous fascinating phenomena, e.g. scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022) and emergent behavior, e.g. abilities at arithmetic operations (Shen et al., 2023), circuit computing tasks (d'Ascoli et al., 2023), or high-level linguistic proficiency.

1.2. Arrow of Time and Language Models

While decomposing measures into a sequence of conditional probabilities is natural, it is not a priori clear why following the time direction of language to do so is optimal (except for downstream tasks, e.g. making a chatbot): what is the best order when predicting the token probabilities? A natural idea to investigate this question is simply to reverse the Arrow of Time: to estimate probabilities *backward*. This amounts to training models on time-flipped datasets: we train the same models on the same data slices for next-token predictions, but for each data slice $(x_1, \cdots, x_n)$ we feed the model with $(x_n, x_{n-1}, \cdots, x_1)$ instead.

As a result, instead of (1), we take the product of the estimations of

$$\begin{aligned}
&\mathbb{P}\{X_n = x_n\} \\
&\mathbb{P}\{X_{n-1} = x_{n-1} | X_n = x_n\} \\
&\vdots \\
&\mathbb{P}\{X_1 = x_1 | X_n = x_n, \cdots, X_2 = x_2\}.
\end{aligned} \tag{2}$$

This yields an estimated probability measure $\mathbb{P}_n^{\leftarrow}$ on $\mathcal{V}^n$.

In this paper, we will speak of *forward/backward (FW/BW) model* to refer to the same (architectural) model trained with the same hyperparameters (learning rate, batch size, training time, ...) but fed with (batches of) $(x_1, \ldots, x_n)$ and $(x_n, \ldots, x_1)$ from the same dataset respectively. In other words, both models are the same, except that the FW model is trained to predict the *next* token, while the BW one is trained to predict the *previous* token.

Problem 1. For a measure $\mathbb{P}$ and a given model, how do the *forward and backward measures* $\mathbb{P}_n^{\rightarrow}$ and $\mathbb{P}_n^{\leftarrow}$ differ from one another?

For certain $\mathbb{P}$s, we will see universal asymmetries: for any given architecture and hyperparameters, a substantial difference between the way $\mathbb{P}_n^{\rightarrow}$ and $\mathbb{P}_n^{\leftarrow}$ approximate $\mathbb{P}$ arises.

1.3. Cross-Entropy Loss and Perplexity

LLMs are trained as follows: sample sequences of n consecutive tokens $(x_1, \ldots, x_n)$ from the dataset; then, for $i = 1, \ldots, n$, get a prediction $p_i : \mathcal{V} \to [0, 1]$ for X_i (given previous tokens for the FW model/next tokens for the BW one), compute the loss $\sum_{i=1}^n \ell(p_i, x_i)$ on the observed tokens x_i for a loss function ℓ, perform a gradient step to optimize ℓ, and start again.

In the training of most LLMs, the prime choice for ℓ is the *cross-entropy loss*, defined by $\ell_{\mathcal{C}}(\mathbf{p}_k, x_k) = -\ln \mathbf{p}_k(x_k)$: the negative log of the *predicted probability of the observed token*. It is a *proper scoring rule* (Savage, 1971; Gneiting & Raftery, 2007), uniquely identified by certain modularity properties (Hanson, 2012); in expectation, it gives the number of nats ($\ln 2$ times the number of bits) needed to compress $(x_1, \ldots, x_n)$ when using a coding scheme based on the model's estimated probabilities. Finally, and crucially for us, we have the following:

Remark 2. For $i = 1, \ldots, n$, let $(\mathbf{p}_i^{\rightarrow})_i$ and $(\mathbf{p}_i^{\leftarrow})_i$ denote the predictions of the FW and BW models respectively. Setting $\ell_i^{\leftrightarrows} := \ell_{\mathcal{C}}\left(\mathbf{p}_i^{\leftrightarrows}, x_i\right)$, we have

$$\sum_{i=1}^n \ell_i^{\rightleftarrows} = -\ln \mathbb{P}_n^{\leftrightarrows}\{X_1 = x_1, \cdots, X_n = x_n\}.$$

In particular, if the FW and BW measures coincide, the cross-entropy losses are identical.

Remark 3. If $(x_1, \ldots, x_n)$ is sampled from $\mathbb{P}_n$, denoting by $\mathcal{L}_n^{\leftrightarrows}$ the expectations of $\sum_{i=1}^n \ell_i^{\leftrightarrows}$ (estimated by the test loss of the models during training), we have

$$\mathcal{L}_n^{\leftrightarrows} = \mathrm{D}_{\mathrm{KL}}\left(\mathbb{P}_n \middle|\middle| \mathbb{P}_n^{\leftrightarrows}\right) + H\left(\mathbb{P}_n\right),$$

where H denotes the entropy and D_{KL} the Kullback-Leibler divergence.

Remark 4. It is worth noting that, in spite of its apparent triviality, Remark 2 crucially depends on the choice ℓ as $\ell_{\mathcal{C}}$. Moreover, even if $\mathbb{P}_n^{\rightarrow} = \mathbb{P}_n^{\leftarrow}$, we will generally have $\ell_i^{\rightarrow} \neq \ell_i^{\leftarrow}$ for $1 \leq i \leq n$ (though $\sum_i \ell_i^{\rightarrow} = \sum_i \ell_i^{\leftarrow}$). When $\mathbb{P}_n^{\rightarrow} = \mathbb{P}_n^{\leftarrow} = \mathbb{P}_n$ we have that $\ell_i^{\rightarrow}$ and $\ell_i^{\leftarrow}$ yield two (typically different) decompositions of the log-likelihood of $(x_1, \ldots, x_n)$. For instance, take as a dataset the 81 expressions $A \times B = CD$ for $1 \leq A, B \leq 9$, $0 \leq C, D \leq 9$ (setting $C = 0$ when needed). The FW log-perplexity is concentrated on A and B, each contributing $\ln 9 \approx 2.2$ nats: $\ell_A^{\rightarrow} = \ell_B^{\rightarrow} \approx 2.2$, $\ell_C^{\rightarrow} = \ell_D^{\rightarrow} = 0$. The BW log-perplexity is distributed differently: for instance, for $3 \times 4 = 12$, $(\ell_A^{\leftarrow}, \ell_B^{\leftarrow}, \ell_C^{\leftarrow}, \ell_D^{\leftarrow}) \approx (0, 1.39, 1.1, 1.91)$.

Remark 5. Remark 3 suggests that if $\mathbb{P}_n^{\rightarrow}$ and $\mathbb{P}_n^{\leftarrow}$ coincide (e.g. if both models have learned the true measure $\mathbb{P}$, memorized the training set, or more generally have learned $\mathbb{P}$ 'equally well'), their associated average losses should be equal. If we take a very small dataset or context length, we can expect to have $\mathcal{L}_n^{\rightarrow} \approx \mathcal{L}_n^{\leftarrow}$. If we are to train a FW and a BW model with our setting, any substantial difference in their cross-entropy losses *will necessarily reflect an asymmetry of* $\mathbb{P}$ (w.r.t. its learnability by the models).

As it will turn out, for many types of data (i.e. $\mathbb{P}$s), a consistent difference between FW and BW log-perplexities arises across a wide range of models and hyperparameters.

1.4. Setup and Plan

In Section 1.3, we showed that a difference between FW and BW losses reflects a difference between the measures $\mathbb{P}_n^{\rightarrow}$ and $\mathbb{P}_n^{\leftarrow}$ learned by the FW/BW models, all else being equal: same dataset, same model (architecture and hyperparameters). In such a case, we say that $\mathbb{P}$ (or the corresponding dataset) exhibits an *Arrow of Time (AoT)* with respect to the model and context length n. We speak of a *FW AoT* if the average FW log-perplexity is below the BW one (i.e. if the FW model outperforms the BW one).

This paper aims to investigate the following questions:

- Is there an AoT in large natural language datasets? Does it depend on the language? Does it depend on the context length n?
- Can we formulate a theoretical framework explaining the presence of an AoT? Can we construct simple synthetic datasets exhibiting an AoT? Can the presence of an AoT be explained mathematically from first principles? What should we expect as the model sizes tend to infinity?

The paper is organized as follows:

- In Section 2, we investigate the existence of an AoT, starting from a basic setup and expanding across modalities: languages, architectures, hyperparameters, context lengths.
- In Section 3, we investigate the theoretical origins of AoTs, starting with a simple synthetic dataset exhibiting one; in this case, the difference between $\mathcal{L}_n^{\rightarrow}$ and $\mathcal{L}_n^{\leftarrow}$ can be shown to be related to the hardness of the factoring problem (Section 3.1). We then introduce a more general class of synthetic datasets which we call 'linear languages' (Section 3.2), providing us with a fairly wide class of datasets with a mathematically justified AoT.
- Finally, in Section 4, we summarize our results and outline a number of possible future research directions.

1.5. Relation to Previous Works in Language Modeling

To the best of our knowledge, the question of comparing FW and BW text generation in Language Modeling was first raised in (Shannon, 1951): Shannon ran experiments on the task of predicting the next vs previous letters, noting the theoretical equality between FW and BW entropies; he noted that while the BW prediction appeared to be "subjectively much more difficult" for humans, it led to "only slightly poorer" scores.

A notable recent example is (Sutskever et al., 2014), focusing on machine translation using LSTMs, finding that reversing the source sentence (i.e. training the source LSTM backwards) improves performance. Other well-known examples of related techniques include ULMFiT (Howard & Ruder, 2018) and ELMO (Peters et al., 2018), the already mentioned BERT (Devlin et al., 2019), as well as T5 (Raffel et al., 2023), and XLNET (Yang et al., 2020).

Attempts at combining FW/BW models include (Mou et al., 2016; Liu et al., 2016; Zhang et al., 2018; Serdyuk et al., 2018; Mangal et al., 2019) (using RNNs); or recently (Nguyen et al., 2023), a 'Meet in the Middle' approach which shows how pre-training using FW/BW Transformer models enhance FW-only autoregressive generation; as well as in (Shen et al., 2023), applying the idea of reversing data to improve LLM performance (see also Section 3.1). In these approaches, the FW/BW models are treated as one model, yielding one combined loss. They compare various models on a task (see section 5 of (Nguyen et al., 2023) for a comprehensive review), rather than study potential discrepancies in FW/BW learning.

Some results showing BW models performing equally or even better than FW models can be found in the literature. While (Vinyals et al., 2016) highlights the importance of order (of input and output sequences) for performance, and shows that scrambling words reduces performance, it shows FW and BW seemingly performing equally well. In a recent work (Pfau et al., 2023) use BW GPT models to perform adversarial attacks on LLMs, showing slightly better accuracy BW than FW. An older study (Duchateau et al., 2002) based on trigram models, also seemingly reports better BW than FW performances. Note that these works do not affect our confidence in our results, given the magnitude of our experiments and the level of care involved in their setup.

A number of works, in particularly related to the machine-translation setup, try to use token re-ordering to improve performances in one way or another, see, e.g. (Wu et al., 2018; Oord et al., 2017; Gu et al., 2018; Lee et al., 2018; Ford et al., 2018; Savinov et al., 2022; Welleck et al., 2019; Stern et al., 2019; Gu et al., 2019b; Chan et al., 2019b;a; Gu et al., 2019a; Emelianenko et al., 2019; Mansimov et al., 2020). See (Xiao et al., 2023) for a survey. Given the extensive body of work on the topic, it comes across as somehow surprising that the effect we highlight in our paper is not noted anywhere (besides in the early works of Shannon). Among possible reasons for this could be the use of translation-specific metrics such as the BLEU score (Papineni et al., 2001), rather than cross-entropy losses, and the lack of careful setups comparing FW and BW performance for large models on large datasets, all else being equal.

1.6. Causality and Information Theory

While the AoT effect we highlight in this paper is surprising from the point of view of information theory, there are several theoretical frameworks that appear to be related to this effect:

- Structural Causal Models (Peters et al., 2017) consist of families of random variables linked by certain relationships that (implicitly) involve a notion of time: consider random variables $X_1, \dots, X_n$ such that for each $i \geq 1$, X_{i+1} is a written as $f_i(X_1, \dots, X_i, Z_i)$, where the Z_i's are jointly independent. While the presence of such a decomposition is not special to the order in which the variables are labeled, an order may be singled out in certain cases if we put constraints on the structure of f_i and Z_i (e.g. that f_i, Z_i are 'simple' in some sense): if we e.g. reverse the order of the variables $(\tilde{X}_1, \dots, \tilde{X}_n) = (X_n, \dots, X_1)$, it may not be possible to write $\tilde{X}_{i+1} = \tilde{f}_i(\tilde{X}_1, \dots, \tilde{X}_i, \tilde{Z}_i)$, with $\tilde{f}_i, \tilde{Z}_i$ being as 'simple' as f_i, Z_i. This may have an impact on learnability (Scholkopf et al., 2021), akin to that of an AoT.
- Computationally-constrained views on information theory, in particular the recent framework of *$\mathcal{V}$-information* (Xu et al., 2020), allow one to take into account the computational challenges associated with the information extraction; through the lens of the $\mathcal{V}$-information, we see the emergence of symmetry breaking, an example of which is the Arrow-of-Time effect.

2. Empirical Results on Natural Language

In this section, we explore the existence of an AoT for LLMs in natural language datasets. We first reveal the presence of an AoT in a basic setup (GPT2 models on English and French with context window of length 256, see Section 2.2.1 below). We then decline our explorations over more than 50 model modalities (GPT/GRU/LSTM architectures, 6 GPT sizes, context window lengths of 16/32/64/128/256/512 and 8 languages), and rule out possible tokenization artifacts. We observe a consistent FW AoT in these setups, including a number of takeaways concerning its magnitude.

2.1. Setup

For the identification of an AoT in a dataset, we make sure that both the FW and BW models are trained with the exact same specifications: the only reason for a difference between the models' performances is hence the token prediction order. In all experiments, the models are trained from scratch, using He initialization (He et al., 2015).

2.1.1. Dataset and Tokenization

We conduct our natural language experiments on the CC-100 dataset (Wenzek et al., 2019; Conneau et al., 2020), which provides large monolingual text datasets for various of languages and is reasonably homogeneous across languages. This dataset is made of Commoncrawl snapshots, filtered for quality by comparing the data with a Wikipedia-trained model (Wenzek et al., 2019) (use the Huggingface viewer to explore the dataset). For each language, we train from scratch a BPE tokenizer (Sennrich et al., 2016), with a vocabulary size of 50257, the same as GPT2 (Radford et al., 2019), including the beginning of sentence $\langle \text{BOS} \rangle$ token.

To train on length-n data batches, we split the dataset into 'sentences' of $n - 1$ tokens, with a stride of $\frac{n}{2}$, ensuring that each token can be seen at least once with reasonable context. For the FW model, we add the $\langle \text{BOS} \rangle$ token at the start of the sequence, while for the BW model, we add the $\langle \text{BOS} \rangle$ token at the end, and flip the token order before feeding it to the model. We withhold $\sim 250k$ sentences from the dataset for validation.

2.1.2. Models, Hyperparameters and Training

While some experiments involve other autoregressive models (GRU, LSTM), for most training jobs we use the decoder-only Transformer (GPT) (Radford et al., 2018); our implementation (with code in the supplementary material) is derived from minGPT (Karpathy, 2023). All GPT experiments use learned positional embeddings and a dropout rate of 0.1. Other hyperparameters may depend on the experiment, see the Appendix.

For all models, we use the AdamW optimizer (Loshchilov & Hutter, 2019) with base learning rate of 10^{-4} and a learning rate schedule with a warmup, followed by cosine annealing with warm restarts (Loshchilov & Hutter, 2017). These hyperparameters are mostly kept constant across different experiments, although the period of the warm restarts might be tweaked to synchronize the end of training with the end of a cycle, see Appendix A for details.

2.2. Results

In this section, we present results of various experiments confirming the presence of a FW AoT in English and French datasets, and provide compelling evidence for its universal existence in natural languages by considering six other languages (five distinct families in total).

2.2.1. Arrow of Time in English and French

We begin in this section by analyzing the difference in FW vs BW training dynamics for a Transformer of size GPT2-Medium ($\sim 405M$ parameters, context window length of 256) on the CC-100 datasets for English and French. We train the FW and BW models for the equivalent of 1 epoch of the French dataset ($\sim 27B$ tokens), avoiding memorization.

As is seen in the zoom-in of Fig 1., after an initial short transition period, the BW model loss separates from its FW counterpart and settles slightly above it, and then follows an almost parallel trajectory. This consistent difference

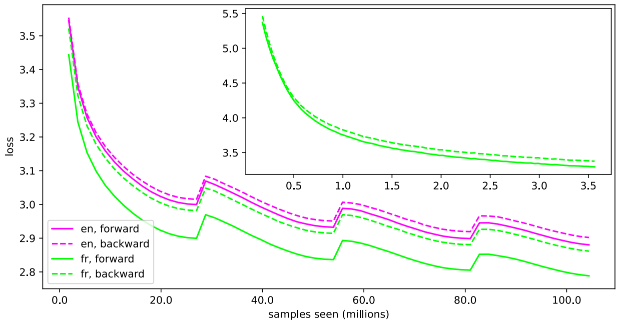

Figure 1: English vs French validation losses (French training losses in the zoom-in, early loss values cropped for readability).

throughout training (even persisting through warm restarts) points to the existence of an AoT both in English and in French: at the end of training, we see the following losses for English: FW: 2.88, BW: 2.902 a difference of $+0.76\%$; and for French: FW: 2.788, BW 2.862, a difference of $+2.65\%$. Interestingly, the magnitude of this effect is different for English and French.

As will be discussed in the next subsections, the findings are quite universal: they can be consistently expanded to various settings, across models, languages, and context lengths.

2.2.2. Context Window Size

In this section, we examine the influence of long-range correlations on the AoT, by studying its relationship with the context length. Intuitively, for a very small context length, we should see virtually no AoT; with very few tokens, models approach the optimal solutions similarly, as they have fewer degrees of freedom. For instance, in the extreme case of a context of length 2, models are only tasked with learning a two-variable function $\mathcal{V}^2 \to [0, 1]$, i.e. to learn the frequencies of 2-grams, which should be (equally) easy in both directions. It is likely that an AoT emerges for larger context lengths (and for reasonably large models).

We test the dependence on the context window by training the same GPT-Medium model, but with context lengths spanning from 16 to 512 tokens, both on English and French. As can be seen in Fig. 2, the magnitude of the AoT in both English and French increases with the context size, suggesting the importance of long-range dependences.

2.2.3. Model Size

In this section, we investigate the effect of model size for GPT models (other models are discussed in the next subsection). As in Section 2.2.2 above, it is natural to expect small models to struggle to exhibit an AoT that would depend on sophisticated, long-range dependences. To test this, we train GPT models of different sizes, from $5M$ to $405M$ parameters, all with a context length of 256. Interestingly the AoT is much smaller at very small model sizes, reinforcing the idea that long-range dependences are key; as the model size keeps growing beyond that, the difference tends to grow. Note also that larger BW models typically outperform smaller FW ones.

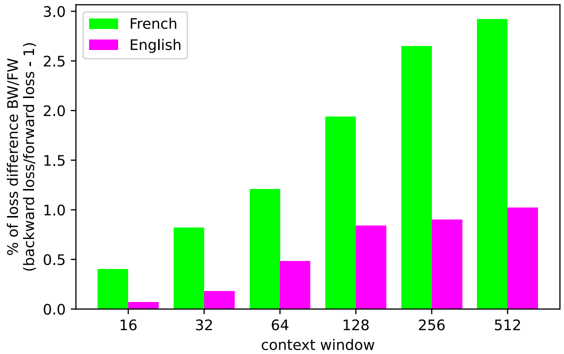

Figure 2: BW/FW losses percentage difference for different context lengths

Table 1: Final FW losses and relative BW differences.

Size	Nano 4.92M	Micro 13.7M	Mini 22.0M	Small 55.6M	GPT1 162M	Med 405M
Fr-FW	4.525	3.964	3.683	3.293	2.979	2.788
Fr-BW	+0.15%	+0.63%	+1.49%	+1.64%	+2.07%	+2.65%
En-FW	4.599	4.064	3.799	3.416	3.081	2.880
En-BW	-0.33%	+0.1%	+0.11%	+0.26%	+0.49%	+0.76%

2.2.4. Other Models

While most results in this paper are focused on GPT models (the current state of the art for language modeling), the question of the AoT can naturally be asked for other autoregressive models. We investigate this for GRUs and LSTMs (three sizes each), again with a context length of 256.

Once more, for sufficiently large models, we observe a consistent AoT throughout modalities, confirming that the observed AoT goes beyond Transformer models; rather, it appears to be intrinsic to the dataset. It is interesting for instance that for the English dataset, the smaller BW model performs slightly better than the FW one. This however convincingly disappears for larger context sizes and models.

Table 2: Final FW GRU/LSTM losses and relative BW differences.

Size	GRU S 4.92M	GRU M 13.7M	GRU L 22.0M	LSTM S 55.6M	LSTM M 162M	LSTM L 405M
Fr-FW	3.905	3.692	3.363	3.901	3.566	3.314
Fr-BW	+0.26%	+0.3%	+0.62%	+0.1%	+0.45%	+0.66%
En-FW	4.030	3.712	3.483	4.015	3.653	3.418
En-BW	-0.07%	+0.22%	+0.34%	-0.27%	+0.11%	+0.15%

2.2.5. Other Languages

The above experiments confirm the existence of an AoT for English and French across various modalities. An exciting

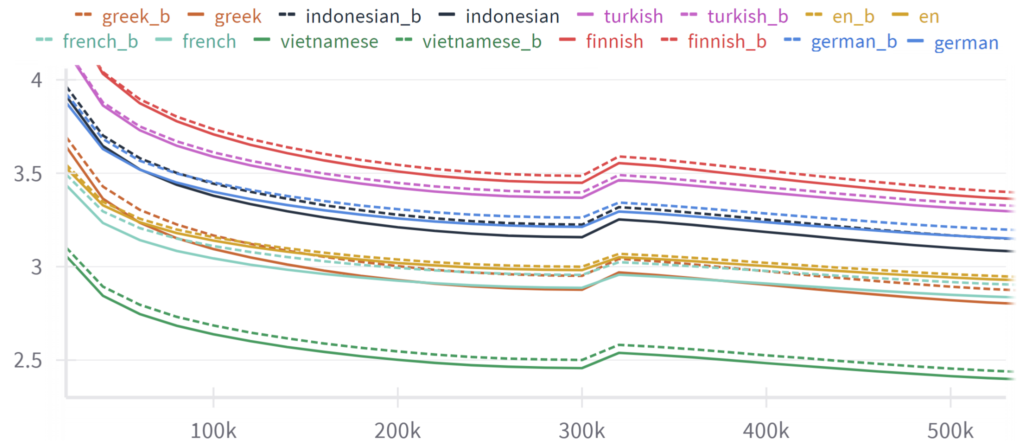

Figure 3: Validation loss curves for FW and BW models during training. Consistently, the BW loss is higher than its FW counterpart. This persists through the warm restart of the learning rate, which causes a bump in the loss.

question that naturally arises is whether this might be a universal property of natural languages. To begin to explore this question, we train models of two sizes (GPT2-Medium and GPT2-XL) on six more languages.

Table 3: Final losses for different languages, Medium/XL models. Format: [Final FW loss]/[BW relative difference].

	German	Turkish	Finnish	Vietnamese
Med	3.148/+1.46%	3.292/+0.94%	3.359/+1.07%	2.396/+1.67%
XL	2.892/+1.59%	3.084/+1.17%	2.975/+1.85%	2.099/+2.81%

	Greek	Indonesian	French	English
Med	2.794/+2.4%	3.079/+2.18%	2.834/+2.4%	2.926/+0.61%
XL	2.494/+3.05%	2.741/+3.17%	2.586/+2.51%	2.683/+0.63%

From Table 3, we can see that in all the cases we tested, a FW AoT emerges, although its magnitude appears to vary from language to language, suggesting some universality of this phenomenon across human languages. Fig. 3 showcases the stability of this AoT during training, across languages. Three more languages (Tagalog, Hebrew and Arab) were tested at the suggestion of reviewers, confirming the universality of the AoT in human languages (see Appendix A.5).

2.2.6. Possible Artifacts

While the training procedures are perfectly symmetric with respect to the two directions, it is important to rule out any other possible sources of asymmetry. One possible source could in principle be the tokenization; indeed, the BPE tokenizer is trained in the FW direction. To rule out this possibility, we inverted (at character level) two datasets (Greek and French), and re-trained a BPE tokenizer on the result. We trained a GPT2-Medium on it, and confirmed that the direction of the BPE tokenization has no effect on the training dynamics: in this case, the FW (respectively BW) model performs very closely to the BW (respectively FW) model on the original tokenization, thereby showing exactly the same AoT. See Appendix A.6 for details.

Additionally, one might ask about the variation in the final losses due to initialization. Although the agreement of all the different experiments show that this is negligible w.r.t. the AoT effect, we quantify this influence by repeating experiments for Greek, see app. A.7.

2.3. Key Takeaways

The above experiments suggest the universality of the phenomenon of AoT across languages, models, and hyperparameters. More specifically:

- A very consistent AoT emerges for large enough models, trained for long enough, and with a large enough context window; in the other cases, the effects are less clear.
- An important finding is that the magnitude of the AoT increases with the context length: this suggests the importance of long-range correlations; relatedly, the model's size can influence its ability to use the information of its whole context window.
- While most of our training is done with GPT models, we observe the same type of results for GRUs and LSTMs, suggesting that AoTs are intrinsic to datasets.
- An interesting phenomenon is that the magnitude of the AoT greatly depends on the language, even if its presence and direction are universal. Explaining this convincingly remains a fascinating challenge.

In Section 3, we introduce a framework to reveal the emergence of the AoT in synthetic datasets and propose mechanisms to explain how this can apply to natural languages. In Section 4, we discuss how these somehow surprising results open the door to many possible investigations.

3. Computability and Irreversibility

As discussed in 1.3 above, from an information-theoretic point of view (abstracting away computability), there should be no difference between FW/BW models. However, as shown in 2.2, we see a consistent AoT for various types of architectures across multiple modalities, which increases with larger context windows. As a result, any plausible explanation must explain why certain probabilities are harder either to be (1) *represented* or (2) *learned* with BW models than with FW ones. Naturally (1) is stronger than (2): models cannot learn what they cannot represent (e.g. if there exists no set of model weights that solve the problem). In this section, we provide simple mathematical models of data illustrating how both mechanisms can arise and naturally contribute to the AoT. We start with a simple mathematical model using prime number multiplications, illustrating how the computational hardness of reversing certain information-preserving operations generates an AoT. We then construct a more general class of data models based on binary operations, allowing one to reveal an AoT based on sparsity and complexity theory ideas.

3.1. Number Factoring and Arrow of Time

Perhaps the most classical example of information-preserving, yet hard to invert, computation is number factoring: given two large primes p, q with $p < q$, it is relatively easy to compute $n = pq$; while n contains the same information as p and q, recovering them from their product is (believed to be) very hard. This problem is the basis of much of asymmetric cryptography. In this section, we study how FW/BW models perform when trained on a dataset based on this idea; we study the theoretical entropy distribution when reading the data FW and BW and compare this to the experimental values for FW/BW GPTs.

3.1.1. Synthetic dataset

For fixed $k \geq 1$, consider the language of strings of the form $p \times q \leftrightarrow \mathrm{rev}\,(pq)$, with $p < q$ primes, $p, q < 10^k$ and $\mathrm{rev}\,(pq)$ being the product of p times q, written in reverse order (see (Shen et al., 2023; Lee et al., 2023)). The numbers p and q are padded to be of k digits exactly and the $\mathrm{rev}\,(pq)$ is padded to be of $2k$ digits exactly (e.g. for $k = 4$: $0019 \times 0023 \leftrightarrow 73400000$). The symbols $\times$ and $\leftrightarrow$ are written as multiple tokens (3 tokens for $\times$, and 7 tokens for $\leftrightarrow$), to facilitate the learning of non-trivial operations by GPTs (Thomas Ahle, 2023). For a fixed k, $\mathbb{P}$ is thus supported on $4k + 10$ token sequences; in our experiments, we set $k = 5$ and take 10^8 such random ordered pairs. Intuitively, computing the right-hand side (RHS) of the symbol $\leftrightarrow$ given the left-hand side (LHS) should be easy, while computing the LHS from the RHS should be much harder (at least finding q; given q and $\mathrm{rev}\,(pq)$, finding p should be easier).

3.1.2. Nats of entropy

In order to better understand the experimental results, we compute the aggregate entropy (in nats) on p, q and $\mathrm{rev}\,(pq)$ when reading the strings $p \times q \leftrightarrow \mathrm{rev}\,(pq)$ FW and BW (we do not compute the entropy on each token individually, and the entropy on the symbols $\times$ and $\leftrightarrow$ is 0). For instance, for $k = 5$, there are $\ln\left(\pi\left(10^5\right)\right) = 9.17$ (with $\pi\,(x) = \#p : p \leq x$) nats of entropy over the possible prime numbers $< 10^5$, which drops to 8.98 nats of entropy on p (because of the ordering), and (averaging over p) 8.67 nats of entropy on q; this results in 17.64 nats of entropy for the pair (p, q), which is roughly $2 \times 9.17 - \ln(2)$ (we subtract the bit of information due to the ordering); since $\mathrm{rev}\,(pq)$ is determined by p and q, its conditional entropy is naturally zero. Reading the string backward, the 17.64 nats of entropy are concentrated on $\mathrm{rev}\,(pq)$; the rest is fully determined, and thus has zero entropy.

3.1.3. Experimental Results

Training a model with a GPT2-Medium on the $p \times q$ dataset yields the log-perplexities recorded in Table 4. The FW model is able to reach the information-theoretical limits on p and q; the conditional cross-entropy loss on $\mathrm{rev}\,(pq)$ is low but non-zero, indicating that the model (imperfectly) learns to multiply the prime numbers. In contrast, the results for the BW model show a far-from-optimal perplexity on $\mathrm{rev}\,(pq)$, which points to the difficulty for the model to recognize the products of two primes; knowing $\mathrm{rev}\,(pq)$, almost no information on the prime factor q is extracted: only $8.98 - 8.41 = 0.57$ nats, i.e. less than one bit. The 'division' is much more learnable, with all but 0.02 nats of information learned. All in all, the total FW log-perplexity is 22.2 nats, while the BW one reaches 30.2 nats.

Table 4: Final perplexities for the prime numbers dataset

	p	q	$rev(pq)$
FW	8.98	8.67	4.55
BW	0.02	8.41	21.56

3.1.4. Discussion

The above setup shows a significant AoT for the $p \times q \leftrightarrow \mathrm{rev}\,(pq)$ dataset. This discrepancy can be largely attributed to the asymmetry between the difficulty of factoring versus multiplication: compared to the information-theoretical limit, about 4.55 nats are lost for the multiplication, while 8.43 nats are lost for the factorization. We also see that the different structures of the LHS and RHS (which have the same information-theoretic content as they determine each other) also present a significant difference w.r.t. the models' abilities: while the FW model reaches essentially optimal perplexity for the LHS (i.e. it recognizes primes $< 10^k$), the BW model is very far from optimal on the RHS (i.e. to recognize products of primes pairs $p, q < 10^k$ proves to be much more difficult). While part of the asymmetry is attributable to the models' specifics, as long as the dataset size is kept high enough that all pairs (p, q) cannot be memorized, a significant AoT can be expected: as the model size (and training time) grows, multiplication will eventually be learned (long before the dataset can be memorized, (Shen et al., 2023)), while extracting substantial information from $\mathrm{rev}\,(pq)$ about q should remain very hard. The above data model displays an AoT of types (1) and (2) (see 3 above): certain features turn out to be harder to learn for the BW models, while others simply are likely not even representable by LLMs (of reasonable sizes), as the alternative would yield an efficient factorization algorithm.

Remark 6. Note also that the above also illustrates the importance of long-range dependences for the AoT: if the context length is kept e.g. significantly below k, the FW model will have trouble saying anything about $\mathrm{rev}\,(pq)$, as p will already be forgotten when reaching the RHS, shrinking its advantage over the BW model.

Remark 7. In the above setting, we are rooted in the computational difficulty of the inversion of a bijective function; note that the core of the argument is the *computational difficulty*, rather than bijectivity. Abstracting computability issues, there is still no difference between FW and BW perplexities for optimal predictors, even if a mapping is not injective, or if it is not well-defined as mapping (e.g. if some random noise is added to it). The invertibility merely helps us get a simple computation of the theoretical entropies, and to pinpoint where each model performs suboptimally; it is however not directly related to the presence of an AoT.

3.2. Binary Operations

The model of Section 3.1 shows how an AoT *can appear* in a dataset: in that example, based on computational complexity ideas, we could handcraft a synthetic dataset that is both practically and theoretically harder for a BW model than a FW one. This still leaves the question of *why* an AoT would arise in a dataset such as natural language, as in Section 2, and why in one direction rather than another, i.e. why FW models would consistently outperform BW ones. In this section, we introduce synthetic datasets based on operations on the space of m-bit registers identified with $\mathbb{F}_2^m$ ($\mathbb{F}_2$ denotes the field of integers mod 2). We focus on languages based on $\mathbb{F}_2$-linear circuits and relate their learnability to their sparsity, using this to explain a difference between FW and BW learnability; we then provide a setup motivating the specific FW direction of the AoT in natural languages; we then provide experiments validating our framework; we conclude by discussing extensions to the nonlinear case.

3.2.1. Linear Sparse Circuit Dynamics

Consider the class of measures $\mathbb{P}_n$ on a *linear language* formed by sequences of $n = 2m + 1$ tokens, of the form $x \leftrightarrow y$ where x, y are random uniform on $\mathbb{F}_2^m$, but related by a bijective linear map; $\leftrightarrow$ is counted as a token. For each $\mathbb{P}_n$, we can write $y = f^{\rightarrow}_{\mathbb{P}_n}(x)$ and $x = f^{\leftarrow}_{\mathbb{P}_n}(y)$ for $f^{\leftrightarrows} : \mathbb{F}_2^m \to \mathbb{F}_2^m$. We define the *sparsity* of a linear map $f : \mathbb{F}_2^m \to \mathbb{F}_2^m$ as the proportion of zero entries of its matrix. We will (informally) say that a matrix is *sparse* if this proportion is relatively high, i.e. close to 1. Intuitively, the sparsity of $f^{\rightarrow}$ (resp. of $f^{\leftarrow}$) is related to how easy it is to learn $\mathbb{P}_n$ (based on random data samples) for a FW model (resp. for a BW model). For GPT predictors, this is studied numerically in Section 3.2.3 below. For a linear language $\mathbb{P}_n$, an AoT will thus emerge if the sparsities of $f^{\rightarrow}$ and $f^{\leftarrow}$ are significantly different. It is common knowledge that the inverse of a sparse matrix is generally less sparse (e.g. (Duff et al., 2017), Section 15.6). This is the basis for the following claim (verified numerically in Appendix B.2.2):

Claim 8. If A is a sparse random $m \times m$ matrix in $\mathbb{F}_2$ conditioned to be invertible, the matrix A^{-1} has typically lower sparsity. Similarly, if we perturb a invertible matrix M by a random sparse matrix A, we have that the corresponding perturbation of the inverse $(M + A)^{-1} - M^{-1}$ is typically less sparse than A.

This claim can be used to show that in natural settings, if we want to condition e.g. on $f^{\rightarrow}$ being sparse, this will result in an $f^{\leftarrow}$ that is comparatively less sparse, and vice versa. In 3.2.2 below, we propose a communication setup where the sparsity of $f^{\rightarrow}$ is naturally favored, yielding a FW AoT as observed in the natural languages (see 2.2 above).

3.2.2. A Simple Communication Setup

In the previous section, we have shown that the emergence of an AoT is natural in the sparse setting: if we e.g. condition $f^{\rightarrow}$ to be sparse, this will yield an inverse $f^{\leftarrow}$ that is less sparse. To motivate the importance of sparsity, and in particular of FW sparsity (for the presence of a FW AoT), we give a simple communication setup.

Suppose Alice and Bob are (human) agents with FW predictors having learned a common language $\mathbb{P}_B$, and Carol is an (alien) agent with a BW predictor having learned $\mathbb{P}_B$ as well. Now suppose Alice wants to teach Bob a new language $\mathbb{P}_A$ by sending him samples from $\mathbb{P}_A$; how easy this is will typically depend on how far away $\mathbb{P}_A$ is from $\mathbb{P}_B$, i.e. how sparse $f^{\rightarrow}_{BA} = f^{\rightarrow}_A - f^{\rightarrow}_B$ is. Assume Alice is only able to teach $\mathbb{P}_A$ to Bob if $f^{\rightarrow}_{BA}$ is sparse enough (note that Alice needs to learn $\mathbb{P}_A$ herself, it is reasonable to assume that she will only be able to do so if $f^{\rightarrow}_{BA}$ is sparse enough). Conditioning on Alice being able to teach $\mathbb{P}_A$ will hence yield (with high probability) a FW AoT: following Claim 8 above, $f^{\leftarrow}_{BA} = f^{\leftarrow}_A - f^{\leftarrow}_B$ will be typically less sparse than $f^{\rightarrow}_{BA}$ and $\mathbb{P}_A$ will be harder for Carol to learn than for Bob. This will ultimately impact the language structure: if e.g. $f^{\leftarrow}_{BA}$ was often sparser than $f^{\rightarrow}_{BA}$, it would be profitable to 'restructure' the language, expressing $y \leftrightarrow x$ rather than

$x \leftrightarrow y$. This suggests that selection pressure may cause languages to evolve to take a form where $f^{\rightarrow}_{BA}$ is often sparser than $f^{\leftarrow}_{BA}$, yielding a consistent FW AoT.

3.2.3. Experimental Results

In this section, we present experimental results supporting the claims of the previous sections. We first consider a dataset made of linear languages with $x, y \in \mathbb{F}_2^{25}$, for different sparsities of $f^{\rightarrow}$. We train a GPT1 model on these datasets (see Appendix B.2) and plot the final losses in Fig. 4, confirming that sparser matrices are easier to learn.

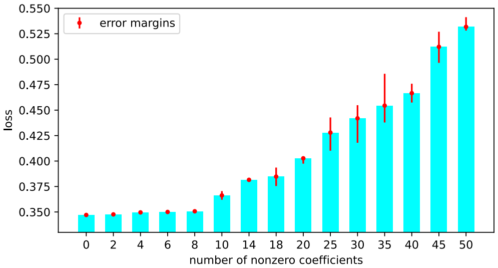

Figure 4: Models loss at the end of training vs $f^{\rightarrow}$ sparsity.

In the second experiment, we consider a model's ability to learn a sparse update, given a learned prior: we first train FW/BW models on a linear language with a $m = 20$ sparse FW matrix, until it is learned perfectly, then generate a new linear language by a sparse perturbation of the matrix. Table 5 shows the losses of both models after 400 gradient descent steps. Again, we see that the FW model adapts better to the sparse modifications (see also Appendix B.2.3).

Table 5: GPT losses for various perturbations of the learned prior.

	2-bit flips	4-bit flips	6-bit flips
FW	0.347 ± 0.009	0.354 ± 0.007	0.367 ± 0.014
BW	0.353 ± 0.008	0.371 ± 0.011	0.387 ± 0.013

3.2.4. Non-Linear Case and Extensions

If we consider a more general model of languages compared to Section 3.2.1, based on arbitrary functions $f : \mathbb{F}_2^m \to \mathbb{F}_2^m$, the notion of sparse map needs to be adapted to consider sparse *circuits*, made of relatively few logic gates (AND/OR/XOR) in the linear circuits. We can expect that for a random sparse circuit model $f^{\rightarrow}$, the inverse (or any pre-image computation) will typically be much less sparse. This is suggested by the fact that inverting circuits is expected to be computationally hard (roughly the content of the $P \neq NP$ conjecture). We can hence expect an AoT for similar reasons. Due to the computational hardness of inverting nonlinear circuits, a difference with the linear case can be expected to arise: the nature of the AoT in this case can also be of Type 1 (see Section 3); in some cases, a BW model may simply be unable to represent what the FW model learns, as in the example of Section 3.1.

4. Discussion

In this paper, we have investigated the abilities of autoregressive LLMs, which predict tokens sequentially: for a given measure (dataset), we compare the abilities of two models (FW/BW). Theoretically, if both models learn to represent the same measure, their average log-perplexities should coincide. We discover the existence of an *Arrow of Time (AoT)* for natural language datasets: across a wide variety of models and hyperparameters, all else being equal, FW models exhibit a consistently *lower perplexity* than BW ones; this difference emerges as soon as the model is large enough; its causes appear rooted in long-range correlations in the data, as the effect magnitude increases with the context length. We propose a framework to explain this phenomenon, based on complexity and sparsity ideas: we construct examples of synthetic datasets based on operations that display an asymmetry in terms of computability (despite being information-theoretically reversible); finally, we propose a setup where a FW AoT like the one seen in natural language can spontaneously emerge. Our work suggests a number of possible future research directions:

- Are AoTs universal across all human languages?
- Are there AoTs in other types of languages, e.g. computer code, binaries, DNA code, or bitmap files?
- How to explain the variation in magnitude of the AoT across languages?
- Are there AoT scaling laws with respect to model sizes?
- Are natural language AoTs of Type 1 or 2 (in the sense of Section 3)?
- For very long training times, is there a difference between train and test AoTs?
- Can AoTs, Causality, and $\mathcal{V}$-Information (see Section 1.6) be understood under a common framework?
- What about AoTs in continuous settings, e.g. for video?
- Is there a link with other AoTs, e.g. in thermodynamics?
- Is the presence of an AoT in data a sign of life or intelligent processing?
- Can we generalize the idea of flipping the order of the tokens to other permutations of the context window?
- Are AoT and computational hardness deeply linked?

In conclusion, the concept of AoT appears to be related to subtle properties of natural language data, revealed through their interplay with autoregressive LLMs. This idea's applicability seems wide, and a promising new tool to reveal the presence of deep structural features in data. Further study of its theoretical origin could prove fruitful towards uncovering links between AoTs and complexity theory.

Acknowledgements

The authors would like to thank Stéphane d'Ascoli, Samy Bengio, Gloria Capano, Diego Dorn, Franck Gabriel, Ron Maimon, Jacob Menick, Christos Papadimitriou, João Penedones, Matthieu Wyart, Nicolas Zlatoff, as well as the anonymous reviewers for interesting discussions and suggestions. Support from the Blavatnik Family Foundation, the Latsis Foundation, the NCCR SwissMAP, and from an EPFL FSB Seed Funding Grant are gratefully acknowledged.

Impact Statement

This paper presents work with the goal of advancing the field of Machine Learning, and our scientific understanding of language models. There are many potential societal consequences of a greater understanding of this discipline, and thus, indirectly, of our work, however, none of them feel direct enough to be specifically highlighted here.

References

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, May 2020. URL http://arxiv.org/abs/2005.14165. arXiv: 2005.14165. 1

Chan, W., Kitaev, N., Guu, K., Stern, M., and Uszkoreit, J. KERMIT: Generative Insertion-Based Modeling for Sequences, June 2019a. URL http://arxiv.org/abs/1906.01604. arXiv:1906.01604 [cs, stat]. 1.5

Chan, W., Stern, M., Kiros, J., and Uszkoreit, J. An Empirical Study of Generation Order for Machine Translation, October 2019b. URL http://arxiv.org/abs/1910.13437. arXiv:1910.13437 [cs]. 1.5

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., GuzmÃ¡n, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. Unsupervised Cross-lingual Representation Learning at Scale, April 2020. URL http://arxiv.org/abs/1911.02116. arXiv:1911.02116 [cs]. 2.1.1

d'Ascoli, S., Bengio, S., Susskind, J., and AbbÃ©, E. Boolformer: Symbolic Regression of Logic Functions with Transformers, September 2023. URL http://arxiv.org/abs/2309.12207. arXiv:2309.12207 [cs]. 1.1

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv:1810.04805 [cs]. 1, 1.5

Duchateau, J., Demuynck, K., and Wambacq, P. Confidence scoring based on backward language models. In *ICASSP*, pp. 221–224, 2002. 1.5

Duff, I. S., Erisman, A. M., and Reid, J. K. *Direct Methods for Sparse Matrices*. Oxford University Press, January 2017. ISBN 978-0-19-850838-0. doi: 10.1093/acprof:oso/9780198508380.001.0001. URL https://doi.org/10.1093/acprof:oso/9780198508380.001.0001. 3.2.1

Emelianenko, D., Voita, E., and Serdyukov, P. Sequence Modeling with Unconstrained Generation Order, October 2019. URL http://arxiv.org/abs/1911.00176. arXiv:1911.00176 [cs]. 1.5

Ford, N., Duckworth, D., Norouzi, M., and Dahl, G. The Importance of Generation Order in Language Modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2942–2946, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1324. URL http://aclweb.org/anthology/D18-1324. 1.5

Gneiting, T. and Raftery, A. E. Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477): 359–378, March 2007. ISSN 0162-1459, 1537-274X. doi: 10.1198/016214506000001437. URL http://www.tandfonline.com/doi/abs/10.1198/016214506000001437. 1.3

Graves, A., Srivastava, R. K., Atkinson, T., and Gomez, F. Bayesian Flow Networks, August 2023. URL http://arxiv.org/abs/2308.07037. arXiv:2308.07037 [cs]. 1.1

Gu, J., Bradbury, J., Xiong, C., Li, V. O. K., and Socher, R. Non-Autoregressive Neural Machine Translation, March 2018. URL http://arxiv.org/abs/1711.02281. arXiv:1711.02281 [cs]. 1.5

Gu, J., Liu, Q., and Cho, K. Insertion-based Decoding with automatically Inferred Generation Order, October 2019a. URL http://arxiv.org/abs/1902.01370. arXiv:1902.01370 [cs]. 1.5

Gu, J., Wang, C., and Zhao, J. Levenshtein Transformer, October 2019b. URL http://arxiv.org/abs/1905.11006. arXiv:1905.11006 [cs]. 1.5

Hanson, R. LOGARITHMIC MARKETS CORING RULES FOR MODULAR COMBINATORIAL INFORMATION AGGREGATION. *The Journal of Prediction Markets*, 1(1):3–15, December 2012. ISSN

1750-676X, 1750-6751. doi: 10.5750/jpm.v1i1.417. URL http://www.bjll.org/index.php/jpm/article/view/417. 1.3

He, K., Zhang, X., Ren, S., and Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, February 2015. URL http://arxiv.org/abs/1502.01852. arXiv:1502.01852 [cs] version: 1. 2.1

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., Driessche, G. v. d., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training Compute-Optimal Large Language Models. *arXiv:2203.15556 [cs]*, March 2022. URL http://arxiv.org/abs/2203.15556. arXiv: 2203.15556. 1.1

Howard, J. and Ruder, S. Universal Language Model Fine-tuning for Text Classification, May 2018. URL http://arxiv.org/abs/1801.06146. arXiv:1801.06146 [cs, stat]. 1.5

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling Laws for Neural Language Models. *arXiv:2001.08361 [cs, stat]*, January 2020. URL http://arxiv.org/abs/2001.08361. arXiv: 2001.08361. 1.1

Karpathy, A. karpathy/minGPT, May 2023. URL https://github.com/karpathy/minGPT. original-date: 2020-08-17T07:08:48Z. 2.1.2

Lee, J., Mansimov, E., and Cho, K. Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement, August 2018. URL http://arxiv.org/abs/1802.06901. arXiv:1802.06901 [cs, stat]. 1.5

Lee, N., Sreenivasan, K., Lee, J. D., Lee, K., and Papailiopoulos, D. Teaching Arithmetic to Small Transformers, July 2023. URL http://arxiv.org/abs/2307.03381. arXiv:2307.03381 [cs]. 3.1.1

Liu, L., Utiyama, M., Finch, A., and Sumita, E. Agreement on Target-bidirectional Neural Machine Translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 411–416, San Diego, California, 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1046. URL http://aclweb.org/anthology/N16-1046. 1.5

Loshchilov, I. and Hutter, F. SGDR: Stochastic Gradient Descent with Warm Restarts, May 2017. URL http://arxiv.org/abs/1608.03983. arXiv:1608.03983 [cs, math]. 2.1.2

Loshchilov, I. and Hutter, F. Decoupled Weight Decay Regularization, January 2019. URL http://arxiv.org/abs/1711.05101. arXiv:1711.05101 [cs, math]. 2.1.2

Mangal, S., Joshi, P., and Modak, R. LSTM vs. GRU vs. Bidirectional RNN for script generation. August 2019. 1.5

Mansimov, E., Wang, A., Welleck, S., and Cho, K. A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, February 2020. URL http://arxiv.org/abs/1905.12790. arXiv:1905.12790 [cs, stat]. 1.5

Mou, L., Yan, R., Li, G., Zhang, L., and Jin, Z. Backward and Forward Language Modeling for Constrained Sentence Generation, January 2016. URL http://arxiv.org/abs/1512.06612. arXiv:1512.06612 [cs]. 1.5

Nguyen, A., Karampatziakis, N., and Chen, W. Meet in the Middle: A New Pre-training Paradigm, March 2023. URL http://arxiv.org/abs/2303.07295. arXiv:2303.07295 [cs]. 1.5

Oord, A. v. d., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G. v. d., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. Parallel WaveNet: Fast High-Fidelity Speech Synthesis, November 2017. URL http://arxiv.org/abs/1711.10433. arXiv:1711.10433 [cs]. 1.5

OpenAI. GPT-4 Technical Report, March 2023. URL http://arxiv.org/abs/2303.08774. arXiv:2303.08774 [cs]. 1

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, pp. 311, Philadelphia, Pennsylvania, 2001. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL http://portal.acm.org/citation.cfm?doid=1073083.1073135. 1.5

Peters, J., Janzing, D., and SchÃ¶lkopf, B. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017. ISBN 978-0-262-03731-0 978-0-262-34429-6. URL https://library.oapen.

org/handle/20.500.12657/26040. Accepted: 2019-01-20 23:42:51. 1.6

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations, March 2018. URL http://arxiv.org/abs/1802.05365. arXiv:1802.05365 [cs]. 1.5

Pfau, J., Infanger, A., Sheshadri, A., Panda, A., Huebner, C., and Michael, J. Eliciting Language Model Behaviors using Reverse Language Models. October 2023. 1.5

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving Language Understanding by Generative Pre-Training. pp. 12, July 2018. 1, 2.1.2

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language Models are Unsupervised Multitask Learners. pp. 24, February 2019. 1, 2.1.1

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, September 2023. URL http://arxiv.org/abs/1910.10683. arXiv:1910.10683 [cs, stat]. 1.5

Savage, L. J. Elicitation of Personal Probabilities and Expectations. *Journal of the American Statistical Association*, 66(336):783–801, December 1971. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.1971.10482346. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482346. 1.3

Savinov, N., Chung, J., Binkowski, M., Elsen, E., and Oord, A. v. d. Step-unrolled Denoising Autoencoders for Text Generation, April 2022. URL http://arxiv.org/abs/2112.06749. arXiv:2112.06749 [cs]. 1.5

Schaeffer, R., Miranda, B., and Koyejo, S. Are Emergent Abilities of Large Language Models a Mirage?, May 2023. URL http://arxiv.org/abs/2304.15004. arXiv:2304.15004 [cs]. 1

Scholkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., and Bengio, Y. Towards Causal Representation Learning, February 2021. URL http://arxiv.org/abs/2102.11107. arXiv:2102.11107 [cs]. 1.6

Sennrich, R., Haddow, B., and Birch, A. Neural Machine Translation of Rare Words with Subword Units, June 2016. URL http://arxiv.org/abs/1508.07909. arXiv:1508.07909 [cs]. 2.1.1

Serdyuk, D., Ke, N. R., Sordoni, A., Trischler, A., Pal, C., and Bengio, Y. Twin Networks: Matching the Future for Sequence Generation, February 2018. URL http://arxiv.org/abs/1708.06742. arXiv:1708.06742 [cs, stat]. 1.5

Shannon, C. E. Prediction and entropy of printed english. *Bell Systems Technical Journal*, pp. 50–64, 1951. (document), 1.5

Shen, R., Bubeck, S., Eldan, R., Lee, Y. T., Li, Y., and Zhang, Y. Positional Description Matters for Transformers Arithmetic, November 2023. URL http://arxiv.org/abs/2311.14737. arXiv:2311.14737 [cs]. 1.1, 1.5, 3.1.1, 3.1.4

Stern, M., Chan, W., Kiros, J., and Uszkoreit, J. Insertion Transformer: Flexible Sequence Generation via Insertion Operations, February 2019. URL http://arxiv.org/abs/1902.03249. arXiv:1902.03249 [cs, stat]. 1.5

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [cs]*, September 2014. URL http://arxiv.org/abs/1409.3215. arXiv: 1409.3215. 1.5

Thomas Ahle. This week I trained an 800K transformer to learn 5 digit multiplication., September 2023. URL https://twitter.com/thomasahle/status/1702723749798354976. 3.1.1

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention Is All You Need. *arXiv:1706.03762 [cs]*, June 2017. URL http://arxiv.org/abs/1706.03762. arXiv: 1706.03762. 1

Vinyals, O., Bengio, S., and Kudlur, M. Order Matters: Sequence to sequence for sets, February 2016. URL http://arxiv.org/abs/1511.06391. arXiv:1511.06391 [cs, stat]. 1.5

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent Abilities of Large Language Models, June 2022. URL http://arxiv.org/abs/2206.07682. Number: arXiv:2206.07682 arXiv:2206.07682 [cs]. 1

Welleck, S., Brantley, K., DaumÃ© III, H., and Cho, K. Non-Monotonic Sequential Text Generation, October 2019. URL http://arxiv.org/abs/1902.02192. arXiv:1902.02192 [cs, stat]. 1.5

Wenzek, G., Lachaux, M.-A., Conneau, A., Chaudhary, V., GuzmÃ¡n, F., Joulin, A., and Grave, E. CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data, November 2019. URL http://arxiv.org/abs/1911.00359. arXiv:1911.00359 [cs, stat]. 2.1.1

Wu, L., Tan, X., He, D., Tian, F., Qin, T., Lai, J., and Liu, T.-Y. Beyond Error Propagation in Neural Machine Translation: Characteristics of Language Also Matter. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3602–3611, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1396. URL http://aclweb.org/anthology/D18-1396. 1.5

Xiao, Y., Wu, L., Guo, J., Li, J., Zhang, M., Qin, T., and Liu, T.-y. A Survey on Non-Autoregressive Generation for Neural Machine Translation and Beyond, July 2023. URL http://arxiv.org/abs/2204.09269. arXiv:2204.09269 [cs]. 1.5

Xu, Y., Zhao, S., Song, J., Stewart, R., and Ermon, S. A Theory of Usable Information Under Computational Constraints, February 2020. URL http://arxiv.org/abs/2002.10689. arXiv:2002.10689 [cs, stat]. 1.6

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. XLNet: Generalized Autoregressive Pretraining for Language Understanding, January 2020. URL http://arxiv.org/abs/1906.08237. arXiv:1906.08237 [cs]. 1.5

Zhang, Z., Wu, S., Liu, S., Li, M., Zhou, M., and Xu, T. Regularizing Neural Machine Translation by Target-bidirectional Agreement, November 2018. URL http://arxiv.org/abs/1808.04064. arXiv:1808.04064 [cs]. 1.5

A. Details on training on natural languages

In this appendix, we provide more details on the training of our models on natural languages.

For any experiment, the precise hyperparameters used can be found in the code repository found at github.com/frotaur/ICMLBackPerp, under the folder 'Training Parameters', in $.json$ format. Those files can also be used to reproduce any experiment using the codebase, as explained in the README.md of the repository The branch 'main' contains the code used to generate the experiments of the paper, while the 'Rebutt' branch contains updated, cleaner code, as well as the necessary data to run the extra experiments on Tagalog, Hebrew and Arabic.

All experiments (save for the 512 context size) were run on a single A100 GPU, adjusting the batch size to fit the available memory.

Concerning the shuffling of the dataset, we proceed for all the experiments as follows: we begin by splitting the textual dataset into 'sentences' of the appropriate context size n, with a stride of $n/2$ (i.e., if we have a context size of 4 and the text is *ABCDEF*, this results in two sentences, *ABCD* and *CDEF*). This is to ensure that all tokens appear in the training data with at least some context. After that, we shuffle the obtained sentences with a set seed. We withhold 250k sentences (1000 batches at batch size 250) for validation. The inversion of the tokens is made at the level of each batch; in this way, when training, the FW/BW models see the data in the same order, preventing the emergence of undesirable differences.

A.1. Model sizes

Table 6 provides more detail on the model sizes used in the paper.

Table 6: Model sizes used throughout the experiments. d_{embed}: number of embedding dimensions. n_{heads}: number of attention heads. n_{layers}: number of transformer blocks (attention + MLP). *parameters*: total number of parameters, including the last linear layer which projects on vocabulary size (commonly referred to as the 'head').

GPT2 model name →	Nano	Micro	Mini	Small	GPT1	Medium	XL
d_{embed}	48	128	192	380	768	1024	1600
n_{heads}	3	4	6	10	12	16	25
n_{layers}	3	4	6	10	12	24	48
parameters	4.92M	13.7M	22.0M	55.6M	162M	405M	1.6B

In the MLP layer, all models have one hidden layer with $4 * d_{embed}$ hidden dimensions, a.k.a. an MLP ratio of 4.

A.2. Different languages

A.3. Context Window size

For the testing of the influence of the context window length, we use a GPT2-Medium model, with context window lengths going from 16 to 512 tokens. Because of the different context lengths, it will take a model with a small context length many more gradient steps to see the same amount of data. For this reason, we do not train all models up to the equivalent of 1 epoch of the French dataset ($\sim 26.6B$ tokens), but rather train them for sufficiently long so that the perplexity differences stabilize, and that their losses converge. Due to the cosine annealing learning rate schedule, we stop the training at the end of a cosine decay, to avoid the bump in the loss caused by a warm restart (note however that this has little to no effect on the perplexity differences).

In Table 7, we record the number of steps (i.e. minibatches) that were seen for each context length.

Table 7: Number of batches seen during training for the different context lengths. Note that the recorded batch size is the 'effective' one, that is, potentially obtained through aggregation of smaller batch sizes.

Context length →	16	32	64	128	256	512
batch size	500	500	500	400	180	156
seen batches	1.74M	1.62M	1.64M	1.02M	0.5M	0.6M

For this experiment, because of our memory limitations, we trimmed down the English dataset, which was too big when working with context windows of lengths 16 and 32. Note that the 256 context length experiment in this section is thus slightly different from the one recorded in Table 3, due to the different datasets (as well as the batch size and the cosine annealing period).

A.4. Other Models

The GRU and LSTM implementations we used for these experiments are those natively implemented in the `pytorch.nn` module of the Pytorch Python library. In each case, we choose the 'input size' (i.e., the number of embedding dimensions for each token) to be equal to the 'hidden size' (i.e., the number of hidden dimensions in the RNN's hidden state).

Table 8: Parameters for different sizes of LSTM and GRU models.

	Small	Medium	Large
hidden$_{dim}$	256	512	768
n$_{layers}$	1	3	5

Although RNNs have technically no limit on the context length, for training purposes, to allow for a backward pass, we feed them with batches of texts of lengths 256.

A.5. Other Languages

For the experiments on other languages, we decided to stop the training at the equivalent of 1 epoch of Greek, which was one of the smallest datasets on which we were training (along with Turkish). This choice was maintained for all languages, and models never saw the same datapoint more than once (note that we also tried training on Greek for two epochs, as shown in Fig. 5; this suggests that our results remain valid beyond one epoch).

For GPT2-Medium, a batch size of 90 was used[1], with one warm restart during training. For the XL models, a batch size of 26 was used, aggregated 6 times for an effective batch size of 156, which didn't allow for a warm restart before the one epoch of Greek.

We also tested (using GPT2-Medium, with an adjusted learning rate schedule, due to the smaller size of the datasets) Hebrew, Arabic and Tagalog (see Table 9 for final losses), at the suggestions of anonymous reviewers.

Hebrew and Arabic constituted an example of languages written right-to-left; a priori, we would not expect this to affect the emergence of an AoT: after all, tokens are still processed in the 'spoken' order by the model, so the writing direction does not affect training. Still, there could have been an influence on the language itself from the writing direction, which cannot be detected from our results.

Tagalog is an example of a language with 'verb-initial word order', a relatively rare class of languages, which was not included in our list. Here, our expectations are again confirmed as an AoT appears also for this example. This reinforces the idea that the AoT for natural language emerges from long-range correlations. The specifics of the grammar and the order of words in a sentence are therefore not that important.

Table 9: Final losses for extra languages. Arabic and Hebrew are at 1 epoch of training, and Tagalog at 7 epochs, due to the small size of the dataset. Format: [Final FW loss]/[BW relative difference].

	Tagalog	Arabic	Hebrew
Med	2.368/+1.48%	3.446/+1.91%	3.288/+2.37%

For completeness, we also attempted to run the training for Greek for 2 epochs, to see if memorization of the dataset may affect the Arrow of Time. In Fig. 5, we display the validation loss during training. Comparing the difference in performance at 1 and 2 epochs, it remains almost exactly the same. It would be interesting to test this with bigger models, and several epochs of training.

[1] Due to an oversight in the code, the batches were not aggregated in groups of 2 as expected, but the loss was still renormalized by dividing it by 2. This amounts to a very slight change in learning dynamics, but does not affect any of the results. Similarly, all graphs/reported results display the correct loss. In case one wants to reproduce exactly the results of the paper, the loss should be divided by 2 before backpropagation.

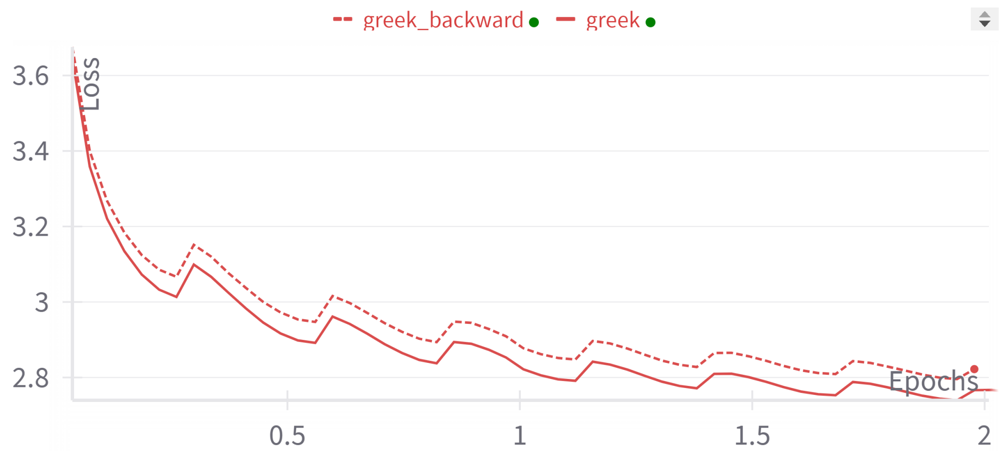

Figure 5: Validation loss for two epochs of training on the greek dataset, for forward and backward models.

A.6. BPE Tokenization

For the tokenization, we use the Huggingface implementation of the BPE Tokenizer (link), using the method `Tokenizer.train_from_iterator`. The tokenizers are trained on the same CC-100 dataset on which we train the model.

To exclude potential tokenization asymmetries (see section 2.2.6), we perform extra experiments in which we train the BPE tokenizer in reverse. To do so, we reverse the language dataset at the character level (not at the byte level, as this would make the output of the model unreadable because of multi-byte characters in utf-8), then train the BPE tokenizer on this new dataset. We then train a FW and a BW model on this character-flipped dataset, tokenized with the new BPE tokenizer.

To make things clearer, we will call a model ‘backward’ (BW) if it processes tokens in the opposite order w.r.t. the natural reading direction (hence the ‘previous-token predictor’ on the ‘character-flipped dataset’ corresponds to what we will call the FW model). Thus, if the Arrow of Time is a property of the language and not a tokenization artifact (as we expect), we expect the arrow of time to be in the same direction, independently of the tokenization scheme. Fig. 6 confirms this; relying on the reverse BPE tokenization introduces very slight differences in the losses. This difference is negligible compared to the AoT effect in both Greek and French, so we can conclude that the AoT is not a tokenization artifact.

In figures 6 and 7, we display the loss curves during training for the french and greek models, trained using both the normal and reversed BPE tokenizers. The tokenizer switch has minimal impact on the losses, and most importantly, it does not affect the AoT direction.

A.7. Initialization variance

The AoT computed in our experiments were obtained with a single training run. One might ask if such a difference remains significant compared to variations in loss due to initialization. The consistency throughout experiments shows that the AoT is significant, but in this section we set to verify the magnitude of the initialization variance. Due to computational costs, it is not possible to obtain error bars for all the experiments. Instead, we focus on Greek, for which we re-run the GPT2-Medium training 4 times in total. Computing the error bars, we obtain a loss (at one epoch) of 2.802 ± 0.005 FW, and 2.871 ± 0.003 BW. This gives us an AoT magnitude of $2.46\% \pm 0.18\%$, where we can see that the variations due to initialization are negligible w.r.t. the magnitude of the AoT.

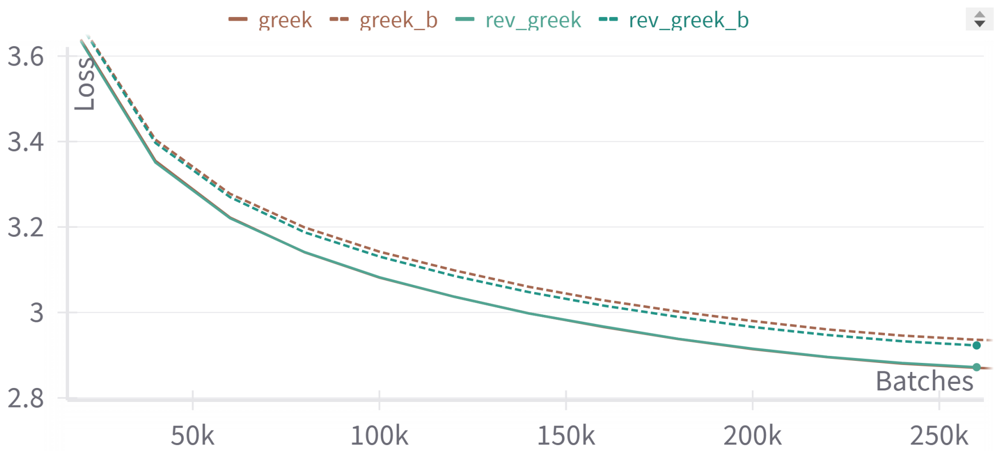

Figure 6: Validation curves for models training on the Greek dataset on forward and backward (label prefixed with 'rev') BPE tokenizations. The Arrow of Time remains the same in the FW direction, despite the different tokenization schemes.

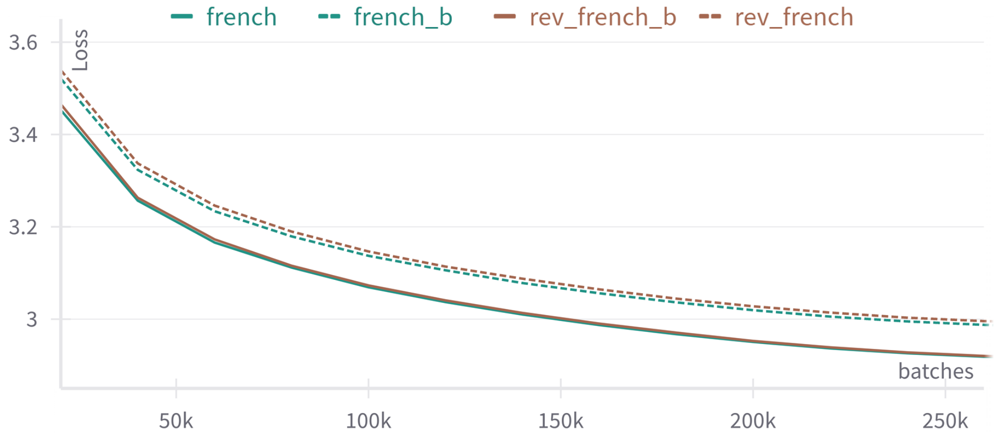

Figure 7: Validation curves for models training on the french dataset on forward and backward (label prefixed with 'rev') BPE tokenizations. The Arrow of Time remains the same in the FW direction, despite the different tokenization schemes. It seems the reverse tokenization is slightly suboptimal, slightly degrading the losses of both FW and BW models.

B. Linear Language Toy Model

B.1. Matrix Inverse Sparsity

Here, we substantiate the assertions of Claim 8 by running numerical experiments. To this end, we wish to look at $n \times n$ matrices in $\mathbb{F}_2$ with a given number of non-zero elements (which we will call k), and compute k in the inverse. To generate invertible matrices with extremely high sparsity (low k), we proceed as follows. We start with the identity matrix Id, which is the only invertible matrix (up to permutations, which do not affect sparsity) when $k = n$. To generate a matrix with approximately k non-zero elements, we flip $k - n$ elements of Id at random. This allows us to often obtain invertible matrices when k is close to n, which would not be the case if we simply selected the k non-zero elements at random. When k becomes bigger than n, the initial presence of the identity is quickly erased.

We choose $n = 30$, generate matrices with $k = [30, 250]$, and record the average number of non-zero elements in the inverse, given k fixed. Fig. 8 clearly confirms that the sparsity of the inverse is generally much lower.

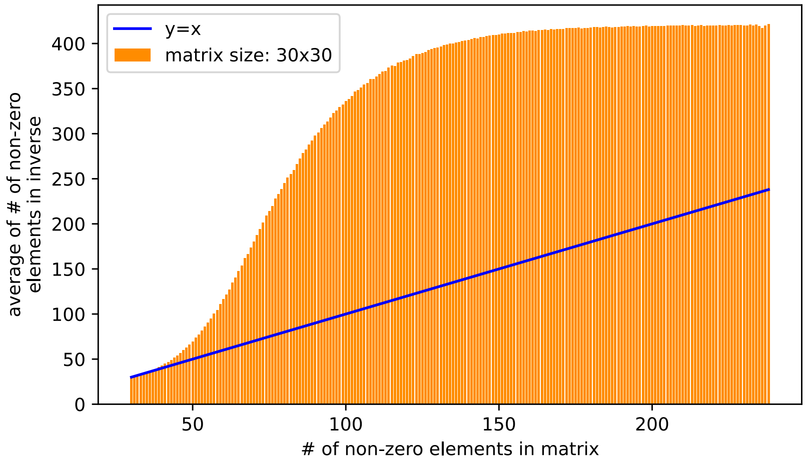

Figure 8: Plot displaying the connection between the sparsity of a matrix and its inverse. It is clear that on average, the inverse of a sparse matrix is less sparse. The number of non-zero elements for the inverse caps at 450.

B.2. Linear Languages Experiments

In this section, we give more details on the experiments of Section 3.2.3.

B.2.1. Linear Language Dataset

Given a matrix M of size $n \times n$, the associated linear language dataset will contain sentences of $2n + 7$ tokens in the form $x\_\_\_\_\_\_\_y$, where $x, y \in \mathbb{F}_2^n$, and the underscores are added as padding, providing the model with more tokens if needed to perform more complex computations. The vector x is drawn at random, and y is computed with $y = Mx$. We finish by randomly flipping each bit with probability $p = 0.01$ (which we call 'adding' perturbations), aimed at smoothing out the probabilities output by the model. This is necessary for the 'fine-tuning' experiments, as otherwise the models become too confident in their predictions, and any small change M results in a huge change in the loss, leading to a catastrophic forgetting of the learned prior.

B.2.2. Sparsity Levels

In the first experiment, we generate a linear language model in $\mathbb{F}_2^{25}$, with $p = 0$ perturbations, and matrices with a number $25 + k$ of non-zero elements, where $k \in [0, 2, 4, 8, 10, 14, 18, 20, 25, 30, 35, 40, 45, 50]$. We then train a transformer model of size GPT1 (see Table 6) on $600k$ sentences, with batch size 200. Note that the context size of the model matches exactly the number of tokens in one sentence. Final losses are reported in 4. Note that for lower sparsities, the trend is not obvious: this is due to the high variance in the final learning rate, as the learning of only a few non-zero elements is binary, depending on the initialization, the model either learns the matrix perfectly quickly, or it usually struggles to find the last few non-zero elements. Fig. 9 displays this behavior in the case $k = 4$. Note that the perturbations somewhat reduce this behavior, but don't cancel it completely.

Fig. 10 displays typical learning dynamics for this problem, for $k = 8$ (high sparsity) and $k = 40$ (medium sparsity). We remind that in principle, given a large enough model, and enough training steps, the model should be able to find the optimal solution (hardness of type (2), see Section 3).

B.2.3. Sparse updates

In the second experiment, we choose a linear language in $\mathbb{F}_2^{20}$. We generate the dataset in the same way explained in Section B.2.1. We begin by training a FW model and a BW one on this language, until both models learn it almost perfectly (batch size 200). For this reason, we choose a very sparse matrix, with $k = 6$ (in this specific example, the inverse has $k = 10$). As expected, this takes much longer for the BW model, as displayed in Fig. 11.

Once this 'prior' is learned, we generate perturbation of the learned matrix by flipping e entries of the matrix randomly,

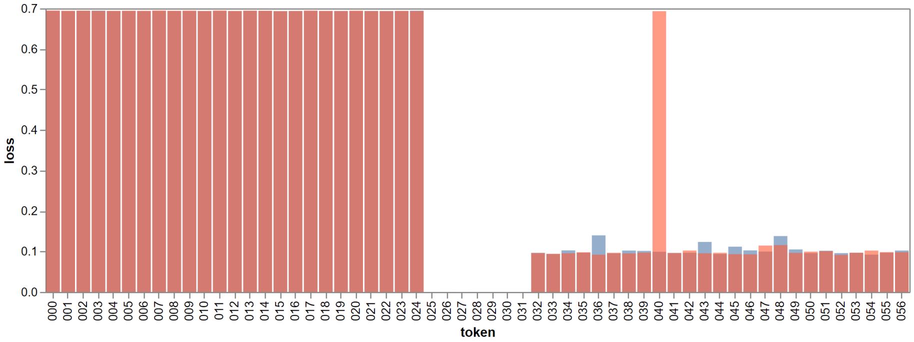

Figure 9: Average perplexities for each token in the linear language after $600k$ sentences, for two runs with $k = 4$. One of the models is very close to the optimal solution, while the other is missing a single token. It usually takes a long time for the model to correct this, leading to higher variance in the final losses.

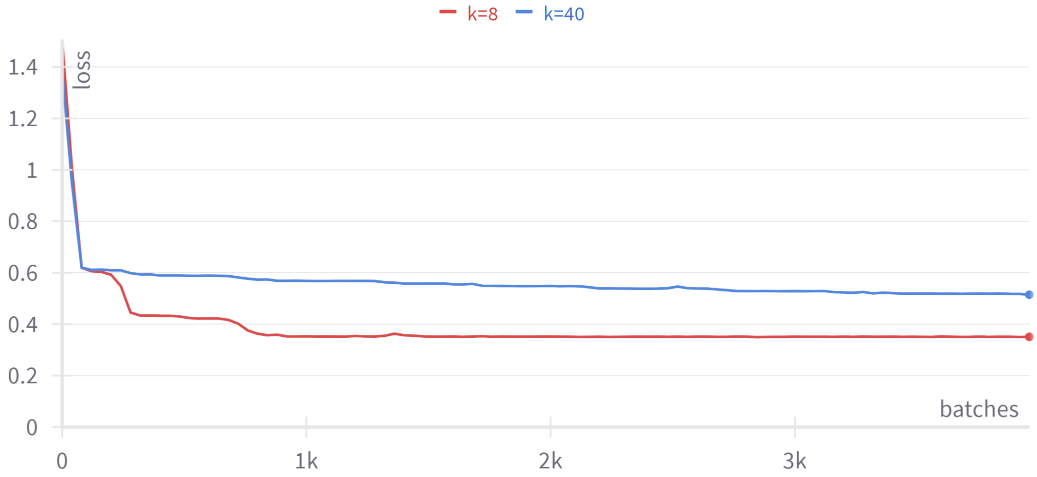

Figure 10: Loss during learning for $k = 8$ and $k = 40$ sparsities. The first plateau simply arises when the model learns to guess all coefficients randomly. Subsequently, the $k = 8$ experiences plateaus each time it learns more non-zero coefficients. The learning of $k = 40$ is much smoother, as discovering non-zero coefficients doesn't lead to perfect predictions right away.

conditioned to the fact that it should remain invertible. We then train the models further on this new dataset, for a relatively small amount of steps (400 gradient steps). We also lower the learning rate to 8×10^{-6}, with 10 steps of warmup, again to prevent catastrophic forgetting of the prior. The training dynamics are displayed in Fig. 12, in the case $e = 4$.

We observe that the FW model adapts better than the BW one, and this is due to the fact mentioned in Claim 8, namely that a sparse FW update will generically result in a less sparse BW update.

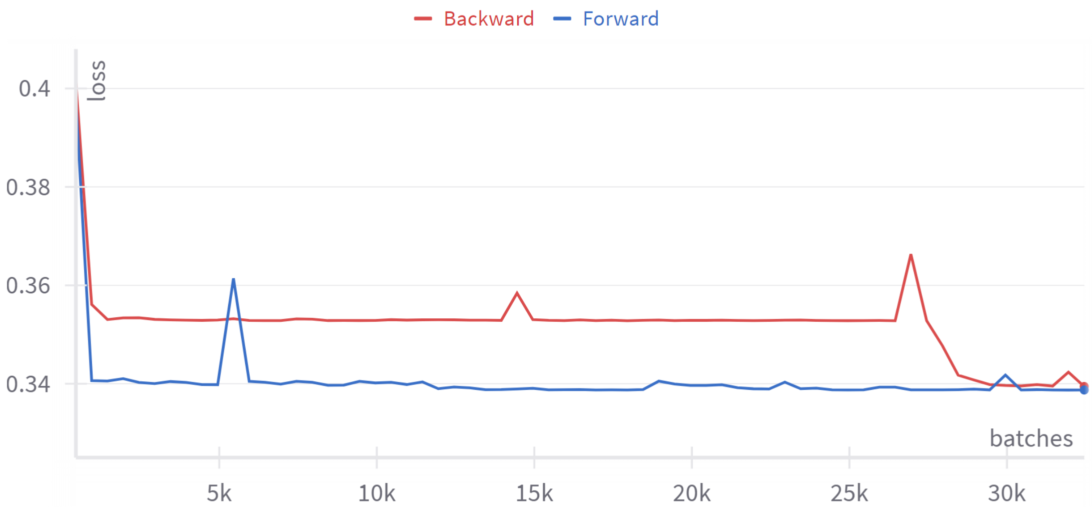

Figure 11: Training loss for the FW and BW models, trying to learn a linear language with a matrix $k = 6$. While the FW model learns the quasi-optimal solution very quickly, the BW model remains stuck on a plateau for a long time. This in fact corresponds to a single element of the predicted vector which was missing, as in Fig. 9.

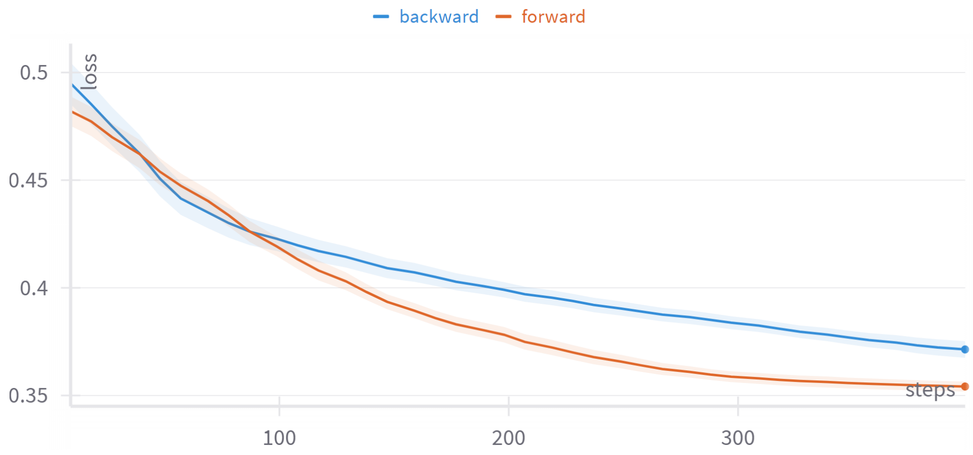

Figure 12: Averaged loss for forward and backward models, when trying to learn a sparse forward perturbation of the Linear language. In the first ~ 100 steps, the curves are similar as both models decrease their confidence in the new, perturbed tokens, setting them back to random chance. Subsequently, they begin learning the perturbation, where the forward model is clearly at an advantage.