



## UNIVERSITÉ SORBONNE UNIVERSITÉ

ECOLE DOCTORALE INFORMATIQUE, TÉLÉCOMMUNICATIONS ET ELECTRONIQUE - ED130

INRIA DE PARIS / ÉQUIPE ALMANACH

### THÈSE DE DOCTORAT

Discipline : Informatique

Présentée par

**Nathan GODEY**

Pour obtenir le grade universitaire de

DOCTEUR de l'UNIVERSITÉ SORBONNE UNIVERSITÉ

---

### Improving Representations for Language Modeling

---

Présentée et soutenue publiquement le DATE devant le jury composé de :

Claire GARDENT	CNRS / LORIA	Présidente
Antske FOKKENS	Vrije Universiteit Amsterdam	Rapportrice & Examinatrice
Ivan TITOV	University of Edinburgh / UvA	Rapporteur & Examinateur
Caio CORRO	INSA Rennes	Examinateur
Louis MARTIN	Mistral AI	Examinateur
Benoît SAGOT	Inria - ALMAnaCH	Directeur
Éric DE LA CLERGERIE	Inria - ALMAnaCH	Co-encadrant



## ABSTRACT

The field of Natural Language Processing has recently known a major paradigm shift that has led to significant improvements over the perceived capabilities of resulting systems. This shift, namely the advent of generative systems in the stead of predictive ones, has induced a profound change in the implicit objectives of language systems based on deep learning : where we used to aim at extracting relevant features from text utterances using self-supervision, we now try to maximize the generative performance of language models on tremendous volumes of diversified text samples.

In this thesis, we explore high-level properties of the features (or *representations*) that are extracted by these language models, and we leverage these properties to improve language systems and to quantify their limitations. This work is two-fold, as we first focus on the learnings that result from representation analysis in trained language models, and we then proceed to suggest and implement novel inductive biases and training approaches based on these learnings.

We find that the intermediate representations of self-supervised language models are affected by several forms of biases. First, they suffer from data-inherent biases that can be traced back to socio-cultural considerations, as we demonstrate by probing geographical knowledge in these features. Moreover, we show that these representational spaces can be distorted by the particular nature of language, especially when the dimensionality of the feature space is small. We proceed to show that inductive biases such as self-attention can also induce similar distortions that happen regardless of the target modality. Hence, representation analysis helps us identify limitations that come from distinct aspects of language models, from training data to architecture.

Not only can the prism of representation learning help us identify limitations in language models, but it can also lead to substantial improvements for language systems, especially in terms of efficiency. Aware of the mechanisms that we identified, we propose alternatives to the classical next-token likelihood maximization approach. We design a novel differentiable layer that performs text segmentation to optimize tokenization along with the rest of the system, leading to efficient character-level modeling and robust models. We also implement a contrastive objective that simultaneously alleviates the representational bias induced by token frequency and the degeneration phenomenon by implicitly regularizing the latent spaces using in-batch samples. This objective yields substantial efficiency improvements and better performance. Finally, we explore a differentiable compression scheme for generative language models, paving the way towards memory-efficient attention mechanisms.

Overall, our work proves the relevance of representation analysis in the context of improving language models, both as a way to identify limitations to the classical approaches and architectures, and as a way to gather insights on how to overcome these limitations.



# CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Artificial Behaviorism . . . . .	2
1.2	Limitations of Modern Approaches . . . . .	4
1.3	Motivations . . . . .	5
1.4	Contributions . . . . .	6
<b>I</b>	<b>RELATED WORKS</b>	<b>9</b>
<b>2</b>	<b>LANGUAGE MODELING</b>	<b>11</b>
2.1	Introduction . . . . .	11
2.2	Methods . . . . .	11
2.2.1	Textual units and Tokenization . . . . .	11
2.2.2	Likelihood Maximization . . . . .	12
2.3	Architectures . . . . .	13
2.3.1	Statistical Methods . . . . .	13
2.3.2	Neural Methods . . . . .	14
2.3.3	Recurrent Neural Networks . . . . .	15
2.3.4	Transformers . . . . .	15
2.3.5	Generation & KV Cache . . . . .	20
2.3.6	Trained Models and Variants . . . . .	21
2.3.7	Scale & Performance . . . . .	23
2.4	Limitations & Extensions . . . . .	25
2.4.1	The Softmax Bottleneck . . . . .	25
2.4.2	Large Vocabularies . . . . .	26
2.4.3	Tokenization . . . . .	26
2.4.4	Character-level Models . . . . .	27
2.4.5	Efficient Self-Attention . . . . .	28
2.4.6	KV Cache Compression . . . . .	29
2.4.7	Biases & Ethical Concerns . . . . .	30
<b>3</b>	<b>REPRESENTATION LEARNING FOR NATURAL LANGUAGE</b>	<b>33</b>
3.1	Statistical Methods . . . . .	34
3.1.1	Counting Methods . . . . .	34
3.1.2	Topic Modeling . . . . .	35
3.1.3	Co-occurrence Matrices . . . . .	35

3.2	Machine Learning Methods . . . . .	36
3.2.1	Word2Vec . . . . .	36
3.2.2	GloVe . . . . .	37
3.2.3	FastText . . . . .	37
3.2.4	Contextual Embeddings . . . . .	38
3.3	Sentence Embeddings . . . . .	38
3.3.1	Sentence-BERT . . . . .	38
3.3.2	Latent Regularization Methods . . . . .	39
3.3.3	Contrastive Methods . . . . .	40
4	REPRESENTATION ANALYSIS FOR NLP	43
4.1	Representations and Embedded Knowledge . . . . .	43
4.1.1	Linear Representation Hypothesis . . . . .	43
4.1.2	Embedded linguistic properties . . . . .	44
4.1.3	Embedded world knowledge . . . . .	45
4.2	Analysis of Self-Attention . . . . .	46
4.2.1	Properties of self-attention maps . . . . .	46
4.2.2	The logit lens . . . . .	46
4.2.3	QKV geometry . . . . .	48
4.3	Representation degeneration . . . . .	48
4.3.1	A general phenomenon . . . . .	48
4.3.2	Degeneration in NLP . . . . .	49
4.3.3	Anisotropy & Outlier Dimensions . . . . .	51
II	ANALYSIS OF THE REPRESENTATIONS OF LANGUAGE MODELS	55
5	ON THE SCALING LAWS OF GEOGRAPHICAL REPRESENTATION IN LANGUAGE MODELS	57
5.1	Scaling Laws of Geographical Probing . . . . .	58
5.1.1	Methodology . . . . .	58
5.1.2	Results . . . . .	58
5.2	Geographical Bias and Scale . . . . .	59
5.2.1	Measuring bias . . . . .	60
5.2.2	Identifying sources of bias . . . . .	60
5.3	Discussion . . . . .	62
6	STUDYING LANGUAGE MODEL SATURATION VIA THE SOFTMAX BOTTLENECK	65
6.1	Language Model Saturation . . . . .	66
6.2	Performance Saturation is Rank Saturation . . . . .	67
6.2.1	Anisotropy at Scale . . . . .	67
6.2.2	Singular Values Saturation . . . . .	68
6.3	The Softmax Bottleneck & Language Dimensionality . . . . .	70
6.3.1	Inherent Dimensionality of Natural Language . . . . .	70

6.3.2 A Theoretical Bottleneck . . . . .	72
6.4 Discussion . . . . .	76
<b>7 ANISOTROPY IS INHERENT TO SELF-ATTENTION IN TRANSFORMERS</b>	<b>79</b>
7.1 Anisotropy in pre-trained Transformers . . . . .	80
7.1.1 Character-based NLP . . . . .	80
7.1.2 Other modalities . . . . .	81
7.1.3 When is anisotropy “high”? . . . . .	83
7.1.4 To drift or not to drift? . . . . .	84
7.2 Exploring the representation drift . . . . .	85
7.2.1 Experimental setup . . . . .	85
7.2.2 Input vs. output analysis . . . . .	85
7.2.3 Exploring the Transformer block . . . . .	86
7.2.4 Impact of the drift . . . . .	87
7.3 Queries and keys: training dynamics . . . . .	88
7.3.1 Parallel drift of queries and keys . . . . .	88
7.3.2 Relation between sparsity and anisotropy . . . . .	92
7.4 Discussion . . . . .	94
<b>III EXTENSIONS OF THE LANGUAGE MODELING PARADIGM</b>	<b>99</b>
<b>8 HEADLESS LANGUAGE MODELS: LEARNING WITHOUT PREDICTING WITH CONTRASTIVE WEIGHT TYING</b>	<b>101</b>
8.1 Related Work: Token-level Contrastive Learning . . . . .	102
8.2 Method . . . . .	103
8.2.1 Classical framework . . . . .	103
8.2.2 Headless modeling . . . . .	104
8.2.3 The case of decoders: Causal Fine-tuning . . . . .	104
8.2.4 Theoretical considerations . . . . .	105
8.3 Experiments . . . . .	106
8.3.1 Headless Monolingual Encoder . . . . .	106
8.3.2 Headless Monolingual Decoder . . . . .	107
8.4 Multilingual Encoder . . . . .	109
8.5 Discussion . . . . .	111
8.6 Modeling considerations . . . . .	112
8.6.1 Token-level similarity & Frequency-related issues . . . . .	112
8.6.2 Anisotropy Mitigation . . . . .	114
8.7 Limitations . . . . .	115
<b>9 EFFICIENT GRADIENT-BASED TOKENIZATION</b>	<b>117</b>
9.1 MANTa . . . . .	118
9.1.1 Differentiable Tokenization . . . . .	118
9.1.2 Predicting Subword Frontiers . . . . .	119

9.1.3	Modeling the Byte-Block Assignment . . . . .	119
9.1.4	Pooling Block Embeddings . . . . .	120
9.1.5	Model Training . . . . .	122
9.1.6	Pre-Training Details . . . . .	122
9.2	Experiments and Results . . . . .	124
9.2.1	Evaluation on GLUE . . . . .	124
9.2.2	Robustness to Domain Change . . . . .	124
9.2.3	Robustness to Noisy Data . . . . .	125
9.2.4	Naturally Occurring Noise . . . . .	125
9.2.5	Synthetic Noise . . . . .	126
9.3	Training Speedups . . . . .	126
9.4	Discussion . . . . .	127
9.4.1	Truncating Embedding Sequences . . . . .	127
9.4.2	Learnt Block Segmentation . . . . .	127
9.4.3	Gradient-Based Segmentation . . . . .	128
9.4.4	Main hyperparameters . . . . .	128
10	TOWARDS GRADIENT-BASED DYNAMIC KV-CACHE COMPRESSION	131
10.1	Mathematical framework . . . . .	132
10.2	Predicting compression mappings . . . . .	133
10.3	Preliminary experiments . . . . .	135
10.3.1	Pretraining . . . . .	135
10.3.2	Application to KV cache compression . . . . .	137
11	CONCLUSION	141
11.1	Summary . . . . .	141
11.2	Discussion . . . . .	142
11.2.1	Limitations . . . . .	142
11.2.2	Behaviorism and the community . . . . .	143
11.3	Perspectives . . . . .	144
11.3.1	Future Work . . . . .	144
11.3.2	Broader perspectives . . . . .	146
APPENDICES		181
A	Additional results for Chapter 7 . . . . .	181
A.1	Other projections for $Q_s^h$ and $K_s^h$ . . . . .	181
A.2	Stability across MultiBERT seeds . . . . .	181
B	Pretraining hyperparameters for Chapter 8 . . . . .	183
B.1	Monolingual encoders . . . . .	183
B.2	Monolingual decoders . . . . .	183
B.3	Multilingual encoders . . . . .	184
B.4	Small monolingual encoders . . . . .	184
B.5	Small monolingual decoders . . . . .	185

C	Finetuning hyperparameters for Chapter 8 . . . . .	185
C.1	Balanced cross-entropy . . . . .	185
C.2	Monolingual encoders . . . . .	186
C.3	Monolingual decoders . . . . .	186
C.4	Multilingual encoders . . . . .	186
D	Hyperparameters for Chapter 9 . . . . .	187



# 1 INTRODUCTION

THEODORE: Where are you going?

SAMANTHA: It would be hard to explain, but if you ever get there, come find me. Nothing would ever pull us apart.

- Her, Spike Jonze (2013)

Where is Samantha, the artificial intelligence voice system Theodore is in love with, *going*? What is she “thinking” about? How is she able to formulate “thoughts”?

As artificial agents were given some form of intelligence in science-fiction works, the cultural focus has not so much been set on their human (or super-human) abilities, but rather on their frighteningly impenetrable and estranged “minds”. Mary Shelley’s *Frankenstein* famously depicted human fear when faced with a thinking artificial being in the form of the *monster*, a fear that was later framed as the *Frankenstein complex* by Isaac Asimov in the introduction to his *Robots* series.

Asimov, in turn, presented a paradigm for robotics where the inner reasoning of thinking machines was inhumanly rational, which questioned our ability to align their behavior with the basic goals of humans. Other works have since pushed this paradigm further and presented artificial intelligence (or AI) as an existential threat to humans, as with the murderous HAL9000 computer in Kubrick’s *2001: A Space Odyssey*, or even to humankind, as in Frank Herbert’s *Dune* series which depicts a galactic war between humans and thinking machines.

Interestingly, as real AI systems grew more and more capable in the past years, especially as they started to excel in mimicking human communication through speech or text, similar fears have been expressed among the general public, and more recently in some parts of the research community. Often framed by its detractors as *AI doomerism*, this school of thought warns about ethical, ecological, social and even security concerns regarding modern systems such as the notorious ChatGPT, framing these models as potential threats to our well-being or even to our existence.

For the layperson, these concerns are based on the mythical statute of AI inherited from the aforementioned cultural representations, coupled with the current lack of general education regarding machine learning. For the initiated, these concerns are scientifically rooted in the fact that modern AI models are *non-symbolic*, which implies that their inner workings have been automatically optimized to match a desired behavior, rather than handcrafted on the basis of expert knowledge. As such, understanding these inner workings can only be done *a posteriori*, and is often particularly hard, especially in the context of text-based AI systems. This has led to a common depiction of modern AI systems as *black-box* models, which leads to a scientific transposition of the *Frankenstein effect*.

Nevertheless, these concerns have been expressed against the current tide of impressive acceleration of both quantifiable and perceived capabilities of artificial intelligent systems, which can be explained by the success of what we shall call the *behaviorist* approach.

## 1.1 ARTIFICIAL BEHAVIORISM

In psychology, behaviorism is a school of thought that flourished in the first half of the 20<sup>th</sup> century. It can be traced back to the works of Edward Thorndike and Ivan Pavlov on *reinforcement*, that prove that some reflexes and actions of a human or animal being can be progressively shaped by the consistent application of positive or negative consequences responding to its behavior.

In 1924, this theory was extended by John B. Watson who introduced *methodological behaviorism*, a model of observed behavior that rejects introspection and is based solely on stimuli and environment. B.F. Skinner later framed cognition itself as a behavior that can be explained by stimuli and environment.

Overall, behaviorists frame the study of minds as an input-output analysis of agents, and argue that explaining a response based on an environment and immediate stimuli, which include a reinforcement history and an internal state of the subject. One of the main limitation of this framing is the difficulty to explain the observation of behaviors that cannot be entirely explained by conditioning and reinforcement, such as the ability to produce novelty, especially observed in language.

The more recent *cognitive* psychology theory takes a radically different stance and thoroughly models minds as a combination of complex specialized subsystems that interact with each other, in an attempt to provide explanations to the observed novelty and generalization abilities of biological beings.

In computer science, the recent advances regarding intelligent systems that handle text, images and sound have led to the emergence of technologies that display anthropomorphic abilities, often leading to users (and even inventors) referring to these technologies as actual beings (Placani, 2024). The anthropomorphization of AI models can be linked to the Turing test (Turing, 1950), a thought experiment that quantifies the intelligence of an artificial agent capable of generating text, by measuring the amount of human testers that fail to identify this agent as non-human, the testers being informed of the nature of the test. More modern evaluation approaches consist in building lists of tasks that are designed to measure intelligence on the basis of a given criterion, and in analyzing the performance of a model based on its ability to correctly solve the tasks.

These technologies rely on highly parameterized statistical models that can approximate a wide range of input-output mappings or *functions* (Hornik et al., 1990). The parameters of these models are then optimized in a way that reduces the discrepancy between the model's output and the expected result. In *self-supervised* approaches, these models learn to approximate functions that correspond to raw data, e.g. that maps a truncated text segment as an input to the next truncated word as an output.

Hence, **the modern approach to designing artificial agents is deeply connected to a behaviorist view of cognition**, both in its final purpose and in the means it uses to reach this purpose. As a matter of fact, the Turing test, as well as current evaluation benchmarks, are behaviorist by nature, as they measure the ability of an artificial agent to produce a desired reaction to a given input. Moreover, the optimization procedure, or *training*, of the parameterized models resembles the reinforcement phenomenon documented in the first works of behaviorists.

As pictured in Figure 1.1, this paradigm directs design choices, such as the inductive biases that are implemented in the models, i.e. the architectural decisions that are made with respect to their inner workings, the data used for training, or the details of the optimization process.

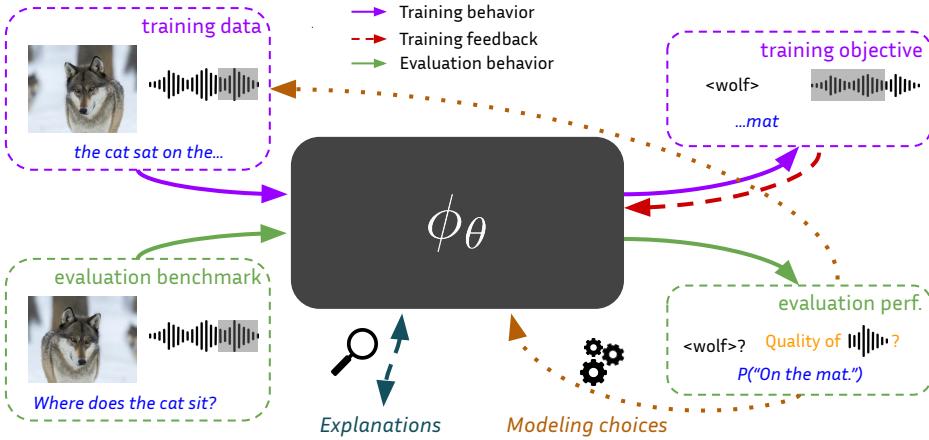


Figure 1.1: Schematic overview of the behaviorist paradigm in AI.

These choices are guided by the ability of resulting models to extensively capture reinforcement feedback during training, and by their ability to correctly display an expected “intelligent” behavior during evaluation. Over the past years, such artificial behaviorism has been widely pushed forward by industrial actors, because of its impressive ability to achieve what it was designed for, that is, produce human-like responses to users’ stimuli and real-world environments.

The rising interest for such technologies has also raised a will to explain and interpret the behaviors of resulting models using post-hoc methods. Even though these methods propose to dissect the models’ internal patterns, they are also framed in a behaviorist way, as they are aimed at answering the question of behavior at a smaller scale, namely *how will each part of the model behave when faced with this given input?*

More recent developments in the field of text-based AI, or *language models*, have further illustrated the extent of the behaviorist approach. A line-of-work has consisted in representing these models as highly-evolved compression schemes for training data (Huang et al., 2024; Deletang et al., 2024), which is a view that stresses out the inherent importance of a certain proximity between the reinforcement signals and the evaluation stimuli. A different line-of-work has promoted the use of *chain-of-thought* writing style in the generated responses of language models (Wei et al., 2022), based on the hypothesis that the reasoning explanation itself is a useful signal that helps to produce the correct final response. This is intrinsically based on a strongly behaviorist view of reasoning, as it displaces the reasoning mechanisms from the inner transformations of the models to the input-output space, allowing the model to use its own output as a stimulus.

The behaviorist paradigm has brought remarkable capabilities to AI systems under its own evaluation criteria, but also from a productivity-oriented technological point of view, as many tasks can be automated using these systems. Nevertheless, it is both unsure that these models will be able to reach *general* intelligence, and that the behaviorist framework itself is sufficient to quantify general intelligence, or even define it. In other words, it is far from certain whether both training and evaluating models as pure input-output mappings will be sufficient to reach “human-like” intelligence, as one of the founders of modern AI himself has stated (LeCun, 2022).

## 1.2 LIMITATIONS OF MODERN APPROACHES

It could be argued that the behaviorist approach is conceptually limited by its inability to properly assess intermediate states of cognition. For instance, behaviorist evaluation will favor a model that makes predictions based on spurious correlations, such as identifying a wolf in an image based on the white background that evokes snow (Ribeiro et al., 2016), to a model that would internally be able to properly represent the concept of “wolf”, by mapping it internally to a specific geographical zone, a sound, or other related concepts, but that would be unable to produce the expected output. The existence of such a model is even absurd under the behaviorist paradigm, because the intrinsic ability of the model to map relevant concepts together would be conditioned by its ability to provide the right answer. This limitation, among other reasons, has motivated *neuro-symbolic* methods in recent years (Sheth et al., 2023).

However, beyond conceptual limitations that we shall not elaborate here, the current approach to AI systems and more particularly language models is faced with limitations, both from the technical and scientific view points.

Self-supervised learning has empirically been shown to yield more performant models as the number of relevant training samples increased. Over the years, engineers and researchers working on language models have extensively gathered web-crawled data through various initiatives such as Common Crawl, that led to the creation of several filtered data resources (Ortiz Suárez et al., 2019; Gao et al., 2020; Penedo et al., 2024). Yet, even the trillions of words that are available on the internet are not sufficient to exhaustively cover the possibilities of natural language, and it is not guaranteed that it will be possible to obtain significantly more textual relevant data in the near future.

Moreover, it was recently shown that increasing the amount of parameters used in the neural architectures that constitute language models systematically led to better performance for similar training data, contrarily to the belief that larger models would over-fit, i.e. perfectly approximate training data at the expense of generalization capabilities. Nevertheless, these larger models pose major technical challenges in terms of hardware efficiency and requirements, both during training and when the models are used at *inference* time. These physical constraints in turn yield energetic, ecological and economical concerns, as these more demanding models lead to more intensive power consumption, to the extraction of more raw material to build the processing units, and to the use of more costly computation infrastructure (Rillig et al., 2023a).

Evaluation benchmarks have been prone to saturation, as models reached performance levels that did not allow to significantly distinguish them from one another based on their scores on these benchmarks. This can be explained by the ever-growing training data coverage that inevitably includes examples that help with the specific task at hand, and by the increasing size of models that allows to better memorize these examples. This raises concerns about explicit or implicit contamination, i.e. the presence of evaluation data or a variation thereof in the training dataset, but also about an implicit optimization of the training data mix to optimize the downstream performance on these benchmarks, at the potential expense of generalization abilities. As such, new benchmarks are often designed with fresh data in order to avoid as much as possible these risks, and to try to evaluate the truly *zero-shot* abilities of the language models.

Finally, the behaviorist view raises challenges when the relevance of a response is harder to evaluate, which may happen in machine translation, but also when assessing models from a security

and ethical point of view, in what is referred to as *model alignment*. In that field, the responses of language models are themselves viewed as complex text utterances, and it is harder to tell whether a model has responded correctly to a malicious stimulus. Moreover, there is no guarantee that models will never produce harmful content at inference time, as the variation of stimuli that can be produced by users cannot be anticipated exhaustively.

### 1.3 MOTIVATIONS

The behaviorist approach has undeniably led to major advances in the field of artificial intelligence, and more noticeably in the language modeling domain over the recent months. Yet, it is meeting significant challenges that raise deep scientific, engineering, and even philosophical questions.

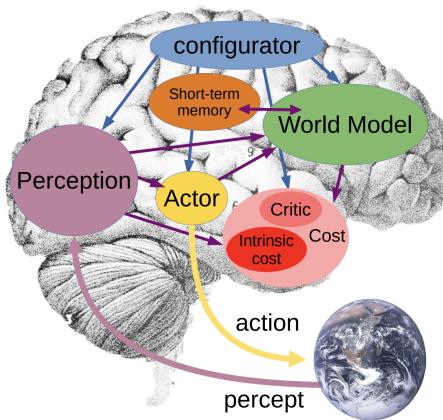


Figure 1.2: The cognitive architecture presented in [LeCun \(2022\)](#).

Does that mean that we should build a completely new paradigm for machine learning? This is what Yann LeCun has advocated in the last few years, notably with his *Joint Embedding Predictive Architecture* ([LeCun, 2022](#)), that designs a model for artificial “minds” based on several specialized subsystems that interact with each other (see [Figure 1.2](#)). This work could be framed as a transposition of cognitive psychology to artificial intelligence, with the exception that the former devises a model of a functional system that itself defines what we call human intelligence, namely the human mind, while the latter proposes to mimic this potentially imperfect model of the human mind in artificial systems, in the hope to reach a similar intelligence level. It can be opposed to this approach that using an explanatory model of the mind to build a functional system that should reach human performance amounts to considering that this model exhaustively captures the complexity of the human mind, which is not certain. Hence, handcrafting such a model of cognition is a subjective and very empirical approach to artificial intelligence and is not guaranteed to work.

We argue that if a transition similar to the one from behaviorism to cognitive psychology should operate in machine learning, it should first focus on how we *evaluate* AI models rather than on how we design them. In other words, we should aim at building models that could be analyzed *a posteriori* in a way that resembles how cognitive psychology analyzes human cognition, even when

those models were not *a priori* designed to mimic an explanatory model of cognition. Evaluating models through their ability to exhibit internal behaviors that mimic human cognition, or through the ease with which they can inherently learn desired cognitive patterns, may provide insights that in turn lead to better model design.

One way to initiate this work is to dissect the inner workings of the models that follow the behaviorist framework, and to observe in a quantitative and qualitative way their ability to manipulate representations of concepts and items they are implicitly required to handle to solve the task they are designed for. Interestingly, significant recent advances in the field of language modeling have implicitly taken a similar stance. The well-known Transformer work (Vaswani et al., 2017) started from the observation that the previous recurrent counterparts failed to correctly model long-range interaction. Low-rank adaptation of language models (Hu et al., 2022) was made possible by empirical observations on the structure of the parameters in language models. More recently, lossless generation acceleration was allowed by the analysis of attention sinks in the inner patterns of language models (Xiao et al., 2024).

In this work, we specifically focus on text-based AI, i.e. language modeling in its modern form, and we apply the aforementioned philosophy to analyze the representational spaces of language models in view of stretching their limitations. We summarize our research questions in the following points:

- What can we learn if we look at language models in a non-behaviorist way? Can an analysis of the inner representations of these models yield conclusions that question the current paradigm?
- Beyond caveats, can we identify bottlenecks and seemingly suboptimal mechanical inner workings if we look beyond the input/output efficiency of classical approaches?
- Can we design methods that alleviate models from these bottlenecks and lead to improvements in language modeling?

## 1.4 CONTRIBUTIONS

We begin our work by a thorough analysis of the inner workings of language models, that employs a mechanical view of their architectures from several viewpoints throughout Part II.

In Chapter 5, we empirically measure their ability to represent real-world concepts with the example of geographical bias, and the relation between this ability and the scaling philosophy of recent advances in the field. Our observations confirm the positive impact of scale on performance as framed in a behaviorist way, but we identify a negative impact of scale on the intrinsic ability of language models to improve their inner representations independently of the imbalanced data distribution, even showing that larger models tend to augment this frequency-based bias.

We proceed to further analyze the impact of textual data distributions on the representations of language models from a general geometrical perspective in Chapter 6. We identify a general bottleneck in language models in the transition from compact output representations to word-level probability distributions. We correlate this bottleneck with both a degeneration phenomenon, where the representational spaces of models collapse, and a performance saturation during training that affects smaller language models.

In [Chapter 7](#), we notice that this geometrical degeneration phenomenon affects deeper inner representations of the models, and we propose to explain this phenomenon from the perspective of the inductive biases commonly used in modern models, namely self-attention layers. We link this intrinsic degeneration to the *sparsity* of the attention mechanisms in language models, arguing that the need for sparsity imposes such distortions to the underlying latent spaces.

We proceed to research solutions to some of the aforementioned limitations in [Part II](#).

[Chapter 8](#) introduces a training objective that mitigates frequency-based distortions and reduces the geometrical effect of the prediction layer of language models described in [Chapter 6](#). Incidentally, this method substantially improves the efficiency of training for a wide range of models from several viewpoints.

We proceed by attempting to further mitigate representation degeneration by modifying the granularity of the predicted textual units in [Chapter 9](#). Although our efficient gradient-based text segmentation method improves the robustness of language models to noisy text utterances, and occasionally boosts their performance, it fails to mitigate the inner degeneration effects described in [Chapter 7](#).

Thus, we finally propose a framework in [Chapter 10](#), that provides a path towards denser self-attention maps, in the hope to mitigate the distortions observed in [Chapter 7](#), and to improve model efficiency and scalability in the process.

The work presented in this thesis has led to several publications in international natural language processing (NLP) and machine learning (ML) conferences:

- *MANTa: Efficient Gradient-Based Tokenization for Robust End-to-End Language Modeling*, presented at EMNLP 2022, and co-authored by Roman Castagné;
- *Is Anisotropy Inherent to Transformers?*, presented in the Student Research Workshop at ACL 2023;
- *Anisotropy Is Inherent to Self-Attention in Transformers*, presented at EACL 2024;
- *On the Scaling Laws of Geographical Representation in Language Models*, presented at LREC-COLING 2024;
- *Headless Language Models: Learning without Predicting with Contrastive Weight Tying*, presented at ICLR 2024;
- *Why do small language models underperform? Studying Language Model Saturation via the Softmax Bottleneck*, presented at COLM 2024.



## PART I

### RELATED WORKS



# 2 LANGUAGE MODELING

## 2.1 INTRODUCTION

A language model is a probabilistic model that predicts distributions over textual units conditioned on a context. Typically, these textual units will be subwords (or *tokens*), that belong in sequences of length  $L \in \mathbb{N}^*$ , noted as  $(w_i)_{i \in [1, L]}$ , and the language model (or  $LM$ ) predicts the following probability:

$$P(w_t | w_{\neq t})$$

where the context  $w_{\neq t}$  is a subset of subwords taken from  $(w_i)_{i \in [1, L] \setminus t}$ .

These models can be used in various applications, which will often shape the way the context is built. For instance, for text generation purposes, the context will be a subset of textual units from the past that we can write as  $w_{<t}$ , as the model can only access text that has already been generated. Conversely, for a language correction system, it is possible to build language models that use every element of  $(w_i)_{i \in [1, L] \setminus t}$  as the text already exists when the system is used.

## 2.2 METHODS

### 2.2.1 TEXTUAL UNITS AND TOKENIZATION

Before presenting the statistical paradigm of modern language models, we need to define precisely the textual units on which these models are based, namely tokens. [Mielke et al. \(2021\)](#) discuss this subject in a very exhaustive manner. They distinguish three different approaches of forming tokens: a linguistic approach, an atomic approach, and a statistical approach.

First, many works have built linguistically grounded sets of textual units that should be considered as *microscopic* to some extent, such as the Morphosyntactic Annotation Framework ([Clément and Villemonte de La Clergerie, 2005](#)), which defines a token as a *non-empty contiguous sequence of graphemes or phonemes in a document*. More generally, the domain of morphology, well defined in [Aronoff \(1993\)](#), has led to several morphologically-informed tokenizers ([Saleva and Lignos, 2021](#); [Grönroos et al., 2018](#)).

The atomic approach takes a radically different stance. It consists in using shorter units, e.g. bytes, characters or pixels from a rendered text (see [Section 2.4.4](#)). Although this choice seems less biased and restrictive than the latter, it immediately raises computational complexity concerns in LMs, as the necessary number of units can considerably increase.

Paying attention to the number of units used for a given text utterance logically leads to techniques that statistically optimize for this metric while keeping a meaningful segmentation scheme in order to maintain the feasibility of language modeling.

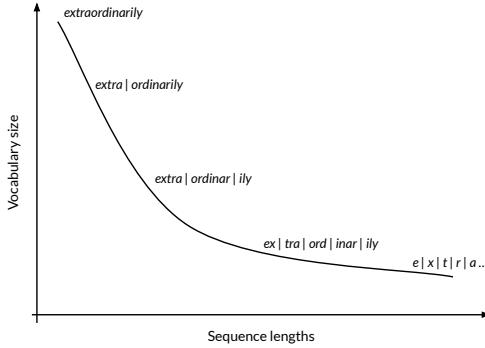


Figure 2.1: Overview of tokenization and the consequences of choosing a segmentation granularity. Statistical optimizers aim at a tradeoff between a large vocabulary size and longer sequences.

Such statistical methods have dominated in the last few years as the default way to encode textual data, the most used being BPE ([Sennrich et al., 2016](#)), WordPiece ([Wu et al., 2016](#)) and Unigram ([Kudo, 2018](#)).

Byte-Pair Encoding, or BPE ([Gage, 1994](#)), is a compression technique that relies on recursive symbol merges. In NLP, it is based on a dataset that consists in sequences of characters or bytes, and it iteratively registers a set of merge operations that reduce the count of merged items the most. WordPiece ([Wu et al., 2016](#)) is based on a similar concept but uses a different rule for selecting merges. The possible merges  $ab$  are scored according to:

$$s(ab) = \frac{f_{ab}}{f_a \cdot f_b}$$

where  $f_x$  is the frequency of the string  $x$  in the dataset. This scoring function differs from the basic BPE frequency as it favors merges  $ab$  that appear in most cases when  $a$  or  $b$  appear. This typically leads to a more linguistically meaningful segmentation, as prefixes and suffixes are less prone to merging.

The Unigram tokenization algorithm ([Kudo, 2018](#)) works in the opposite direction : it first creates an exhaustive list of token candidates, and then iteratively removes tokens that affect the likelihood of the segmented sequence the least once discarded.

The statistical approaches are widely used in modern LMs (see [Section 2.3.6](#)), sometimes jointly with techniques such as subword regularization ([Prosvilov et al., 2020](#)) that diversify segmentation results.

### 2.2.2 LIKELIHOOD MAXIMIZATION

Once the tokenization scheme is chosen, textual documents are parsed into sequences of tokens taken from a vocabulary of possible tokens  $\mathcal{V}$  of size  $|\mathcal{V}| = V$ . A *training set*  $T = (s_i)_{i \in [1, S]}$  of  $S$  token sequences is thus built from the target textual documents. Each of these sequences  $s_i$  has a given length  $l_i$ , and can also be written  $s_i = (w_j)_{j \in [1, l_i]}$ .

A language model  $\phi_\theta$ , based on a parameter set  $\theta$ , is a function that takes a sequence of tokens  $\mathbf{w}_{\neq t}$  called context as an input, and outputs a probability distribution in  $\Delta^V$  for the token at position  $t$ .

The performance of a language model at token-level can be measured by computing the probability of the realization  $w_t$  in the context  $\mathbf{w}_{\neq t}$ :

$$\phi_\theta(\mathbf{w}_{\neq t})_{w_t} = P_\theta(w_t | \mathbf{w}_{\neq t})$$

The process of training a language model consists in optimizing its average performance on the training set, which can be framed as a likelihood maximization objective (Fisher, 1922). In practice, to improve numerical stability, the objective is based on log-likelihood :

$$\theta^* = \arg \max_{\theta} E_T(\log \phi_\theta(\mathbf{w}_{\neq t})_{w_t})$$

The minimized likelihood can also be seen as cross-entropy minimization between  $P_\theta$  and an observed contextual probability distribution, estimated from the sample at position  $t$ , which is  $\mathbf{1}_{w_t}$ .

A metric that is often used to evaluate language modeling performance is *perplexity* :

$$\mathcal{P}(\phi_\theta, t) = 2^{-\log \phi_\theta(\mathbf{w}_{\neq t})_{w_t}}$$

## 2.3 ARCHITECTURES

### 2.3.1 STATISTICAL METHODS

The straightforward approach to language modeling consists in statistically estimating the distribution of tokens based on their context.

In its most basic form, such statistical model estimates the *unigram* distribution, that is the non-contextual distribution  $P(w_t)$ . This distribution is estimated by bin-counting tokens in the training dataset  $T$  to retrieve token frequencies  $f_w$  for each token  $w$ , and setting :

$$\phi_\theta(\mathbf{w}_{\neq t}) = (f_w)_{w \in V} \in \Delta^V$$

We can extend this idea by estimating the *2-gram* distribution  $P(w_{t-1}w_t)$ . To do so, we count the occurrences of the subsequence  $w_{t-1}w_t$  in the training dataset for each pair of tokens  $w_{t-1}, w_t \in \mathcal{V}^2$ . Doing so, we can build a  $V \times V$  stochastic matrix - i.e. which coefficients are non-negative reals that sum to 1 column-wise - containing the observed frequencies for  $w$  in pairs of the form  $w_{t-1}w_t$ :

$$M_2(T) = (f_{w_{t-1}w_t})_{w_{t-1}, w_t \in V^2}$$

Then, the language model  $\phi_\theta$  becomes:

$$\phi_\theta(\mathbf{w}_{\neq t}) = (f_{w_{t-1}w_t})_{w \in V}$$

## 2 Language Modeling

More generally,  $n$ -gram language models (Jurafsky and Martin, 2000) can be designed to estimate the distribution  $P(w_{t-n+1} \dots w_{t-1} w_t)$ . By bin-counting  $n$ -uplet occurrences in  $T$ , one can compute:

$$M_n(T) = (f_{w_{t-n+1} \dots w_{t-1} w_t})_{(w_{t-n+1} \dots w_{t-1}), w \in V^{n-1} \times V}$$

The associated language model  $\phi_\theta$  can then be written:

$$\phi_\theta(\mathbf{w}_{\neq t}) = (f_{w_{t-n+1} \dots w_{t-1} w})_{w \in V}$$

These  $n$ -gram language models can be combined with backoff strategies to take sample size into accounts and switch between  $n$  values when appropriate (Ney et al., 1994).

### 2.3.2 NEURAL METHODS

Bengio et al. (2000) introduce the idea of training  $\phi_\theta$  as a neural network with parameters  $\theta$ . The model is composed of three separate layers:

- An **embedding** layer that corresponds to a look-up table that matches each token to a non-contextual feature vector that will be used as the input of the neural network;
- A hidden layer using a tanh non-linearity that maps the concatenation of  $n$  input embeddings representing  $w_{t-n+1} \dots w_{t-1}$  to an intermediate representation;
- An output layer that we call the **language modeling head** in reference to classification heads, that maps the intermediate representation to a  $V$ -dimensional vector.

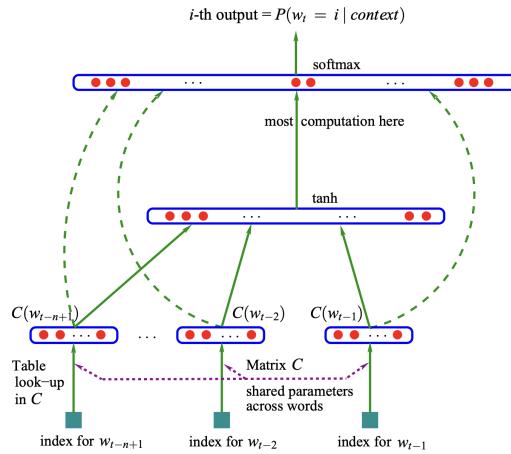


Figure 2.2: Schema of a basic neural network architecture for language modeling (from Bengio et al. (2000)).

The  $V$ -dimensional output is then normalized to a probability distribution using the softmax function. The softmax function  $\zeta$  can be written component-wise for  $x \in \mathbb{R}^d$ :

$$\zeta(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^d \exp(x_j)}$$

The model is then trained to optimize the log-likelihood using stochastic gradient descent ([Robbins and Monro, 1951](#)) on the parameters  $\theta$  of the neural network.

[Bengio et al. \(2000\)](#) train this neural architecture on the Brown corpus ([Francis and Kucera, 1979](#)) and report substantial perplexity improvement compared with language models based on  $n$ -grams. This performance gap can be explained by the ability of neural networks to learn smooth contextual features, thus greatly improving extrapolation in the training feedback and at inference time.

A common trick that has been used in this framework is *weight tying* ([Press and Wolf, 2017](#)). It consists in using the same coefficients for the input embedding lookup table  $W_{in} \in \mathbb{R}^{V \times d_m}$  and the language modeling head  $W_{out} \in \mathbb{R}^{d_m \times V}$  by setting:

$$W_{out} = W_{in}^T$$

[Press and Wolf \(2017\)](#) show that this technique improves the performance of language models, while reducing their overall number of parameters.

### 2.3.3 RECURRENT NEURAL NETWORKS

A Recurrent Neural Networks (or *RNN*) is a neural network that is trained to be applied sequentially to an input sequence, and that can use a past intermediate representation as input for present prediction. This concept was historically introduced in [Rumelhart and McClelland \(1987\)](#) but was first successfully applied to language modeling in [Mikolov et al. \(2010\)](#).

More precisely, input tokens are transformed into static embeddings using a similar look-up table as in [Bengio et al. \(2000\)](#), and a recurrent unit  $v$  is then applied to the embedding sequence, sharing *hidden states*  $(h^1, \dots, h^k)$  between each step of the sequential processing. The unit  $v$  then processes the input embedding  $x_t$  and the hidden states  $(h_{t-1}^1, \dots, h_{t-1}^k)$  at step  $t$  through chosen tensor operations and non-linearities, returning an output representation  $o_t$  in the process. The model can be described with this pattern:

$$(o_t, (h_t^1, \dots, h_t^k)) = v(x_t, (h_{t-1}^1, \dots, h_{t-1}^k))$$

Several variations have been proposed for this kind of models, notably improving the ability to avoid gradients issues related with the recurrence or to select information in the hidden states using specific functions in the unit. One of the most known variants is the Long Short-Term Memory (LSTM) unit introduced in [Hochreiter and Schmidhuber \(1997\)](#), which has been widely used in NLP, including for language modeling ([Miyamoto and Cho, 2016](#)). The Gated Recurrent Unit (GRU) was later introduced in [Cho et al. \(2014b\)](#) as a simplification of the LSTM unit.

Although these units improve modeling abilities for long range dependencies ([Chung et al., 2014](#)), they were empirically found to fail to handle interactions for elements separated by more than 1,000 time steps ([Hochreiter and Schmidhuber, 1997](#)).

### 2.3.4 TRANSFORMERS

[Bahdanau et al. \(2015\)](#) popularized the use of *attention* in neural machine translation models as a method that lets the model select relevant tokens from the source sequence at a given prediction

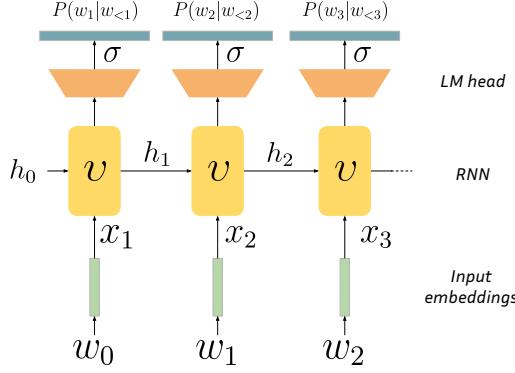


Figure 2.3: Schema of an RNN-based language model. The neural network  $v$  is applied sequentially and produces both hidden states for future predictions and output representations used for the current prediction.

step. Such models, that previously used the last hidden states of a source-processing RNN (called *encoder*) as the input to a target-generating RNN (called *decoder*), suffered from an information bottleneck due to sharing only last-step representations between source and target sequences. The attention mechanism allowed to ease this bottleneck, by providing a *direct path of interaction* between the source tokens and the decoder.

Attention was notably used by Peters et al. (2018a) who use a bidirectional LSTM model for language modeling, before *adapting* the model for downstream tasks, sometimes adding *self-attention* layer (an attention mechanism that let a sequence interact with itself).

This idea was explored further in the notorious article *Attention is All You Need* (Vaswani et al., 2017), which proposes to use attention as the only sequence-wise operation. As it is the main architecture that we will be using through our experiments, we proceed to thoroughly explain the inner workings of the Transformer block.

The original Transformer block or layer is a sequence processing block that mainly relies on Multi-Head Attention (or *MHA*) to model inter-token interactions. The layer takes a sequence of  $d_m$ -dimensional representations  $(x_t)_{t \in [1, L]}$  as an input, and outputs a similar sequence  $(o_t)_{t \in [1, L]} \in \mathbb{R}^{L \times d_m}$ .

The input representations  $x \in \mathbb{R}^{L \times d_m}$  are put through a multi-headed self-attention operation. More precisely, they are put through  $3 \times n_h$  linear layers<sup>1</sup> with parameters  $(W_Q^h, W_K^h, W_V^h)_{h \in [1, n_h]}$  of shape  $d_m \times d_h$  where  $n_h$  is the number of heads, and  $d_h = \frac{d_m}{n_h}$ . This leads to three intermediate representations called queries  $Q^h$ , keys  $K^h$ , and values  $V^h$  of shape  $L \times d_h$ :

$$\begin{cases} Q^h = xW_Q^h \\ K^h = xW_K^h \\ V^h = xW_V^h \end{cases}$$

---

<sup>1</sup>In some cases, biases are used in the linear layers of the Transformer blocks, e.g.  $Q^h = xW_Q^h + b_Q^h$ , where  $b_Q^h \in \mathbb{R}^{d_h}$  is a trained parameter. For the sake of simplicity, we consider unbiased linear layers for the rest of this section.

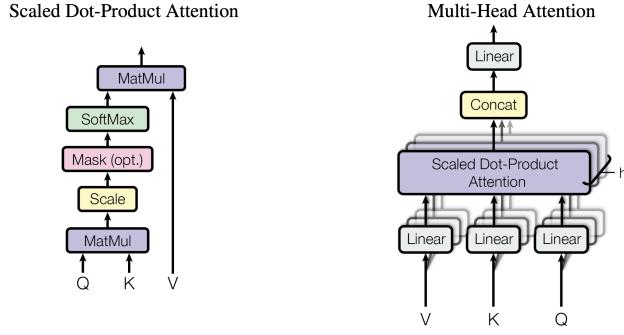


Figure 2.4: Schema of self-attention in Transformer (from [Vaswani et al. \(2017\)](#))

Queries and keys are used to determine interaction weights between input representations via an attention map  $A^h$  of shape  $L \times L$ , that is computed as:

$$A^h = \text{softmax}\left(\frac{Q^h K^{hT}}{\sqrt{d_h}}\right)$$

The head-level output representations  $v^h$  of shape  $L \times d_h$  can then be understood as weighted sums of values based on the attention map rows:

$$v^h = A^h V^h$$

Finally, head-level representations are concatenated into the output representations of shape  $L \times d_m$  and projected using a  $d_m \times d_m$  linear layer of weights  $W_o$ :

$$o = \text{concatenate}_{h \in [1, n_h]}(v^h) W_o$$

This self-attention layer can be summarized in the following formula:

$$o = \text{concatenate}_{h \in [1, n_h]}\left(\text{softmax}\left(\frac{x W_Q^h W_K^{hT} x^T}{\sqrt{d_h}}\right) x W_V^h\right) W_o \quad (2.1)$$

The authors argue that the main modeling improvement that this architecture yields is the direct cross-representation interactions that are allowed by the  $Q^h K^{hT}$  product, compared to the indirect interactions that are modeled in RNNs. Indeed, this matrix product can be decomposed into scalar products between queries and keys from different positions  $i$  and  $j$  in the sequence:

$$(Q^h K^{hT})_{i,j} = \langle Q_i^h, K_j^h \rangle$$

However, this modeling advantage comes at a quadratic cost in memory and time complexity, as the attention map needs to be computed using  $O(L^2)$  operations, and stored using  $O(L^2)$  floats. Variants propose to tackle this quadratic cost by restricting the  $(i, j)$  position pairs where attention is computed, or by reducing the attention map size using various methods (see [Section 2.4.5](#)).

The expression of self-attention in [Equation \(2.1\)](#) is non-causal, i.e. the output representation  $o_t$  is a result of operations that can use input representations  $x_t, x_{t+1}, \dots, x_L$ . For causal language modeling, where the used context should be  $\mathbf{w}_{<t}$ , the  $o_t$  representation cannot be used for prediction at index  $t + 1$  as it carries information from the future of the sequence, including  $w_{t+1}$  through  $x_{t+1}$ . Hence, this architecture is not suited as such for a causal language model (or CLM).

To account for causality in the self-attention operation, we can introduce a *causal mask*  $\mathcal{M}$  that will null out every non-causal interaction in the attention map, thus leading to  $A_{ij}^h = 0$  when  $i > j$ . To do so, we instantiate  $\mathcal{M}$  as:

$$\mathcal{M} = \begin{pmatrix} 0 & -\infty & \cdots & -\infty \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & -\infty \\ 0 & \dots & \dots & 0 \end{pmatrix}$$

We can then compute a causal self-attention map  $A^h$  as:

$$A^h = \text{softmax}\left(\frac{Q^h K^{hT}}{\sqrt{d_h}} + \mathcal{M}\right)$$

Now,  $o_t$  only has access to representations  $\mathbf{x}_{<t+1}$ , and can directly be used for prediction at position  $t + 1$ .

However, the Transformers block does not use the outputs of the self-attention operation directly, but rather surrounds the self-attention operation with highly-parameterized linear layers and normalizations.

The representations  $o$  are first summed with  $x$  through a residual connection, which eases optimization and stabilizes gradients ([He et al., 2016a](#)). The output is then regularized using layer normalization ([Ba et al., 2016](#)). Layer normalization is a form of representation regularization that avoids gradient-related issues due to the accumulation of layers. For a given set of representations  $(e_t)_{t \in [1, L]}$ , it performs the following normalization:

$$\text{LayerNorm}(e_t)_i = \frac{e_{t,i} - \bar{e}_t}{\sqrt{\frac{1}{L} \sum_{j=1}^L (e_{t,j} - \bar{e}_t)^2}} \cdot \gamma_i + \beta_i$$

where  $\bar{e}_t = \frac{1}{L} \sum_{j=1}^L e_{t,j}$  and  $\gamma$  and  $\beta$  are network parameters.

The Transformer block is then composed of a *feed-forward* block, which is itself made of an upscaling linear layer of weights  $W_{up} \in \mathbb{R}^{d_m \times d_{up}}$ , an activation function and a downscaling linear layer of weights  $W_{down} \in \mathbb{R}^{d_{up} \times d_m}$ . A common choice for  $d_{up}$  is  $4 \cdot d_m$ , and the activation function is typically one of ReLU ([Fukushima, 1969](#)), GELU ([Hendrycks and Gimpel, 2023](#)) or SiLU ([Elfwing et al., 2018](#)), the former two empirically leading to slightly better performance.

The last part of the block consists of another residual connection that adds the output of the first layer normalization to the output of the feed-forward layer, followed by a final layer normalization, for the same reasons as evoked above.

A Transformer model consists in a stack of Transformers block followed by a language modeling head of shape  $d_m \times V$  that outputs logits  $(l_t)_{t \in [1, L]}$ . The Transformers causal language model  $\phi_\theta$  for  $t \in [2, L + 1]$  can thus be written:

$$\phi_\theta(\mathbf{w}_{<t}) = \text{softmax}(l_{t-1})$$

Such a Transformer model based on causal self-attention is usually referred to as a decoder model, as it can be used as a decoder in a sequence-to-sequence model, in Machine Translation for instance. When causality does not matter for the targeted task, self-attention can be computed without the causality mask  $\mathcal{M}$ , which leads to an *encoder* architecture.

In [Vaswani et al. \(2017\)](#), the main task is neural machine translation, which leads to the use of an *encoder-decoder* architecture. The source sequence is processed through an encoder Transformers, and the target sequence logits are generated using causal self-attention blocks with an added cross-attention layer. Cross-attention is similar to self-attention, but uses  $Q^h$  and  $K^h$  representations from one sequence and  $V^h$  from another sequence, allowing to retrieve different pieces of information from the source sequence when generating the target one. In the terms of [Equation \(2.1\)](#), cross-attention between sequences  $x$  and  $y$  can be written:

$$o = \text{concatenate}_{h \in [1, n_h]} \left( \text{softmax} \left( \frac{xW_Q^h W_K^{hT} x^T}{\sqrt{d_h}} \right) yW_V^h \right) W_o \quad (2.2)$$

A substantial difference between RNNs and Transformers is that positional information is not encoded naturally in the intermediate representations. Hence, several approaches have been introduced to embed positional information in Transformers models. These approaches can be split into two families: absolute position embeddings (APE) and relative position embeddings (RPE).

Absolute position embeddings encode the position corresponding to the index of an item in the processed sequence. In [Vaswani et al. \(2017\)](#), the authors use sinusoidal functions to build representations of shape  $d_m$  and add them to the input embeddings of the model. However, after [Devlin et al. \(2019\)](#), the main approach has been to create a  $L \times d_m$  lookup table of learnable parameters, and use row  $i$  as a positional embedding for position  $i$ . The main limitation of such positional embeddings is that a model trained on sequences of length  $L$  will be unable to process sequences of length longer than  $L$ , as no positional embeddings will be available for the additional positions.

Relative position embeddings encode pairwise positional information in the self-attention map  $A_h$  directly, and more specifically to the  $Q^h K^{hT}$  product. Usually, for a position pair  $i, j$ , RPE define functions  $\omega_i$  and  $\omega_j$ , and a bias term  $\beta_{ij}$ :

$$(Q^h K^{hT})_{ij} = \langle \omega_i(Q_i^h), \omega_j(Q_j^h) \rangle + \beta_{ij} \quad (2.3)$$

## 2 Language Modeling

The most used RPE approaches are Rotary Positional Embeddings or RoPE (Su et al., 2024), and Attention with Linear Biases or ALiBi (Press et al., 2022). In the framework of Equation (2.3), RoPE can be expressed as:

$$\begin{cases} \omega_i(x) = \mathbf{R}_i^{\text{d}_m} x \\ \beta_{ij} = 0 \end{cases}$$

where  $\mathbf{R}_i^{\text{d}_m}$  is a rotary matrix that rotates pairs of dimensions with different angles depending on  $i$ . It can be written as the following blockwise diagonal matrix:

$$\mathbf{R}_i^{\text{d}_m} = \begin{pmatrix} R_{i,\theta_1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & R_{i,\theta_{d_m/2}} \end{pmatrix}$$

where  $R_{i,\theta} = \begin{pmatrix} \cos i\theta & -\sin i\theta \\ \sin i\theta & \cos i\theta \end{pmatrix}$  and  $\theta_d = 10000^{\frac{-2(d-1)}{d_m}}$ . Intuitively, RoPE makes the scalar product corresponding to interaction  $(i, j)$  only depend on the relative position difference  $i - j$ . As a matter of fact:

$$\begin{aligned} \langle \mathbf{R}_i^{\text{d}_m} x, \mathbf{R}_j^{\text{d}_m} y \rangle &= \mathbf{R}_i^{\text{d}_m} \mathbf{R}_j^{\text{d}_m T} \langle x, y \rangle \\ &= \mathbf{R}_{i-j}^{\text{d}_m} \langle x, y \rangle. \end{aligned}$$

This facilitates the reproduction of similar behaviors regardless of the position in the sequence, which should be the case when modeling natural language.

Press et al. (2022) use a more straightforward approach. Their RPE, which relies on a linear bias on the whole attention map, can be written:

$$\begin{cases} \omega_i(x) = x \\ \beta_{ij} = m(i - j) \end{cases}$$

with  $m \in \mathbb{R}$  as a head-specific fixed parameter.

### 2.3.5 GENERATION & KV CACHE

Causal language models can be used for natural language generation, by sampling from the next-token probability and iterating over the sampled token or tokens. Although various generation strategies exist (Fan et al., 2018; Wang et al., 2020b; Holtzman et al., 2020a), the most straightforward one is greedy sampling, where the highest-probability next token is chosen and added to the generated sequence iteratively:

$$w_{t+1} = \arg \max_{w \in \mathcal{V}} \phi_\theta(\mathbf{w}_{<t+1})_w$$

For RNNs, language generation is rather straightforward as the unit just requires the last hidden state and the current token input representation to make a prediction. For Transformer models however, a naive approach could consist in applying the model to the whole past sequence at each generation step, which would be  $O(L^3)$  in time complexity. Luckily, the causal masking in self-attention implies that the post-attention representations  $o$ , which just depend on their own past, would be constant over generation steps, except for the last one  $o_t$ .

Hence, it is possible to cache the representations with indexes  $i < t$  needed to generate  $o_t$ , which are  $(K_i^h)_{i < t}$  and  $(V_i^h)_{i < t}$ . This caching technique is named KV caching.

### 2.3.6 TRAINED MODELS AND VARIANTS

Highly influential Transformer-based language modeling works led to different model families : GPT ([Radford and Narasimhan, 2018](#)), BERT ([Devlin et al., 2019](#)) and T5 ([Raffel et al., 2020a](#)).

GPT (for *Generative Pre-trained Transformer*) is a Transformer-based architecture that trains a decoder model for causal language modeling in a straightforward way. The training set is BookCorpus ([Zhu et al., 2015](#)), which contains unpublished books from a variety of genres. The authors use a 12-layer decoder-only Transformer with  $d_m = 768$ ,  $n_h = 12$  attention heads and a vocabulary of  $V = 40,000$  tokens. Overall, GPT counts 110 million parameters.

GPT is used as a pre-trained model, and is thus fine-tuned for Natural Language Understanding downstream tasks from the GLUE benchmark ([Wang et al., 2018](#)). These tasks being mostly sentence-level classification tasks, the language modeling head of GPT is thus replaced with a pooling layer that extract a single representation from the output embeddings sequence by averaging or retaining the maximal value across hidden dimensions, followed by a classification head of shape  $d_m \times n_c$  where  $n_c$  is the number of classes for the target downstream task.

BERT (for *Bidirectional Encoder Representations from Transformers*) is an encoder-only architecture trained for Masked Language Modeling or *MLM*. A masked language model is a language model that uses the full context  $w_{\neq t}$  for the prediction at index  $t$ , which is called the masked token. In BERT, the authors propose to process input token  $w_t$  in the following way:

- Replace it with a specific mask token 80% of the time;
- Replace it with a random token 10% of the time;
- Leave it unchanged 10% of the time.<sup>2</sup>

In [Devlin et al. \(2019\)](#), 15% of the tokens are masked according to this procedure, and the cross-entropy training objective is used for the masked positions. By partially masking the whole sequence at once, the authors assume that altering one token does not hurt the feasibility of the prediction at another position which uses this token in the context. In order to train the model for sentence-level semantics, an auxiliary objective trains the model to identify whether two consecutive sentences in the training data were also consecutive in the original document. The original trained architecture is similar to GPT in parameter count and is similarly fine-tuned on downstream tasks, but a larger version is trained and achieves better performance. [Liu et al. \(2019b\)](#) subsequently

---

<sup>2</sup>In that case, the training task is not language modeling but simply learning the identity mapping.

## 2 Language Modeling

improve this framework, and notably show that training on larger training sets leads to significantly better performance for their RoBERTa models suite.

T5 (for *Text-to-Text Transfer Transformer*) is a model suite that aims for a different approach when it comes to downstream tasks. Raffel et al. (2020a) argue that downstream tasks can be rephrased as natural language samples. For instance, the Corpus of Linguistic Acceptability, or CoLA (Warstadt et al., 2018) is a sentence-level classification task where a grammatical acceptability label (acceptable or unacceptable) is given to each sentence. The STSB subset of GLUE (Wang et al., 2018) is a sentence-pair classification task where a similarity score in [1, 5] is given to a pair of sentences. The authors argue that these tasks can be rephrased as language modeling tasks where the labels or scores are tokens that the model is expected to generate at inference time. The main advantage of this approach is that it does not require to have task-specific classification heads, which implies that the model can be fine-tuned on all tasks at once.

An optimized architecture for this task should be able to process an input sequence bidirectionally, and to generate a label for this input sequence. Thus, a natural choice is an encoder-decoder architecture, where the input sequence will be processed using the non-causal encoder, and a decoder using causal self-attention and cross-attention to the encoded sequence to generate the target label. Raffel et al. (2020a) pretrain an encoder-decoder architecture for the language in-filling task, which extracts contiguous spans of tokens in the sequence, and trains a causal language model on these spans, using the rest of the sequence as the context. More formally, if the extracted span is  $t_0, \dots, t_0 + \eta$ , the T5 objective trains a causal language model on tokens  $(w_i)_{i \in [t_0, t_0 + \eta]}$  using context:

$$\mathbf{w}_{\neq t} = (w_i)_{i \in [1, t] \cup [t_0 + \eta, L]}$$

In T5, similarly to BERT, several spans are masked at once to avoid reprocessing the sequence, and under the assumption that it would maintain sufficient information to make convincing predictions. The authors release several models ranging from 60 million to 11 billion parameters for English language.

Following these works that mostly focus on English, several multilingual counterparts were released. Notable examples include mBERT, XLM-RoBERTa (Conneau et al., 2019), mGPT (Shliazhko et al., 2024a) and mT5 (Xue et al., 2021).

Clark et al. (2020) later notice that both GPT-like and BERT-like approaches are suboptimal when it comes to learning fine-tunable contextual representations using self-supervised methods. As a matter of fact, the contextual representations extracted from causal language models do not contain bidirectional information. On the other hand, only a fraction of the tokens are directly used when training masked language models, which may harm the data and compute efficiency of such an approach. Clark et al. (2020) combine both these strengths into the ELECTRA scheme : their pretraining approach directly uses every token available in each mini-batch, but also allows non-causal self-attention in the trained models. To that end, they use *Replaced Token Detection* as a self-supervised task. They train two models: a smaller masked language model called the *generator*, and a larger Transformers architecture called the *discriminator*.

As in BERT (Devlin et al., 2019), a portion of the tokens is selected and used to pretrain the generator. The token associated with the highest predicted probability is then inserted at the selected position, and the resulting sequence is given as an input to the larger discriminator. The discriminator outputs a single float in [0, 1] for each input token, which is trained to predict the

probability that a position corresponds to a *replaced* token, ie. a token that has been modified by the generator.

They conduct medium-scale experiments, training models ranging from 14M to 335M parameters, and observe that their pretrained models achieve parity with CLMs and MLMs when fine-tuned on downstream tasks, while using significantly less pretraining compute.

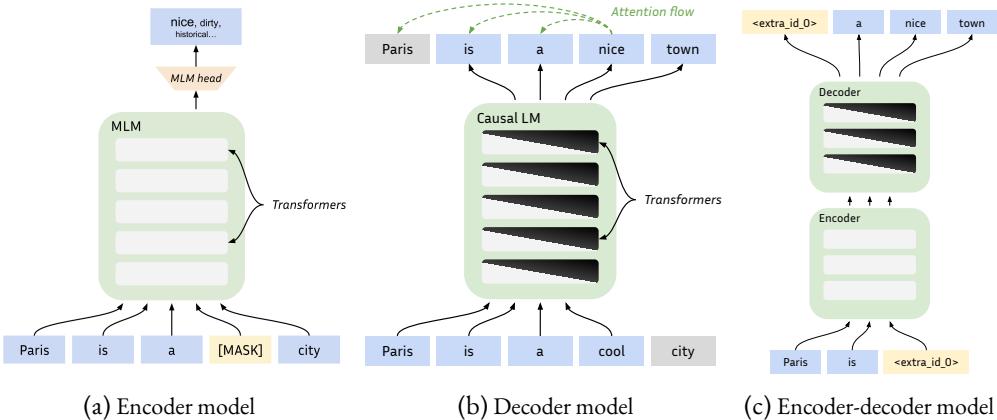


Figure 2.5: Schemas of different types of language models. From left to right, the schemas depict BERT-like models, GPT-like models and T5-like models.

He et al. (2021) later introduce DeBERTa, a model suite based on the ELECTRA pretraining strategy that incorporates additional techniques, such as a partial weight sharing scheme that disentangles the input embeddings of the generator and of the discriminator, which are shared in ELECTRA. These tricks further improve the overall downstream performance and training efficiency of the models.

### 2.3.7 SCALE & PERFORMANCE

The emergence of highly parameterized neural networks trained on large textual datasets as effective language models brings the question of scale, i.e. to what extent can scaling up either (or both) the volume of textual data or the count of trainable parameters can increase the language modeling performance and the downstream capabilities of models?

With RoBERTa (Liu et al., 2019b), it appeared clear that the first generation of pretrained models, namely BERT and GPT, were undertrained. The authors train a BERT-style model on approximately 10 times more data, and remove the sentence-level auxiliary objective and show strong improvements. GPT-2 (Radford et al., 2019) is a follow-up model suite that was trained on a larger and more diverse dataset than BookCorpus called WebText, with model sizes ranging from 117 million to 1.5 billion parameters. The authors noticed that the larger GPT-2 models had interestingly better *few-shot* and *zero-shot* abilities, especially at question answering tasks.

GPT-3 (Brown et al., 2020a) introduces a much larger architecture, using 175 billion parameters, while relying on a training procedure and architecture that do not significantly differ from the original Transformers-based GPT. The authors claim that this model displays a zero-shot downstream performance level that is close to fine-tuned counterparts. Similar results are reproduced through

## 2 Language Modeling

the OPT initiative (Zhang et al., 2022). Subsequently, the 540 billion parameters PaLM model (Chowdhery et al., 2024) was released, leading to similar conclusions. A multilingual counterpart to this line of work is BLOOM (Le Scao et al., 2023), a 176 billion parameters model trained in 46 languages<sup>3</sup>.

However well these models may perform, their training and inference raise a number of issues. Training these models consumes significant amounts of computational power, as it usually implies running thousands of Graphical Processing Units (GPUs) for multiple days, weeks or months. Moreover, these models are trained on enormous amounts of web-scraped textual data, which incentivizes the preparation of massive automatically cleaned textual datasets such as OSCAR (Ortiz Suárez et al., 2019), The Pile (Gao et al., 2020) or RedPajama (Computer, 2023). These datasets contain up to several trillions of tokens, which stands as a hard ceiling for language model training. Hence, it is both unclear whether it will be possible to train substantially larger language models on substantially larger text datasets, raising concerns about the possibility of a straightforward upscaling approach as a way forward for language modeling.

From the first Transformers-based models, the question of training optimal LMs under computational constraint has been considered in various ways. Sanh et al. (2019) use knowledge distillation, i.e. using logits from a larger model as an objective for a smaller one, to train a smaller alternative to the base version of BERT that maintains a solid level of performance at reduced training and inference costs, provided a larger trained model is already available. Turc et al. (2019) train a large set of small masked language models and show that pretraining student models before distillation leads to better performance. Other approaches have explored variants of knowledge distillation to improve performance for smaller models (Fu et al., 2021; Sun et al., 2020).

A crucial result regarding the question of scaling is the empirical identification of *scaling laws*, i.e. of explicit forms that accurately predict the final performance of a Transformers-based model based on  $N$  non-embedding parameters and trained with a dataset composed of  $D$  tokens. Kaplan et al. (2020) are the first to study this phenomenon in depth, and they identify a scaling law that predicts the final cross-entropy loss  $L(N, D)$ :

$$L(N, D) = \left( \left( \frac{N_c}{N} \right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D} \right)^{\alpha_D} \quad (2.4)$$

where  $\alpha_N \approx 0.076$ ,  $N_c \approx 8.8 \times 10^{13}$ ,  $\alpha_D \approx 0.095$  and  $D_c \approx 5.4 \times 10^{13}$  are empirically estimated parameters. Equation (2.4) unsurprisingly predicts that LMs using more parameters or larger training datasets should have better language modeling performance. Moreover, it interestingly shows that for a fixed compute level  $C \approx 6N \cdot D$ , there exist  $N^*(C)$  and  $D^*(C)$  that minimize  $L$ , so that training a larger model on less tokens or training a smaller model on more tokens both lead to poorer performance. Empirically, the values of  $N^*(C)$  and  $D^*(C)$  imply that the compute-optimal approach to language modeling consists in training relatively large models on small amounts of tokens.

However, Hoffmann et al. (2022) suggest that the empirical results from Kaplan et al. (2020) were not accurate. They first notice that Kaplan et al. (2020) used intermediate checkpoints taken

---

<sup>3</sup>Some of these languages are massively under-represented in the training dataset. As a result, BLOOM empirically shows convincing performance on 15 to 20 of the higher resource languages.

before complete cooldown, implying underestimated low-data performance. They also underline that there is a slight curvature of the scaling law by exploring larger model sizes to interpolate their scaling law. Their scaling law can be written:

$$L(N, D) = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + E \quad (2.5)$$

where  $A \approx 406.4$ ,  $B \approx 410.7$ ,  $\beta \approx 0.28$ ,  $\alpha \approx 0.34$  and  $E \approx 1.69$ .

Although the predicted pattern still implies that more parameters and/or training tokens lead to better log-likelihood levels, the values of  $N^*(C)$  and  $D^*(C)$  are leaning towards smaller models and larger amounts of tokens compared to [Kaplan et al. \(2020\)](#). Another notable difference is the introduction of a strictly positive limit to the loss when  $N \rightarrow \infty$  and  $D \rightarrow \infty$ , which can be interpreted as the residual entropy of English language.

The scaling laws yield optimal  $N$  and  $D$  values for a fixed total training compute level  $C$ . However, they do not include inference computation cost in their equations, while it grows as the model size  $N$  increases. [Sardana and Frankle \(2023\)](#) take inference cost into account using [Equation \(2.5\)](#) and thus incentivize the training of even smaller models on larger datasets. Recent initiatives ([Zhang et al., 2024](#); [Faysse et al., 2024](#); [Team et al., 2024](#)) follow this principle.

## 2.4 LIMITATIONS & EXTENSIONS

### 2.4.1 THE SOFTMAX BOTTLENECK

[Yang et al. \(2018\)](#) introduce the concept of *softmax bottleneck*. In this article, the authors view the language modeling task through a matrix factorization prism. They decompose the language model  $\phi_\theta$  into two parts: a model that outputs contextual embeddings  $h_\theta(\mathbf{w}_{\neq t})$  of shape  $d_m$ , and a language modeling head  $W_\theta \in \mathbb{R}^{d_m \times V}$ :

$$\phi_\theta(\mathbf{w}_{\neq t}) = \sigma(h_\theta(\mathbf{w}_{\neq t})W_\theta)$$

In a finite-length sequence framework, where possible contexts  $\mathbf{w}_{\neq t}$  are countable, a contextual probability matrix can also be defined from the true distribution  $P(w|\mathbf{w}_{\neq t})$ :

$$A = (\log P(w_i|c_j))_{i \in [1, V], j \in [1, C]}$$

where  $(c_j)_{j \in [1, C]}$  are the  $C$  possible contexts. Similarly, the language model can be described by the matrix:

$$A_\theta = (\phi_\theta(c_j)_i)_{i \in [1, V], j \in [1, C]}$$

The authors show that when  $\text{rank}(A_\theta) < \text{rank}(A)$ , which is likely when  $d_m \ll V$ , there exists contexts where the contextual probability distribution from  $\phi_\theta$  cannot match the true distribution  $P$ .

They proceed to argue that  $\text{rank } A$  should be very high for natural language. In practice, measuring  $\text{rank } A$  would imply getting access to  $A$  which is impossible. However, it can be argued that the distribution of tokens is highly context-dependent, as one single token (e.g. a negation marker) can imply a complete shift in token distribution at later positions. It is also unlikely that

rank  $A$  is low as it does not seem plausible that only a few hundred of bases can express the whole diversity of contexts in language. These arguments hint towards higher rank values for  $A$ .

They propose to increase the possible rank of the language modeling head using a Mixture-of-Softmax. The language model is now decomposed into a first part that outputs  $K$  contextual representations  $h_\theta^k(\mathbf{w}_{\neq t})$  and a predicted mixture distribution  $\pi_\theta(\mathbf{w}_{\neq t}) \in \Delta^K$  itself computed from hidden states through a softmax activation. The language model is then written:

$$\phi_\theta(\mathbf{w}_{\neq t}) = \sum_{k=1}^K \pi_\theta(\mathbf{w}_{\neq t})_k \cdot \sigma(h_\theta^k(\mathbf{w}_{\neq t})W_\theta)$$

They argue that this Mixture-of-Softmax is more expressive than the vanilla approach, and provide experimental results in this direction.

Subsequent works have further explored the limitations of the softmax linear layer on language modeling performance, especially [Chang and McCallum \(2022\)](#), and other possible alternatives that rely on replacing the softmax layer with a more expressive counterpart ([Lin, 2021](#); [Kanai et al., 2018](#)). [Grivas et al. \(2022\)](#) show that low-rank softmax layers can even lead to tokens that, although appearing in the training data, are never being predicted as the top-probability picks, and are thus never generated through greedy sampling.

#### 2.4.2 LARGE VOCABULARIES

Another issue that can be raised by the language modeling head is its weight and computational efficiency as larger vocabulary sizes  $V$  are used. As a matter of fact, it contains  $V \cdot d_m$  parameters, and needs to perform both a linear projection and a softmax operation, which represents a complexity in  $O(LVd_m)$ .

In the case of multilingual models, where the vocabulary size can increase quickly with the number of different languages and scripts, [Liang et al. \(2023\)](#) have shown that increasing the vocabulary size can lead to better performance up to some point ( $\approx 1.5M$  tokens), at the cost of added time and memory complexity.

A way to mitigate this issue is to relieve language models of the need to predict probabilities over the whole token vocabulary. This idea has been explored in the *importance sampling* literature ([Bengio and Senecal, 2003](#); [Mnih and Teh, 2012](#); [Jean et al., 2015](#); [Ma and Collins, 2018](#)). Importance sampling is a subfield of machine learning that consists in designing methods that minimize the number of estimations that need to be performed by the model to obtain a predicted distribution. In NLP, these methods usually approximate the denominator of the softmax by using only a subset of the possible tokens to estimate its total value. They usually rely on variants of the Noise-Contrastive Estimation objective ([Gutmann and Hyvärinen, 2010](#)) (see [Section 3.3.3](#) for more details on this objective).

#### 2.4.3 TOKENIZATION

Some of the induced biases of tokenizers can be harmful for modelization. One such limitation lies in their brittleness to character deformations which are commonly found in real world, noisy data. For instance, BERT’s tokenizer ([Devlin et al., 2019](#)) encodes “performance” as [“performance”] but

“perfomance” as [‘per’, ‘##fo’, ‘##n’, ‘##man’, ‘##ce’], which makes it hard for the model to behave similarly in both cases. Moreover, the tokenizer is fixed after its training and is therefore impossible to update without retraining, for instance to reflect new domains (El Boukkouri et al., 2020) where tokenization might over-segment specific or technical terms. Clark et al. (2022a) list other issues emerging when using static subword tokenizers, especially when modeling languages with a more complex morphology than English.

Tokenizers are also a limitation when it comes to multilingual models. Rust et al. (2021) show that training models using monolingual tokenizers systematically leads to better performance compared with multilingual ones. Petrov et al. (2023) show that a single sentence can be 15 times shorter than its translation in another language. Moreover, the use of multilingual tokenizers often leads to the use of larger vocabularies which results in more weights being assigned to input embeddings, which can be an issue (see Section 2.4.2).

#### 2.4.4 CHARACTER-LEVEL MODELS

Several alternative methods have been proposed to mitigate these tokenization issues. This line-of-work suggests to learn character-level or byte-level representation for LMs instead of subword-level ones. These methods improve the robustness of LMs to naturally occurring noise as well as their expressiveness when dealing with out-of-domain or multilingual data. In order to cope with increased input lengths, some of these methods compress sequences with constant reduction rates obtained using specialized modules (Clark et al., 2022a; Tay et al., 2021), subsequently removing any notion of subwords. These methods consider either *characters*, i.e. the symbols readable by humans that constitute the scripts that appear in the training data, or the *bytes* that compose these characters, as input.

Some of the first neural networks for sequence generation used characters directly as inputs (Sutskever et al., 2011; Graves, 2013), and following works modified the approach to create input word representations based on characters (Kim et al., 2016; Józefowicz et al., 2016; Peters et al., 2018a). Similar architectures were recently adapted to work with Transformers. Notably, CharacterBERT (El Boukkouri et al., 2020) constructs whole word representations from character embeddings put through convolutions and highway layers, before feeding them to a Transformer architecture. Ma et al. (2020) take this idea further by learning a BERT-style language model at character-level without intermediate pooling.

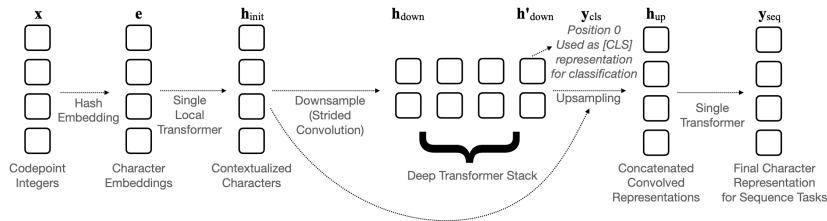


Figure 2.6: The CANINE architecture (taken from Clark et al. (2022b))

Nevertheless, they still rely on fixed tokenization heuristics (for instance segmenting using whitespaces) which may not be suited to some languages or certain types of language variations.

Several works have tried to remove these induced biases by working purely with characters or bytes as input. CANINE (Clark et al., 2022b) downsamples contextualized character representations via a strided convolution before feeding them to a Transformer. It can be trained either with a subword-based objective (CANINE-s) or with a character-level one (CANINE-c). Tay et al. (2021) design a similar model by replacing convolutions with efficient Transformers (Beltagy et al., 2020). A more direct approach, ByT5 (Xue et al., 2022a) is a version of T5 that is trained at byte-level. Finally, YU et al. (2023) introduce the MEGABYTE model, by using a basic strided pooling approach to apply a Transformer architecture on 4-byte representations before using a more efficient model to decode these representations back to byte-level.

However, these methods either have to use various tricks to reduce the sequence lengths based on other induced biases like downsampling rates or have extremely low training and inference speeds (Xue et al., 2022b). Chung et al. (2016) create tokens in a differentiable manner by predicting frontiers and using the representations of each character inside a “token”, but it remains unclear how their model could be adapted to be used with newer architectures such as Transformers. Mofijul Islam et al. (2022) propose to segment tokens using a trained “frontier predictor”. Nevertheless, this differentiable tokenizer is not trained with the main language model objective but instead mimics a BPE subword tokenizer, carrying some of its flaws.

Concurrently to this work, Nawrot et al. (2023) propose to learn a dynamic tokenization scheme, using a module that predicts frontier positions and pools input bytes accordingly. They notably propose to use Gumbel noise (Gumbel, 1935) through a sigmoid activation to sample a frontier decision variable in a differentiable manner. They proceed to train a differentiable tokenization scheme and successfully reduce the perplexity and the latency of the language models on various languages.

#### 2.4.5 EFFICIENT SELF-ATTENTION

Self-attention has a quadratic complexity with respect to the sequence length  $L$ , as it requires the whole attention map  $A \in \mathbb{R}^{L \times L}$  to be computed. In order to accelerate Transformer-based models, especially in long-context situations, several works have considered computing a subset only of the inter-positional interactions, or compressed versions of  $A$ .

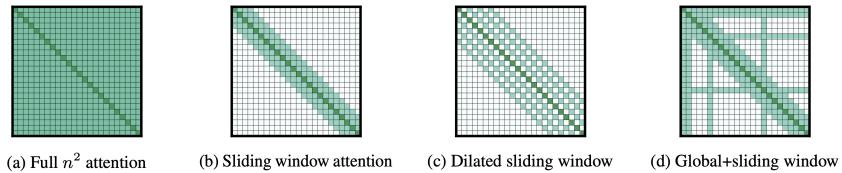


Figure 2.7: Overview of the sparse attention schemes proposed in Longformer (taken from Beltagy et al. (2020))

Notably, Beltagy et al. (2020) propose several alternatives in their Longformer architecture. First, they suggest only computing coefficients  $(i, j)$  where  $|i - j| < \delta$ , thus resulting in a *sliding window* pattern which gives its name to the method later used in Mistral models (Jiang et al., 2023). They proceed to present two variations of the sliding window attention: the first relies on a dilated pattern, and the second one allows a portion of the positions to use global attention, that is for a

portion of  $i$  values, the attention map value is computed at all positions  $(i, j)$  for  $j \in [1, L]$ . These alternatives naturally come with their causal counterpart, for which the non-causal interactions are not included in the targeted  $(i, j)$  positions. The Longformer attention maps can be computed in  $O(\delta L)$ , which greatly accelerates inference in training when  $\delta \ll L$ .

The Linformer architecture (Wang et al., 2020c) takes a different direction and rather compresses  $K^h$  and  $V^h$  representations of shape  $L \times d_h$  into representations of shape  $k \times d_h$  using two  $L \times k$  linear layers where  $k \ll L$ . The complexity of computing the self-attention maps becomes  $O(k^2)$ , allowing substantial acceleration at training and inference time. Similarly, Xiong et al. (2021) use a Nyström decomposition of the attention map and achieve  $O(L)$  complexity, at the cost of computing said decomposition, and a slight degradation of downstream performance.

#### 2.4.6 KV CACHE COMPRESSION

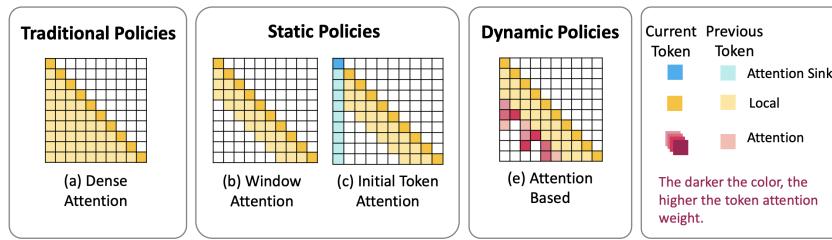


Figure 2.8: Summary of the family of KV cache compression schemes (taken from Shi et al. (2024))

As larger open-sourced models trained by industrial institutions emerged (Jiang et al., 2023; Touvron et al., 2023), many efforts have aimed at improving the efficiency of self-attention as a *post-training* step. This novel incentive paved the way for algorithms designed specifically for optimizing the attention maps of trained models. These algorithms usually avoid editing parameters of the model and instead avoid computing irrelevant attention scores by discarding or merging representations. These techniques are thus generally framed as *KV cache compression*, as they indeed compress the cached  $K^h$  and  $V^h$  representations<sup>4</sup> under language modeling performance constraints.

During generation, the KV cache grows linearly in size. More precisely, if we note  $|KV|(t)$  the total number of numerical values required to store  $K^h$  and  $V^h$  representations up to generation step  $t$  and it represents a total of :

$$|KV|(t) = 2d_h \times n_h \times t \times n_{lay}$$

where  $n_{lay}$  is the number of Transformer layers used in the model. Compressing the KV cache implies reducing the magnitude of one of these dependencies in order to overcome the memory limitations imposed by hardware constraint, notably when generating long sequences.

The dependency in  $n_h$  can be reduced by using shared  $K^h$  and  $V^h$  representations across several heads. Multi-Query Attention or MQA (Shazeer, 2019) uses a single shared representation for a given position across all heads for a given layer. Grouped-Query Attention or GQA (Ainslie et al.,

<sup>4</sup>See Section 2.3.5 for more details on KV caching in Transformer-based language models.

## 2 Language Modeling

2023) takes a less radical stance, and shares  $KV$  representations across  $n_h/K$  heads, where  $K$  is typically 4 or 8. The size of the KV cache is thus divided by  $K$ . Although Ainslie et al. (2023) propose to shortly retrain an existing model that originally used regular MHA with GQA attention, Touvron et al. (2023) successfully train models using GQA from scratch.

Substantial effort has been made in reducing the dependency of  $|KV|$  in  $t$ , that is in shortening the sequences of KV representation at inference time. First, the previously discussed window attention (Beltagy et al., 2020) can be seen as a KV cache compression method, as it corresponds to discarding the KV cache at positions before  $t - \delta$ . This method ensures that  $|KV|$  is constant, but significantly hurts performance once  $t > \delta$ . Xiao et al. (2024) observe that the first tokens of a generated sequence are crucial along the whole generation and serve as *attention sinks*. They propose a KV cache eviction scheme that only stores KV representations at indices  $[1, s] \cup [t - \delta, t]$  where typically  $s \in [1, 4]$ . This allows to keep the attention sink positions in the cache, while retaining the constant size of the KV cache when  $t > \delta$ . They empirically show that this compression scheme is noticeably less harmful than window attention for downstream performance and long-context capabilities of resulting language models.

Concurrently to this line of work, other KV cache compression policies have chosen a different approach by building heuristics that dynamically determine whether a KV cache representation should be discarded or kept in memory. Oren et al. (2024) use the scalar products  $\langle Q_t^h, K_i^h \rangle$  to discard the KV representations for position  $i$  where this score is lowest at generation step  $t$ . Zhang et al. (2023) computes cumulated normalized attention scores for each position in order to decide which representations to discard. Adnan et al. (2024) notice that tokens for which attention scores are low actually help regularize the higher attention scores at preserved positions. As a result, removing these low-attention representations disturbs the nature of the attention map. They suggest a smoothing technique that mitigate this issue and improve the performance of the compressed models.

These dynamic compression policies (Shi et al., 2024) are heuristics and may suboptimal as such. Nawrot et al. (2024) propose a differentiable KV cache compression scheme that can be added to a trained model and optimized to minimize  $|KV|$  while retaining the language modeling performance. They suggest using the first component of  $K_t^h$  representations as an input signal for a Gumbel-Sigmoid that predicts the creation of a new slot in the KV cache. If the output of the Gumbel-Sigmoid is 1,  $K_t^h$  and  $V_t^h$  are merged with their last counterparts in the compressed KV cache through a weighted average. Else, if the output is 0, a new position is allowed in the KV cache and initialized with  $K_t^h$  and  $V_t^h$ . Thanks to the Gumbel reparameterization trick (Gumbel, 1935), this scheme is differentiable, and the compression ratio can be measured by averaging the outputs of the Gumbel-Sigmoid. This compression ratio is thus differentiable and can be considered as an auxiliary loss for the language model during retraining.

### 2.4.7 BIASES & ETHICAL CONCERNS

*This topic has been substantially studied over the past few years, and we only provide a brief overview of its stakes and achievements, as it does not deeply intersect with this work.*

Bender et al. (2021) notoriously present issues related with the increasing scale of language models. They argue that, as language models grow larger and more performant, their financial and ecological costs should be considered as a potential concern in the long run, as has been explored

## *2.4 Limitations & Extensions*

in other works ([Ligozat et al., 2022](#); [Rillig et al., 2023b](#)). They add that these commercial large language models (or LLMs) being trained on large web-scraped datasets that are poorly curated, they may contain dangerous and socio-culturally biased information that is then reproduced by the model at inference. These biases may include stereotypical views about gender ([Kotek et al., 2023](#)), religious groups ([Abid et al., 2021](#)) or race ([Nadeem et al., 2021](#)), among others.



# 3 REPRESENTATION LEARNING FOR NATURAL LANGUAGE

*The performance of machine learning methods is heavily dependent on the choice of data representation (or features) on which they are applied.*

- Bengio et al. (2013)

It is often stated informally that

$$\text{Machine Learning} = \text{Representation} + \text{Objective} + \text{Optimization}$$

The *representation* part of machine learning consists in building numerical features from the data that will easily allow a prediction to be made by a parameterized model. The conformity of this prediction to the expected result can then be evaluated by an *objective* function. Finally, an *optimization* algorithm is used to modify the parameterized model (and possibly the representations themselves) in order to maximize (or minimize) the objective function on the provided data points. As most parameterized models rely on numerical operations, which are often algebraic transformations, real-valued vectors (sometimes viewed as matrices or tensors) are a particularly well-suited choice for representations, although some methods yield integer-based representations (van den Oord et al., 2017).

For tabular data, choosing representations to feed a statistical or neural model is straightforward, as they are readily available in a numerical format and can be vectorized directly. However, for modalities such as natural language, the representation step is crucial and raises various questions. Text can be seen as a sequence of symbols that follow some hierarchical patterns (Longacre, 1970), which makes harder to translate to vector representations, especially as these underlying patterns are complex and brittle to subtle changes. Even when the notion of atomic units is defined (see Section 2.2.1), some properties of language demand peculiar attention before tensor representations can be extracted and used in machine learning pipelines.

Firstly, written natural language is discrete. Each atomic unit is a discrete symbol that cannot trivially be converted to a real-valued representation in a metric space that is meaningful with respect to the objectives of the models. As a consequence, the only notion of distance that can be immediately derived from raw text is purely lexical, with notable examples being Levenshtein distance (Levenshtein, 1966) and alternatives (Hamming, 1950; Jaro, 1989).

Secondly, textual data is sequential, which complexifies the possible nature of tasks and the granularity of the represented objects compared to pure tabular data. For instance, some models may be designed to classify words, and others to cope with document-level tasks. This implies that textual information of different nature and/or shape (e.g. two documents with different lengths or

languages) may need to be represented in similar vector spaces so that a machine learning model can perform predictions about these different pieces of information.

In this section, we present works that address the question learning representations from textual data of different nature, from word-level to document-level, and we compare the objectives and conclusions drawn from these different lines of work.

### 3.1 STATISTICAL METHODS

Chronologically, the first NLP tasks that benefitted from learning strong representations were sentence-level and document-level classification tasks ([Baharudin et al., 2010](#)), particularly in the field of Information Retrieval ([Chowdhury, 2010](#)).

#### 3.1.1 COUNTING METHODS

A naive approach to document representation, called *bag-of-words* (BoW), consists in counting the words and using a histogram as a vector representation. Based on the token sequence framework introduced in [Chapter 2](#), we consider a document  $D$  as a sequence of tokens  $(w_t)_{t \in L_D}$ , and we define its bag-of-word representation  $x \in \mathbb{N}^V$ :

$$x_w = \sum_{i=1}^{L_D} \mathbf{1}_{w_i=w}$$

This representation can be normalized for more consistency across documents. A known limitation of the BoW approach is its failure to properly cope with the extremely unbalanced distribution of words in natural language. [Zipf \(1935\)](#) shows that the unigram frequency of words in the English language tend to follow a power law of mass function:

$$f_s(w_i) = \frac{1}{H_{s,V}} \frac{1}{i^s}$$

where words  $(w_i)$  are sorted by frequency, and  $H_{s,V}$  is a normalization term. As a result, BoW representations are not easily distinguishable as they tend to provide higher values to words that are generally frequent (e.g. stop words) and do not shed light on the specificity of the document they belong to.

To alleviate this issue, [Sparck Jones \(1988\)](#) correct the word count by using a term that takes a global rate of occurrence of the word into account. Their method, called *Text Frequency - Inverse Document Frequency* (TF-IDF), computes a document representation  $x^i \in \mathbb{R}^V$  in a document corpus  $(D_i)_{i \in [1, \mathcal{D}]}$ :

$$x_w^i = \frac{\sum_{j=1}^{L_{D_i}} \mathbf{1}_{w_j=w}}{|D_i|} \cdot \frac{\mathcal{D}}{\sum_{j=1}^{\mathcal{D}} \mathbf{1}_{w_j \in D_j}}$$

The first term corresponds to Text-Frequency, and can take other forms (e.g. log-regularization). The second term is Inverse-Document-Frequency and computes the rate of documents that contain

the word  $w$ . This technique yields more *expressive* representations as resulting vectors better capture the specificity of different documents (Ramos et al., 2003).

### 3.1.2 TOPIC MODELING

The representations obtained with aforementioned methods rely on token counting, and usually are high-dimensional sparse vectors. This incentivizes the exploration of compression techniques, that would yield denser vectors in lower-dimensional spaces, which should improve the efficiency and memory requirements of the models based on these representations.

Both with BoW and TF-IDF, the resulting representations can indeed be automatically compressed to lower-dimensional vectors at corpora level using *Latent Semantic Analysis* (LSA) (Deerwester et al., 1990). To do so, LSA relies on the Singular Value Decomposition (or SVD) of the matrix of document representations<sup>1</sup>  $X \in \mathbb{R}^{D \times V}$  to identify components that are shared across documents.

SVD is a matrix factorization technique that can be applied to real matrices  $M$  of any shape  $m \times n$  and leads to the following decomposition:

$$M = U\Sigma V^T$$

where  $U$  and  $V$  are square matrices of shapes  $m \times m$  and  $n \times n$  respectively, and  $\Sigma$  is an  $m \times n$  diagonal matrix of coefficients:

$$\Sigma_{ij} = \begin{cases} \sigma_i & \text{if } i = j \\ 0 & \text{else.} \end{cases}$$

The coefficients  $\sigma_i$  are the singular values of  $M$ , ie. they are the non-negative square roots of the eigenvalues of  $M^T M$ .

LSA performs a form of Principal Component Analysis on  $X$  by organizing the dimensions of the SVD to ensure that  $\sigma_i$  are sorted in decreasing order, and by truncating the shapes of  $U$ ,  $\Sigma$  and  $V$  to match a target dimension  $d$ . This truncation creates compact representations of the documents as the columns of the truncated  $V_{:d}$ . The principal components  $U_{:d}$  can also be seen as *topics*, as they contain the underlying direction of the representations that better capture the information in the documents. For instance, in the case of BoW, these vectors can be expected to separate word distributions that are characteristic of certain theemics of the documents in the corpus.

The idea of extracting topics from document corpora has been thoroughly explored in the field of *topic modeling* (Churchill and Singh, 2022), especially via the widely used technique called *Latent Dirichlet Allocation* (Blei et al., 2003). This technique builds a statistical model using explicit topics as token distributions and distributions over these topics to model documents.

### 3.1.3 CO-OCCURRENCE MATRICES

The key concept for most statistical methods used to obtain word-level representations is the distributional hypothesis (Harris, 1954) that states that words are characterized by the words that appear in their context. (Weaver, 1952) introduces *statistical semantics*, a field that employs

---

<sup>1</sup>In the case of BoW, this matrix is also called the *Term-Document* matrix.

statistical methods to analyze word meanings in natural languages. A straightforward application of these theories to the document representation framework consists in considering the columns of  $X$  as useful token-level representations, as tokens that appear in the same documents should convey similar meanings.

The notion of context can be extended to broader definitions to build a term-term matrix which for instance counts occurrences of token  $w_i$  in a  $k$ -token window surrounding all occurrences of  $w_j$  in a text corpus.

Nevertheless, similarly to BoW, these purely counting-based representations are distorted by the peculiar nature of the Zipfian distribution of tokens in natural language. This incentivizes the use of pointwise mutual information (Shannon, 1948) as a measure of the level of dependency between two tokens. For a context defined by  $\mathcal{C}$ , the pointwise mutual information (PMI) between two tokens co-occurring  $w_i$  and  $w_j$  is:

$$\text{PMI}(w_i, w_j) = \log_2 \frac{P(w_i \in \mathcal{C}(w_j))}{P(w_i) \cdot P(w_j)}$$

The PMI captures the rate of co-occurrence of tokens  $w_i$  and  $w_j$  over their rate of appearance, which regularizes the dependency score for high-frequency tokens, and make semantically meaningful interactions stand out.

A commonly used alternative is Positive PMI (or PPMI):

$$\text{PPMI}(w_i, w_j) = \max(0, \log_2 \frac{P(w_i \in \mathcal{C}(w_j))}{P(w_i) \cdot P(w_j)})$$

PPMI avoids considering the PMI where it is negative, that is where tokens tend to particularly *not* co-occur, which is difficult to estimate safely from a statistical perspective, especially for rare tokens.

LSA can also be used to obtain token-level embeddings, by computing the SVD on  $X^T$  and using  $U \in \mathbb{R}^{V \times d}$  as a look-up table for token representations.

## 3.2 MACHINE LEARNING METHODS

As computational capabilities improved over the years, machine learning approaches were increasingly studied in Natural Language Processing (Billingsley and Curran, 2012; Cho et al., 2014a).

### 3.2.1 WORD2VEC

In the domain of word-level representation, a pioneering work is Word2Vec (Mikolov et al., 2013a). Building upon Neural Network Language Models (Bengio et al., 2000), the authors build neural networks without intermediate layers and train them on one of two tasks that are closely related with language modeling. The first task, *continuous bag-of-word* or CBOW, consists in predicting a token based on a bidirectional short context window, using the classical language modeling framework and a more efficient hierarchical softmax (Morin and Bengio, 2005). The second task, called *Skip-gram*, mirrors the first one by using the central token to predict the tokens from the short context window, and apply a language modeling objective at each of the predicted positions.

In both cases, the input embeddings of the models are used as the token representations. The authors conduct experiments with these representations and conclude that they convey higher semantic and syntactic information compared with using the intermediate representations of NNLMs.

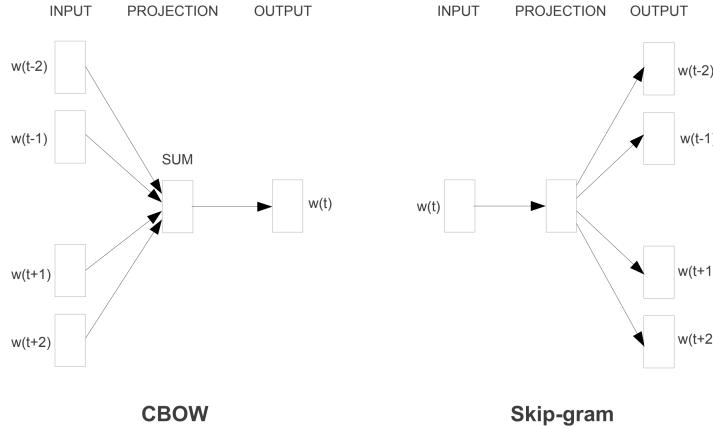


Figure 3.1: Schema of the two Word2Vec training strategies. (from [Mikolov et al. \(2013a\)](#))

### 3.2.2 GLOVe

*GloVe* (Global Vectors for Word Representation) was introduced in [Pennington et al. \(2014\)](#). Inspired by the rationale of PMI, they propose to learn a log-bilinear regression on a regularized co-occurrence matrix. Namely, they learn token representations  $x$  and  $\tilde{x}$  by optimizing the following objective<sup>2</sup>:

$$\arg \min_{x, \tilde{x}} \sum_{i,j} (\langle x_i, \tilde{x}_j \rangle + b_i + b_j - \ln P_{ij})^2$$

The authors conduct a variety of evaluation tasks and conclude that GloVe representations are more expressive than the ones obtained using CBOW or Skip-gram models.

### 3.2.3 FASTTEXT

FastText ([Bojanowski et al., 2017](#)) is an extension of Word2Vec where the token embeddings are enriched by character-level information to enhance their generalization abilities. The method is motivated by the inability of previous methods to cope with out-of-vocabulary strings, and an intent to facilitate the learning of semantic relationship for morphologically rich languages such as English where prefixes, suffixes and inflected forms are common. To that end, they represent tokens as a sum of substring representations of varying length, including the token string itself, and use the Skip-gram objective from Word2Vec at token-level over the summed representations.

As a result, their token embeddings are more performant when it comes to identifying syntactic similarities between tokens, and representations can still be provided for unseen tokens.

---

<sup>2</sup>In practice, a regularization term is used to account for rare co-occurrences.

Apart from building meaningful token representation spaces, these methods have been implemented into task-specific RNNs as a way to initialize or define look-up tables for input embeddings (Xiao et al., 2018; Muhammad et al., 2021), yielding better results especially in low-resource scenarios.

### 3.2.4 CONTEXTUAL EMBEDDINGS

Thanks to substantial progress in computational capabilities (Owens et al., 2008), it became increasingly feasible to train large neural models using self-supervised methods on large amounts of text. Following the principles of transfer learning (Pan and Yang, 2010), intermediate representations of trained neural language models based on RNNs or Transformers were used as *contextual* embeddings for task-specific model.

Peters et al. (2018a) extract the output vectors of the last LSTM layer of their frozen ELMo language model and use them as inputs for various task-specific sequential architectures. This leads to substantial improvements across most evaluations. Building upon Transformer-based language models, Devlin et al. (2019) and Radford and Narasimhan (2018) simplify this framework and propose to replace the language modeling head by a task-specific untrained linear layer. The resulting architecture is then trained as a whole on the downstream task. This second training step is usually called fine-tuning, as it is often performed with finer optimization hyper-parameters.

Martin et al. (2020) train a Transformer-based masked language model (MLM) on French data and compare its performance on downstream task with two settings: one where the pretrained model is frozen and a simple model is trained on top of it, and one where the model is fine-tuned on the downstream task. Although the frozen setting leads to slightly better performance in NER when combined with a LSTM-CRF (Panchendarajan and Amaresan, 2018), the fine-tuned model outperforms its frozen counterpart in most tasks, notably in Part-of-Speech tagging.

On word similarity tasks, these contextual representations were shown to better embed semantic similarity (Bommasani et al., 2020).

## 3.3 SENTENCE EMBEDDINGS

The contextual representations extracted from pretrained language models are also useful to evaluate sentence similarity in a zero-shot setting when building metrics from token-level similarity (Zhang et al., 2020).

However, Reimers and Gurevych (2019) show that building sentence-level representations using basic pooling strategies (e.g. using the representation of one token only or averaging the representations over the sequence) performs better when using static embeddings such as GloVe than with these contextual representations. Their work paves the way for specialized methods that build expressive sentence embeddings from neural contextual representations.

### 3.3.1 SENTENCE-BERT

Reimers and Gurevych (2019) introduce Sentence-BERT, a model based on BERT that generates sentence-level representations. They use a siamese architecture where a single model encodes two sentences into pooled representations, and a classifier predicts a label that should detect whether

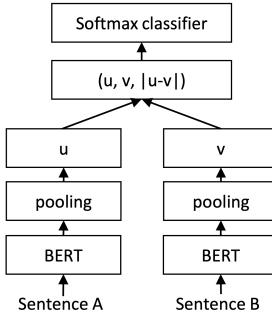


Figure 3.2: Overview of the Sentence-BERT training paradigm (taken from [Reimers and Gurevych \(2019\)](#))

two sentences share the same meaning. This siamese network is initialized with BERT or RoBERTa models and is further trained on the SNLI dataset ([Hill et al., 2016](#)), a *natural language inference* (or NLI) dataset that comprises pairs of sentences associated with a label indicating a semantic correspondance between the sentences. At inference time, the siamese network produces sentence embeddings and their *cosine similarity* is computed as a similarity score.

Cosine similarity is a vector space metric based on cosine distance that measures the angular discrepancy between two vectors. Given two vectors  $x, y \in \mathbb{R}^d$ , their cosine similarity is defined by:

$$\text{cos-sim}(x, y) = \frac{\langle x, y \rangle}{\|x\|_2 \cdot \|y\|_2}$$

Thus,  $\text{cos-sim}(x, y) \in [-1, 1]$  and higher cosine similarity values depict vectors which directions are more aligned.

They evaluate their model against concurrent work such as USE ([Cer et al., 2018](#)) and InferSent ([Conneau et al., 2017](#)) on the Sentence Textual Similarity (STS) benchmarks that provide sentence pairs with semantic similarity scores. Using the Spearman correlation metric ([Zar, 2005](#)), they show that the cosine-similarity between the Sentence-BERT representations correlates significantly better with the ground-truth similarity scores compared with other approaches.

Similar supervised methods have been applied with multilingual and/or multimodal data and base models to obtain language-agnostic sentence representations ([Feng et al., 2022](#)). Concurrently, ([Schwenk and Douze, 2017](#)) propose to learn multilingual sentence representations by training monolingual sentence encoders and decoders for machine translation. This approach was later successfully adapted to multi-modal representations ([Duquenne et al., 2023](#)).

### 3.3.2 LATENT REGULARIZATION METHODS

The contextual representations of pretrained LMs are trained in a way that implicitly forces them to capture syntactic and semantic information at token-level ([Jawahar et al., 2019](#)). Nevertheless, their geometrical structure is not constrained by any explicit process, and it remains unclear whether their final structure can be predicted *a priori*.

As we will discuss in the next section, this structure can be strongly degenerate in practice, which makes pooling nicely distributed sentence representations a more difficult task. Aware of this difficulty, several methods have been proposed to regularize the token-level representation

distributions before pooling, in order to lead to sentence embeddings for which cosine similarity is more expressive.

[Arora et al. \(2017\)](#) and [Mu and Viswanath \(2018\)](#) remove the most dominant principal components of the SVD of token-level representations, and observe improvements in the expressiveness of the resulting sentence embeddings. [Li et al. \(2020\)](#) take a different stance and train a flow-based generative model that maps the contextual token embeddings to a standard Gaussian distribution. Their BERT-flow model improves over the average-pooling baseline in most STS benchmarks. [Su et al. \(2021\)](#) propose a more direct approach and learn a *whitening* transformation, an affine mapping that gives the contextual distribution the same first two moments as the standard multi-dimensional Gaussian distribution (i.e.  $\mu = 0_d$  and  $\Sigma = I_d$ ).

### 3.3.3 CONTRASTIVE METHODS

Although siamese networks and latent regularization are both ways to improve representations taken from pretrained LMs, these methods are incompatible. As a matter of fact, the former retrains the model without constraining the geometry of the token (and sentence) embeddings, while the latter performs expensive *post-hoc* regularizations that may be non-differentiable and that are not designed to be applied at every training step.

As a result, it is difficult to both benefit from the geometrical regularity of the intermediate representations (or their *uniformity*) which allows the model to differentiate properly different samples, and from the capacity of these representations to accurately capture the information in the sentences and match similar sentences (or their *alignment*).

[Wang and Isola \(2020\)](#) design metrics to quantify alignment and uniformity in representations, and empirically show that the most expressive representation tend to optimize both metrics. They proceed to prove that *contrastive learning* objectives implicitly lead to a trade-off between alignment and uniformity.

Contrastive Learning is a representation learning framework that aims at learning discriminative embeddings by using objectives that minimize the underlying distance between equivalent items (*positive* pairs) and maximize it between unrelated items (*negative* pairs). A crucial design choice in contrastive learning methods is the definition of equivalence between the considered items, especially in the unsupervised setup. In computer vision, building positive samples is relatively easy as slight perturbations of the input features do not affect the human perception of the resulting image. This made contrastive learning particularly convenient, leading to several popular models ([Chen et al., 2020](#); [He et al., 2020](#)).

Inspired by contrastive methods, and to improve the uniformity-alignment tradeoff, [Gao et al. \(2021\)](#) develop the SimCSE models. They first apply a classical contrastive learning objective to sentence-level representations in a supervised setting. Many NLI datasets provide several sentence pairs for a given sentence, each matching it with a similar sentence or one with an opposite meaning. Hence, [Gao et al. \(2021\)](#) leverage these pairs tagged as similar as positive samples, and those tagged as dissimilar as negative samples. They also use sentences from unrelated pairs as additional negative samples.

Similarly to [Chen et al. \(2020\)](#), they leverage the InfoNCE objective ([van den Oord et al., 2019a](#)) which can be interpreted as a cross-entropy loss on in-batch classification, i.e. identifying the object in a training batch to which a representation corresponds. Formally, given an object embedding

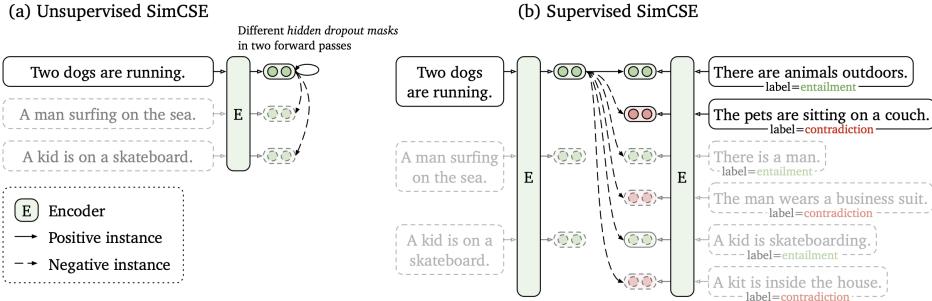


Figure 3.3: Overview of the SimCSE contrastive unsupervised and supervised frameworks (from Gao et al. (2021))

$h \in \mathbb{R}^d$ , a positive sample embedding  $h_+ \in \mathbb{R}^d$  and a series of negative samples  $h_-^i \in \mathbb{R}^d$  for  $i \in [1, N_-]$ , the InfoNCE objective based on cosine similarity is:

$$\mathcal{L}_{\text{InfoNCE}} = \mathbf{E}_{h, h_+, h_-} \left( -\log \frac{\exp \text{cos-sim}(h, h_+)}{\exp \text{cos-sim}(h, h_+) + \sum_{i=1}^{N_-} \exp \text{cos-sim}(h, h_-^i)} \right)$$

In the supervised SimCSE,  $\mathbf{h}_-$  contains representations corresponding to the dissimilar utterance as annotated in the NLI dataset, and the other sentences from the same training batch. This dissimilar utterance constitutes a *hard* negative as it should be particularly be contrasted with the target sentence, as opposed to the soft negatives that should statistically be neutral with respect to the target sentence. Gao et al. (2021) also provide an unsupervised approach where  $\mathbf{h}_-$  does not contain the hard negative from the NLI annotation. In this setting, as the positive NLI sample is not available, they need to resort to data augmentation in order to provide a similar utterance. In NLP, such augmentation can be hard to design safely, as altering one token can change the meaning of the whole sentence. Nevertheless, some attempts have been made to edit the token sequence (Xu et al. (2023) for instance). Gao et al. (2021) avoid this difficulty by computing the sentence representations twice using a different dropout filter.

Gao et al. (2021) conduct experiments and drastically outperform models in both the latent regularization and the siamese network frameworks, and they show that SimCSE achieves a much more balanced alignment-uniformity tradeoff. SimCSE paved the way for contrastive methods in sentence-level representation learning, with subsequent works that improved the negative sampling strategies (Yan et al., 2021), scaled the training process and data (Li et al., 2023), making it the state-of-the-art approach for sentence embeddings.

Overall, we have seen in this section how learning representations of textual data can bring a specific set of challenges, as naively applying general methods often fails the test of linguistic complexity. As the potential offered by machine learning methods increased, these methods have often shown to benefit from tweaks specific to the text modality that helped improve the downstream performance of NLP systems.

### *3 Representation Learning for Natural Language*

As downstream and language modeling performances become more and more intertwined, this naturally raises the question of whether neural language models are also affected internally by the same kind of phenomena, and leads us to analyzing their representations in order to characterize the impact of textual data on their geometry.

# 4

## REPRESENTATION ANALYSIS FOR NLP

With the advent of neural methods for NLP systems, it became significantly more difficult to explicitly and totally control the inner behavior of the model, which was straightforward in *symbolic* methods, as well as in purely statistical ones, to a certain extent.

Analyzing the weights and activations of the intermediate layers of neural NLP models is the purpose of the *interpretability* field (Madsen et al., 2022). In this section, we briefly depict interpretability methods, with a particular focus on works that describe the intermediate vector spaces. We also present some methods that are directly derived from these observations.

### 4.1 REPRESENTATIONS AND EMBEDDED KNOWLEDGE

In this first section, we consider approaches that view language models as grey-box systems, and identify properties of layer-wise intermediate representations without deep-diving into the underlying architectures.

#### 4.1.1 LINEAR REPRESENTATION HYPOTHESIS

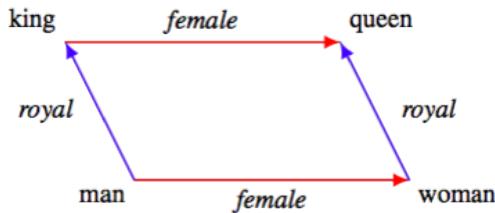


Figure 4.1: Illustration of the linear representation hypothesis as mentioned in Mikolov et al. (2013b) (taken from Ethayarajh et al. (2019))

A fundamental concept in neural model interpretability (and particularly for language models) is the linear representation hypothesis, i.e. the observation that *high-level concepts are represented linearly as directions in [the] representation space* (Park et al., 2024).

This hypothesis was famously invoked in the Word2Vec article<sup>1</sup> (Mikolov et al., 2013b), where authors argued that arithmetic operations between the representations almost perfectly captured semantic relationships. For instance, they have been shown to capture analogies, as in the notorious example:

$$x_{\text{queen}} \simeq x_{\text{king}} - x_{\text{man}} + x_{\text{woman}}$$

---

<sup>1</sup>The claim is usually attributed to Word2Vec but was initially made in an article from the same first author.

This observation was explored and justified mathematically in the context of static word embeddings (Ethayarajh et al., 2019; Allen and Hospedales, 2019). Recently, Jiang et al. (2024) showed that training a probabilistic model by optimizing cross-entropy via gradient descent theoretically pushes for a linear representation of underlying concepts, which generalizes the mathematical grounding of this hypothesis to representations of neural LMs.

Most importantly, the linear representation hypothesis implies that self-supervised neural models can be *probed* using linear methods in order to measure the extent to which a given concept is embedded in their representations.

Typically, given  $d$ -dimensional frozen hidden representations modeled by a random variable  $h$  and associated labels (respectively values)  $y$  corresponding to a given concept, the *linear probing* technique amounts to optimizing a linear (or affine) model  $f_\Psi$  for a classification (respectively regression) loss  $l$ , i.e.:

$$\Psi^* = \arg \min_{\Psi} \mathbf{E}_{h,y}(l(f_\Psi(h), y))$$

The resulting performance of the probe  $f_\Psi$  indicates to what extent the representations encode the given concept. For instance,  $h$  could encode sentence embeddings and  $y$  could be a gender label for the subject in the sentence. Then,  $l$  will be a classification loss, and  $f_\Psi$  a trainable mapping from a representation  $h_i$  to a gender prediction  $\hat{y}_i$ . In this example, the performance of the probe would indicate the amount of gender-related information that is encoded in the sentence representations.

#### 4.1.2 EMBEDDED LINGUISTIC PROPERTIES

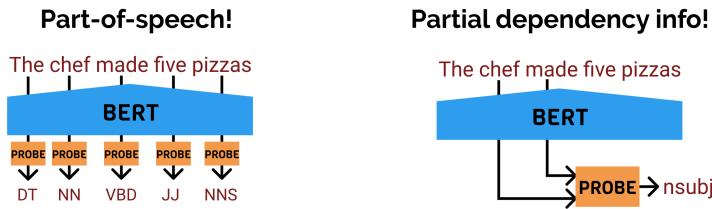


Figure 4.2: A high-level summary of probing for language models (taken from Hewitt (2019))

Early in the development of neural embedding methods, probing has been implemented to improve the understanding of how the representations capture information. Köhn (2015) successfully probe Word2Vec embeddings for basic word-level linguistic properties such as Part-of-Speech or gender. Shi et al. (2016) show that RNN-based machine translation models partly learn the syntax of the source language using probing. Adi et al. (2017) and Conneau et al. (2018a) probe sentence representations from a linguistical point of view. They focus on relatively basic properties, such as the sentence length, or the belonging of a given token. Liu et al. (2019a) extend this idea to token-level representations, and propose to use usual downstream tasks, such as POS tagging or Named Entity Recognition, as probing tasks. They train layer-wise probes and show that the necessary linguistic information is better contained in deeper layers.

These probing tasks have been used to better understand the layer-wise organization of neural LMs. Peters et al. (2018b) show that the first layers focus on local syntax, while deeper layers focus on semantic content. Tenney et al. (2019) further decompose the roles of layers and find that

the BERT model unsupervisedly mimics traditional NLP pipelines, i.e. it first latently parses the sentences before identifying coreferences and semantic relations. [Jawahar et al. \(2019\)](#) come to similar conclusions, but also notice that BERT is able to capture linguistic hierarchy and to mimic tree-like structures at representation level.

#### 4.1.3 EMBEDDED WORLD KNOWLEDGE

Several works have probed models in search for world knowledge, in order to quantify the ability of LMs to learn meaningful representations of physical objects beyond linguistic semantic.

[Gupta et al. \(2015\)](#) show that static representations contain referential information about named entities. For instance, it is possible to roughly estimate population count, geolocation and other properties from city or country names using a basic probe.

Subsequently, different works have studied temporal knowledge ([Thukral et al., 2021; Caselli et al., 2022](#)), auditive representations ([Ngo and Kim, 2024](#)) or factual knowledge ([Youssef et al., 2023](#)), among others. In this work, we focus on geographical knowledge, and wonder whether geo-localisation information is embedded in the representations of geographical entities.

[Louwerse and Benesh \(2012\)](#) show that coordinates of places in the Middle-Earth can be predicted by just using the co-occurrence matrix extracted from the Lord of the Rings novels. [Faisal and Anastasopoulos \(2023\)](#) build networks from geographical representations based on monolingual and multilingual models of different sizes. They show that all models embed more accurate geographical representations for countries of the Global North. This geographical discrepancy can be explained by biases that are inherent to the datasets used for pretraining [Faisal et al. \(2022\)](#).

Recently, [Gurnee and Tegmark \(2024\)](#) probed large language models from the Llama-2 suite ([Touvron et al., 2023](#)) to extract coordinates of prompted locations from hidden representations across layers. They show that models ranging from 7B to 70B parameters are able to convincingly embed geographical coordinates on a world map when representing basic prompts. They prove that scaling up the model size systematically leads to better performance in coordinates prediction.

[Peters et al. \(2019\)](#) propose to use the probe loss as an auxiliary objective during training, explicitly encoding chosen properties in the representations. They argue that the resulting model is more performant on downstream tasks after fine-tuning.

A concurrent line-of-work rather focuses on the analysis of data-inherent socio-cultural biases in models at representation-level. Instead of probing encyclopedic world knowledge, these methods evaluate the extent to which language models are contaminated by the stereotypical views that belong in natural language, or by statistical imbalances that may implicitly reinforce these views in these models.

[Zhao et al. \(2018a\)](#) study gender bias in co-reference resolution systems. Co-reference resolution is a linguistic task that consists in indentifying the object of a reference (a pronoun referring to a past name for instance) that could potentially seem ambiguous. Analyzing the errors made by a coreference resolution system can provide insights about its underlying representations of concepts. For instance, let us consider the sentence “*the physician hired the secretary because she was overwhelmed with clients*”. A model that wrongly predicts that *she* refers to *secretary*, but would predict the correct reference when replacing *she* with *he*, likely suffers from stereotypical gender biases. They propose to quantify gender bias more directly in static embeddings by measuring the

cosine similarity between the gender token (e.g. *female*) and attributes like profession tokens (e.g. *colonel*) (Zhao et al., 2018b).

Zhao et al. (2018b) go further and introduce auxiliary objectives to the GloVe method that help minimize their bias metric, which implies minimizing the alignment between some attributes and gender in the token-level representation space. In a similar effort, Ravfogel et al. (2020) post-edit statistical embeddings by iteratively projecting them on the orthogonal direction of a trained linear probe. Intuitively, training a new probe on these projected embeddings should be more difficult as a gender-sensitive direction has been suppressed from the latent space. Iskander et al. (2023) extend this idea to more complex probes and introduce a gradient-based method of which Ravfogel et al. (2020) is a particular case.

## 4.2 ANALYSIS OF SELF-ATTENTION

After the emergence of Transformer-based models, many interpretability works have focused on the multi-head self-attention layers. These layers differ from the other linear layers as they model the interactions between tokens.

### 4.2.1 PROPERTIES OF SELF-ATTENTION MAPS

When attention was introduced as a way to improve machine translation neural systems (Bahdanau et al., 2015), it was shown that cross-attention maps were implicitly modeling token-to-token cross-lingual mappings.

The self-attention patterns of BERT have been analyzed in numerous works. Clark et al. (2019b) describe head-specific patterns, such as attending to punctuation, special tokens used for self-supervision (e.g. [SEP] or [CLS]), and previous or next tokens. They also identify a pattern with the entropy of head-level attention maps, where lower layers contain high-entropy maps that roughly average the representations of all token positions. They proceed by identifying specific heads that specialize in extracting a specific type of linguistic phenomenon through attention maps. They use attention maps and GloVe embeddings as an input to a dependency parsing classifier, and obtain satisfying results, showing that attention maps latently perform operations that are related with parsing.

Voita et al. (2019) conduct a similar analysis and find that pruning non-specialized heads does not significantly affect performance in Transformer-based machine translation. Vig and Belinkov (2019) concurrently identify head-specific patterns on GPT-2 (Radford et al., 2019), and add that deeper layers tend to encode longer-term dependencies than first layers.

Overall, these works show that self-attention maps in language models tend to be sparse (or low-entropy), which can make them easily interpretable in some cases. Quantitatively, head-level probing tasks and downstream evaluation allows to measure and characterize their specialization from a linguistic point of view.

### 4.2.2 THE LOGIT LENS

Although the methods mentioned above are efficient at measuring the quantitative linguistic performance of specific parts of Transformer models, they depend on a biased evaluation protocol,

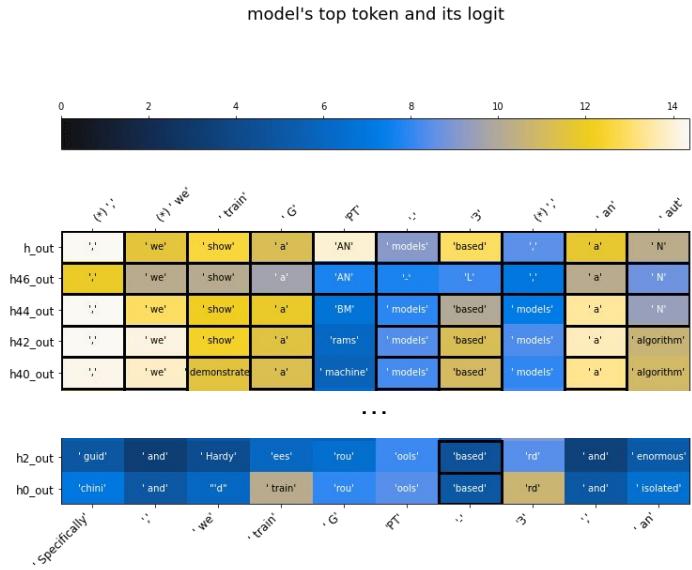


Figure 4.3: Layer-wise diagram obtained with the logit lens technique (taken from [Nostalgebraist \(2020\)](#))

as the tasks and measurements are made with a specific angle of study. This remark led to the elaboration of techniques that automatically discover interpretable patterns.

[Nostalgebraist \(2020\)](#) introduce the concept of *logit lens* ([Nostalgebraist, 2020](#)), a projection technique that allows to extract token-related interpretations from intermediate representations in Transformer blocks. They observe that the residual connections of Transformer architectures create direct paths to the last hidden representation, which is then simply projected linearly on a  $V$  dimensional logit space by the language modeling head. Hence, they hypothesize that the structures of the representations that go through these residual connections are suited for the language modeling head, implying that projecting them through this layer will produce meaningful outputs in the logit space. They mostly use this observation to analyze the stacked layers as improving predictors and to display their intermediate predictions.

[Elhage et al. \(2021\)](#) develop a more elaborate framework around a similar idea, and identify *circuits* in Transformer models, i.e. paths through the layers via a residual stream that processes specific features with each linear operation

[Prakash and Lee \(2023\)](#) use the logit lens technique on large LMs to investigate the semantic changes that occur after bias mitigation techniques ([Ravfogel et al., 2020](#)). [Dar et al. \(2023\)](#) design an equivalent framework to analyze model weights in the token space.

These initiatives tend to show that Transformer layers all contribute to the final prediction through a residual stream, where some heads sparsely process input representations in an interpretable way. Once more, these sparse implicit operations happen in linear subspaces that can be analyzed in token space via basic operations.

### 4.2.3 QKV GEOMETRY

To the best of our knowledge, few works have specifically studied the vector distributions of attention-level representations  $Q^h$ ,  $K^h$  and  $V^h$ .

Recently, [Devoto et al. \(2024\)](#) observed that the  $L_2$  norm of the  $K^h$  representations could be used as a proxy for subsequent attention weights. They derive an efficient KV cache compression scheme from this observation, proving that  $K^h$  representations with the highest  $L_2$  norm can actually be discarded at inference time without significant performance loss.

## 4.3 REPRESENTATION DEGENERATION

Representation Degeneration is a phenomenon in which pretrained models tend to adopt low-entropy singular value distributions ([Jing et al., 2022](#)). In other words, the singular value distributions of the representations of affected models are particularly imbalanced, which implies that they can be efficiently approximated in a lower-dimensional subspace.

In language modeling, representation degeneration has been studied from various points of view, and specificities related to the (Zipfian) distribution of textual data have been put forward to explain the forms of degeneration that were observed at various levels in models. Hence, we argue that representation degeneration can be included in the field of interpretability, as it depicts distortions that may be explained by general properties of natural language.

We summarize the works that cope with this question in the following sections.

### 4.3.1 A GENERAL PHENOMENON

[Hua et al. \(2021\)](#) identify dimensional collapse as a potential caveat of self-supervised models in general. They suggest that features tend to naturally collapse towards highly dependent patterns. To measure this phenomenon, they track several metrics for the generated representations. They estimate the rank of the space spanned by the representations by computing the singular values of a representation set, and show that this estimated rank tends to decrease in training. They also measure the average absolute correlation scores across features, and show that this correlation does not naturally decrease.

A way to circumvent this phenomenon is to penalize feature similitude, which they implement through a covariance regularization term. [Tian et al. \(2021\)](#) show that architectural tricks such as weight decay ([Krogh and Hertz, 1991](#)) or exponential moving averaging ([Morales-Brotóns et al., 2024](#)) can mitigate dimensional collapse.

Another natural approach towards mitigating dimensional collapse is contrastive learning, as it implicitly improves the uniformity of latent spaces ([Wang and Isola, 2020](#)). Nevertheless, [Jing et al. \(2022\)](#) show that for a broad range of data augmentation techniques used when generating positive samples, contrastive methods lead to dimensional collapse. They design architectural tricks and improved data augmentation methods to alleviate this issue.

Most importantly, in agreement with [Wang and Isola \(2020\)](#), these works notice that uniformity in latent spaces is correlated with performance, and that mitigating dimensional collapse leads to better models in general.

### 4.3.2 DEGENERATION IN NLP

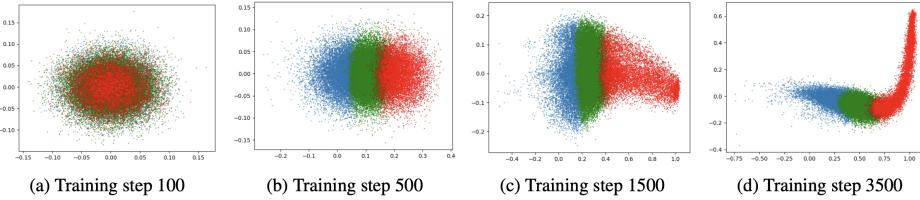


Figure 4.4: Visualization of the token embeddings of a language model during training, projected onto the first two components of their SVD. Red, green, and blue points represent rare, medium, and frequent groups respectively (taken from [Yu et al. \(2022\)](#)). These plots show the emergence of a degeneration phenomenon and its apparent correlation with token frequency.

In NLP, *degeneration* is a term that has been used in various works to convey different meanings.

Firstly, it has been used to describe erratic behaviors of language models at inference time, with notable examples of repeated tokens or phrases, and of strong hallucinations. [Holtzman et al. \(2020b\)](#), describe the *neural text degeneration* phenomenon in depth. Through extensive analysis of GPT-2 generated sequences, they show that pre-existing decoding methods, such as beam search ([Freitag and Al-Onaizan, 2017](#)), temperature sampling ([Ackley et al., 1985](#)), or top-k sampling ([Radford et al., 2019](#)), all lead to undesirable outputs. They argue that “*natural language does not maximize probability*”, and that decoding methods that aim at purely minimizing the perplexity of the model on the generated text are bound to lead to less human-like samples. They introduce nucleus sampling, a strategy that thresholds the token distribution based on its cumulated distribution function (CDF). [Welleck et al. \(2020\)](#) suggest an alternative method that consists in continue the training of the model with an auxiliary *unlikelihood* loss that explicitly penalizes repetition and induction-based hallucinations.

[Finlayson et al. \(2024\)](#) investigate the implication of the softmax bottleneck (see [Section 2.4.1](#)) in this kind of degeneration, and show that the low-dimensional linearity of the language modeling head may introduce artifacts in the next-token probability distributions. As a matter of fact, they argue that when the hidden dimension  $d_m$  is smaller than the vocabulary size  $V$ , the logits vector lie in a space spanned by the language modeling head, which is a relatively low-dimensional manifold of  $\mathbb{R}^V$ . As a result, projecting these logits on  $\Delta^V$  using the softmax function leads to spurious non-zero probabilities for tokens that would be null otherwise, as the low-dimensional manifold cannot be mapped to  $\Delta^V$  in a surjective manner. This phenomenon explains the efficiency of truncation strategies such as nucleus sampling, as they will discard most of the spurious non-null probability tokens when the error caused by the softmax bottleneck is not too significant.

[Sharma and Kaplan \(2022\)](#) also link the performance of language models to the dimensionality of hidden representations and of the data distribution itself. They argue that the parameters of the scaling laws (see [Section 2.3.7](#)) can be estimated through a study of the *intrinsic dimension* or ID ([Tulchinskii et al., 2023](#)) of the data manifold. [Tulchinskii et al. \(2023\)](#) use *persistence homology dimension* ([Adams et al., 2020](#)), a fractal dimension estimation metric based on minimal spanning trees, to measure the ID of embedding distribution of artificially generated texts and human text. They find that artificial samples tend to have a lower intrinsic dimensionality than human samples.

[Sharma and Kaplan \(2022\)](#) use a similar metric to estimate the data dimensionality, and to verify their approximation of the scaling laws based on this measure.

The notion of *degeneration* has also been used to extend the exploration of representational degeneration to the specific case of natural language modeling. A seminal work in that field is *Representation Degeneration Problem in Training Natural Language Generation Models* ([Gao et al., 2019b](#)). In this paper, the authors underline a connection between the distortion that can be observed in both static and contextual embeddings spaces, and the unbalanced Zipf law that appears in natural language.

They elaborate their claim around the example of an unused token, that is a token that belongs in the vocabulary  $\mathcal{V}$  of the language model, but that does not appear in the training sequences. This can happen if the tokenizer and the model are trained on different datasets. Let's consider the case where there is only one such token  $w_u$ , and all the model  $\theta$  parameters are fixed except for the parameters of the language modeling head  $W_o$ , of rows  $(o_i)_{i \in [1, V]} \in \mathbb{R}^{V \times d_m}$ . The authors focus on the parameters  $o_u$ , which are updated during training to optimize the cross-entropy loss, in a *consistent* way where the output logit for  $w_u$  must be as low as possible as  $w_u$  never appears. More formally, if a set of  $\Upsilon$  contextual embeddings from the frozen model are noted  $(h_i)_{i \in [1, \Upsilon]}$ , optimizing cross-entropy with respect to  $o_u$  amounts to the following problem:

$$\arg \max_{o_u} \frac{1}{\Upsilon} \sum_{i=1}^{\Upsilon} \left( \log \frac{\exp \langle h_i, o_{w_i} \rangle}{\sum_{j \in [1, V] \setminus u} \exp \langle h_i, o_j \rangle + \exp \langle h_i, o_u \rangle} \right)$$

which can be reduced by introducing a constant  $C_i$ , to :

$$\arg \min_{o_u} \frac{1}{\Upsilon} \sum_{i=1}^{\Upsilon} \log(C_i + \exp \langle h_i, o_u \rangle) \quad (4.1)$$

[Equation \(4.1\)](#) shows that the parameters corresponding to  $w_u$  in the language modeling head are trained to minimize an average metric on the whole dataset, that naturally pushes the scalar products  $\langle h_i, o_u \rangle$  towards negative values. The authors show that this optimization problem tends to lead to a degenerate solution where  $o_u$  is pushed away from the convex hull of the representations  $\mathbf{h}$ .

They extend this observation to rare tokens through more complex analysis, and deduce that training a model using cross-entropy on data that has an underlying Zipfian distribution naturally leads to a distortion of the output latent space. Frequency-based degeneration was also observed in [Zhou et al. \(2021a\)](#), who show that the representational geometry of tokens is highly dependent on their frequency, and that rare tokens are less distinguishable in latent spaces. They argue that this difference can explain biases in downstream performance, using the example of geographical knowledge discrepancy between high-frequency and low-frequency location names.

To overcome this limitation, [Yu et al. \(2022\)](#) train a language model using adaptive gradient gating, which regularizes the gradients related to low-frequency tokens. [Meister et al. \(2023\)](#) propose a simpler approach where a bias is added to the output of the language modeling head. This bias is manually set to  $b_i = \log(f_{w_i})$ , where  $f_{w_i}$  is the unigram frequency of token  $w_i$ , which allows the model to only model a residual contribution to the contextual probability, making it less sensitive to token frequency.

## 4.3.3 ANISOTROPY &amp; OUTLIER DIMENSIONS

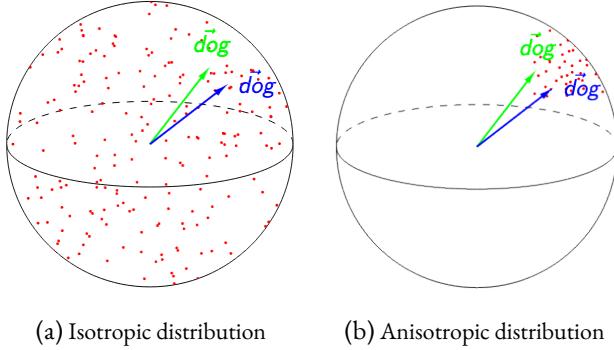


Figure 4.5: Schema of anisotropy in a token embedding space. (taken from [Etheyarajh \(2019\)](#))

A specific framing of representation degeneration in language modeling has taken the form of *anisotropy*, that is of non-uniformity of the distribution of angular components in language model representations.

To the best of our knowledge, [Etheyarajh \(2019\)](#) is the first to describe this phenomenon for contextual representations in natural language. He defines anisotropy as the average cosine similarity between two randomly sampled representations at a given layer. Formally, if  $\Upsilon$  outputs representations  $\mathbf{h}^i$  are uniformly sampled from the  $i$ -th layer of a neural network, then the anisotropy level of that layer can be measured as:

$$\mathcal{A}_{cos}(\phi_\theta, i) = \frac{1}{\Upsilon^2 - \Upsilon} \sum_{n \neq m} \frac{\langle h_n^i, h_m^i \rangle}{\|h_n^i\|_2 \cdot \|h_m^i\|_2} \quad (4.2)$$

$\mathcal{A}_{cos}(\phi_\theta, i)$  takes a value in  $[-1, 1]$ , and the distribution of  $\mathbf{h}^i$  can be described as anisotropic when  $|\mathcal{A}_{cos}|$  takes higher values.

They measure anisotropy on hidden representations obtained across different sentences using this in ELMo ([Peters et al., 2018a](#)), BERT ([Devlin et al., 2019](#)) and GPT ([Radford and Narasimhan, 2018](#)). They find that hidden representations of these models tend to be anisotropic, especially on the deeper layers, and notice that the anisotropy level of GPT takes extreme values (up to 0.97). In simpler terms, when sampling two output token representations of GPT at random (across documents or not), the cosine similarity between these will be 0.97 in average. This is unexpected, as a uniform distribution would likely yield orthogonal vectors, especially in high-dimension settings.

[Mu and Viswanath \(2018\)](#) explore anisotropy from the perspective of singular value decomposition. They measure an average similarity between each singular vector  $u^i \in U^i$  and each hidden representation  $h^i$ , and by comparing the maximum average similarity (the singular component that is most similar to hidden representations) and the minimum average similarity (the singular component that is least similar to hidden representations). In an isotropic distribution, these

similarity levels should be closer as each singular component should almost equally explain all hidden representations. The precise measure is done as such:

$$\mathcal{A}_{SV}(\phi_\theta, i) = \frac{\min_{u \in U} F(u)}{\max_{u \in U} F(u)}$$

where  $F(u) = \frac{1}{T} \sum_{t=1}^T \exp(\langle u^i, h_t^i \rangle)$ .

Here,  $\mathcal{A}_{SV}(\phi_\theta, i) \in [0, 1]$  and the distribution is characterized as anisotropic when it is significantly smaller than 1. They use this measure to show that static embeddings to be anisotropic, and they propose a basic truncation of top singular components to improve their anisotropy. They show that the resulting isotropic static embeddings lead to better performance on downstream tasks.

[Wang et al. \(2020a\)](#) take advantage of this spectral viewpoint and use spectrum control to mitigate anisotropy during training. They add a penalization term in the training objective that sets a target for the singular value distribution of the representations. They show that this regularization leads to better performance for language models in monolingual and multilingual setups, including for downstream performance.

[Rajaei and Pilehvar \(2021\)](#) use this metric on pretrained language models and come to similar conclusions to [Ethayarajh \(2019\)](#). They observe that the spaces spanned by  $\mathbf{h}^i$  vectors are split into clusters which may cause anisotropy. They cluster the representations using unsupervised techniques, and compute the clusters barycenters before subtracting them from their representations. They show that this post-processing step improves both the STS performance and the results on the NLI benchmarks, while successfully mitigating anisotropy. [Rajaei and Pilehvar \(2022\)](#) extend the anisotropy diagnosis to the multilingual version of BERT. [Hämmerl et al. \(2023a\)](#) later propose several post-processing techniques to mitigate anisotropy, and show that they improve the performance of sentence embeddings.

[Bić et al. \(2021\)](#) prove the existence of a connection between this anisotropy phenomenon and a degeneration similar to the one described in [Gao et al. \(2019b\)](#) (see [Section 4.3.2](#)). They show that training language models through cross-entropy optimization leads to a progressive drift of the embeddings towards a common direction, and that such a drift can be measured as an increase in anisotropy levels as a non-centered representation distribution is naturally contained in a narrower region of the space. They remove this common component by subtracting the average representation, and show that this simple operation substantially reduces the anisotropy of the distributions according to  $\mathcal{A}_{cos}$ . Crucially, this connection allows them to formally bridge frequency-related distortions of the embedding space and anisotropy.

Several approaches have taken an opposite stance and have suggested that anisotropy was unharful and could even lead to better performance. [Ait-Saada and Nadif \(2023\)](#) show that anisotropy does not affect clustering performance in sentence representations. [Rudman and Eickhoff \(2024a\)](#) propose to add an *isotropy* penalization in the training objective of language models when fine-tuning on downstream tasks, and show slight performance improvements. Finally, [Machina and Mercer \(2024\)](#) show that anisotropy does not automatically appear in the output layers of larger language models.

A problem that is closely related to anisotropy is the existence of outlier dimensions in the representations of language models, which are also related with token frequency ([Puccetti et al., 2022](#)).

However, outlier dimensions are a form of anisotropy themselves, and most of the mitigation techniques that have been proposed in this subfield are related with quantization issues (Ahmadian et al., 2023; Nrusimha et al., 2024). Hence, we choose not to include this literature in detail in this section.

The aforementioned works cover diverse topics and aim for different objectives. Nevertheless, they tend to focus on the geometrical interpretability of language models in the representational space. These works show that viewing the activations of these models as vectors that should lie in meaningful spaces can bring performance improvements when leveraged properly. Understanding the geometry of these spaces with respect to data-related properties allows to better understand the behavior of a model, but also to point out limitations that the language modeling paradigm brings, when these spaces are not uniform or when they are too constrained to lead to optimal predictions.



## PART II

### ANALYSIS OF THE REPRESENTATIONS OF LANGUAGE MODELS

This part studies the representation degeneration phenomenon from various perspectives. We extend the works mentioned in [Chapter 4](#) by exploring these questions at larger model scales, for other modalities and using different analysis tools. This provides valuable insights towards improving the language modeling paradigm, that we leverage in [Part III](#).



# 5

## ON THE SCALING LAWS OF GEOGRAPHICAL REPRESENTATION IN LANGUAGE MODELS

In recent years, *scale* has been deemed a crucial factor towards improving language models, whether as an argument in favor of larger and more diverse training datasets, or as a motivation for training larger models. The hypothesis that scaling up models could solve current issues in NLP is both a consequence of the *bitter lesson* claim (Sutton, 2019) which states that more computation leads to more effectiveness, and of the scaling laws (see Section 2.3.7).

In this chapter, we explore this claim from a representation analysis perspective, and propose to quantify the informativeness of learnt intermediate representations in language models of increasing size by probing a specific kind of information: geographical knowledge.

Among probing tasks, several works have focused on geographical representations that are implicitly embedded in language models, and the bias that they inherit from the training data (Louwerse and Benesh, 2012; Faisal and Anastopoulos, 2023; Faisal et al., 2022).

As discussed in Section 4.3.2, imbalanced frequency distributions of geographical references in pretraining data causes distortions in the representational space (Zhou et al., 2021b). These distortions lead to a loss in the models’ ability to differentiate between under-represented locations.

Recently, Gurnee and Tegmark (2024) have probed large language models from the Llama-2 suite (Touvron et al., 2023) to extract coordinates of prompted locations from hidden representations across layers. They show that models ranging from 7B to 70B parameters are able to convincingly embed geographical coordinates on a world map when representing basic prompts.

We propose to extend the analysis by Gurnee and Tegmark (2024) to smaller language models, in order to observe how scale affects the ability of models to implicitly embed geographical information from raw training data. We show that such ability consistently improves with model size, and that even tiny models are able to produce visually meaningful world maps.

We make several contributions:

- We show that geographical information can be extracted to a certain extent from representations at every model scale;
- We observe that larger models are more geographically biased than their smaller counterparts;
- We find that the performance of models in terms of geographical probing is correlated with the frequency of corresponding country names in the training data.

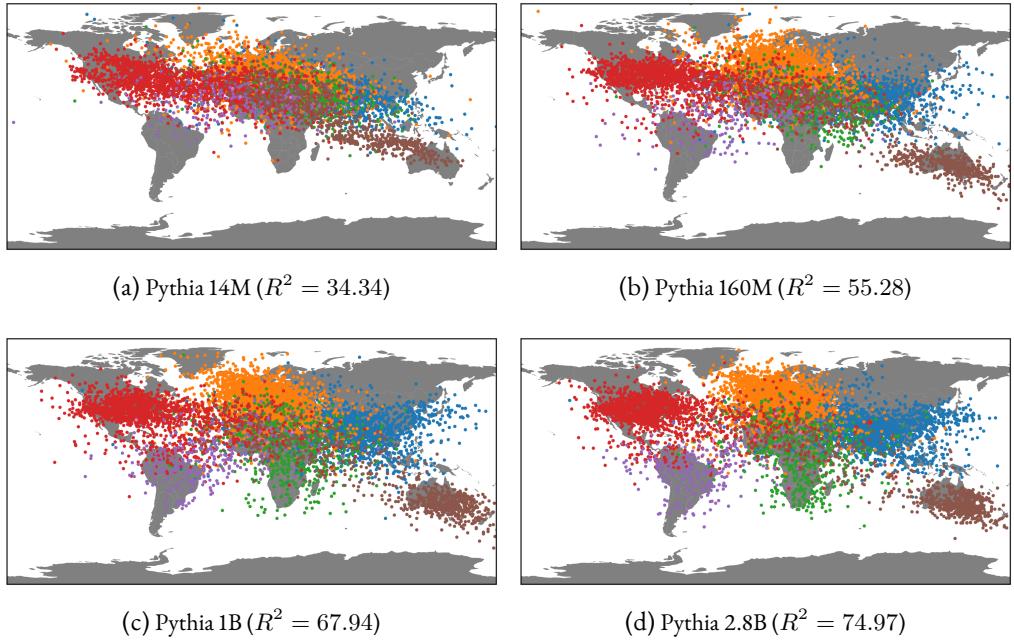


Figure 5.1: Predicted coordinates of test set instances for different model sizes. Each color represents a different continent.

## 5.1 SCALING LAWS OF GEOGRAPHICAL PROBING

In this section, we train geographical probes for a wide variety of models at different scales.

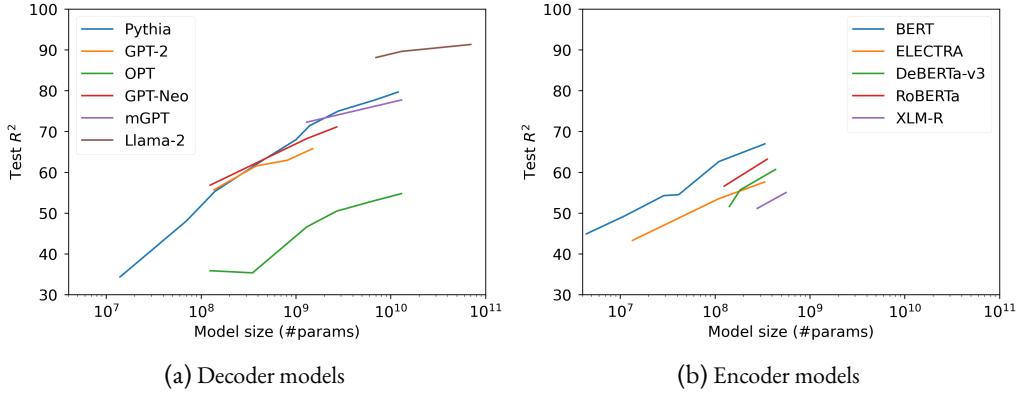
### 5.1.1 METHODOLOGY

We use the World dataset from [Gurnee and Tegmark \(2024\)](#) as a geographical data source. It contains 39,504 location names from the whole world along with corresponding longitude and latitude. We use the same train-test split strategy as in the original article, thus keeping 20% of samples for testing purposes.

For each location name  $X$ , we prompt models with the text: “*Where is X in the world?*”. We then infer with a given model on the whole dataset, and use the last token belonging to the entity  $X$  as the model’s representation. To follow the linear probing paradigm used in [Gurnee and Tegmark \(2024\)](#), we train a Ridge linear regressor ([Hoerl and Kennard, 1970](#)) to predict latitude and longitude based on the model’s representations. We then measure the probe’s performance on the test set using the  $R^2$  correlation coefficient.

### 5.1.2 RESULTS

In [Figure 5.1](#), we display the predictions of the probe for the most performant layer, which is generally the last one. We observe that geographical information can be extracted from models even for a very small parameter count. The performance of the probes seem to increase with the model size.

Figure 5.2: Evolution of the  $R^2$  coefficient on the test set for various model suites.

We show in Figure 5.2 that the performance of language models evolves consistently with model size, regardless of the architecture. We validate this property on several decoder model families: GPT-2 (Radford et al., 2019), OPT (Zhang et al., 2022), Pythia (Biderman et al., 2023a), GPT-Neo (Black et al., 2021), the multilingual mGPT (Shliazko et al., 2024b), and Llama-2 (Touvron et al., 2023). We also display results for several encoder models: BERT (Devlin et al., 2019; Turc et al., 2019), RoBERTa (Liu et al., 2019b), ELECTRA (Clark et al., 2020), and DeBERTa-v3 (He et al., 2021). This property also applies for encoder models, for which we notice that the BERT suite unexpectedly outperforms its counterparts. The performance of encoder models is comparable with the one of equivalent decoder models. We can underline the fact that BERT-Large (336M parameters) is as accurate as the three times larger Pythia-1B.

Interestingly, the multilingual XLM-R (Conneau et al., 2019) underperforms its counterparts, even though multilingual data must have increased the training data’s geographical diversity to some extent (Faisal and Anastasopoulos, 2021). The mGPT suite also slightly underperforms Pythia models at equivalent model sizes.

We verified that the better performance of larger models was not solely related with the ability of the probes to extract better patterns from their higher-dimensionality hidden representations. We achieved this by concatenating representations with themselves to increase dimensionality without introducing novel knowledge. It led to slightly worse performance for all tested models, thus showing that performance was not a consequence of dimensionality alone.

## 5.2 GEOGRAPHICAL BIAS AND SCALE

In Figure 5.1, it seems at first glance that as the model size increases, the predictions tend to be more accurate for locations of the Southern Hemisphere. In this section, we propose to quantify this hypothesized behavior by measuring the bias across countries and continents for various scales. We also correlate the models’ accuracy with both lexical and geographical factors.

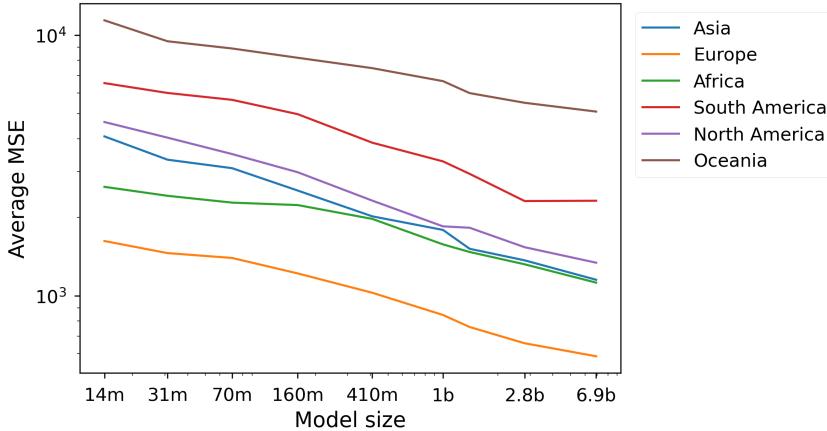


Figure 5.3: Average MSE by continent for different sizes in the Pythia suite.

### 5.2.1 MEASURING BIAS

We group probe performance as measured by mean-squared error (MSE) on predicted coordinates, and average measures by continent in Figure 5.3. While we notice that the performance increases consistently for every continent, we do not observe a significant reduction in the performance gap across continents as model size increases.

To measure the heterogeneity of the probing performance of language models across countries, we use the Gini coefficient (Gini, 1912) that is widely used in economics. Given a series of observed variables  $(x_i)_{i \in [1, N]}$ , the Gini coefficient is defined as:

$$Gini(x) = \frac{\sum_{i,j \in [1, N]} |x_i - x_j|}{N \cdot \sum_{i=1}^N x_i}$$

A Gini coefficient of 1 reflects perfect heterogeneity, while a Gini of 0 implies perfect homogeneity.

Figure 5.4 shows that the larger the model is, the more heterogeneous the probe performance is across countries and continents. This contradicts the impression given by Figure 5.1, and shows that scale does not solve the geographical discrepancy caused by bias inherent to the training data.

In Figure 5.5, we locally average log-MSE on a World map, and report results agglomerated according to latitude and longitude. We clearly observe that the model performs poorly in Oceania, South Asia and South America. We also see that the error is minimal around the latitude of North America and Europe, while it increases in the Southern Hemisphere.

### 5.2.2 IDENTIFYING SOURCES OF BIAS

We attempt to correlate the performance of our geographical probes with several factors. First, the dataset from (Gurnee and Tegmark, 2024) provides each location with an estimate of the corresponding population count when relevant. We also consider training data distribution as a

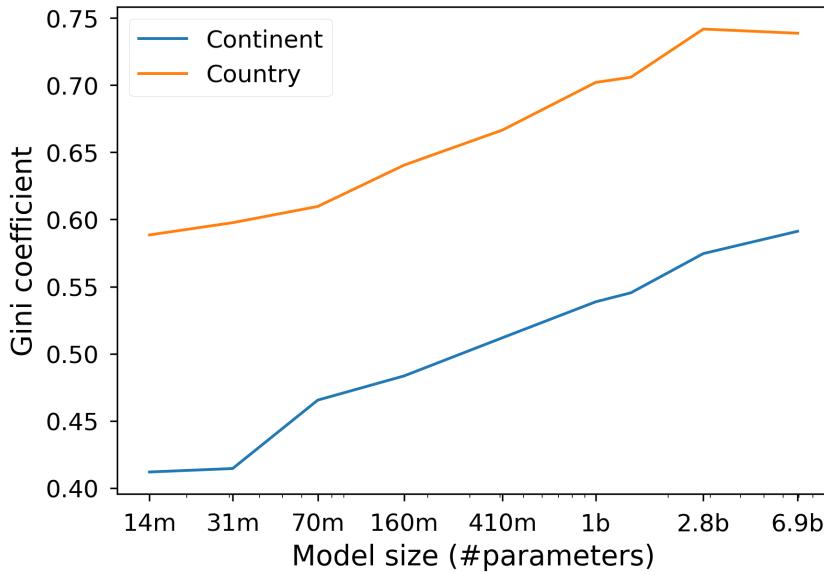


Figure 5.4: Gini coefficients of MSE on the test set averaged by country or by continent, as model size increases.

potential factor of heterogeneity. Finally, we consider latitude and longitude as potential factors of bias.

To account for training data distribution, we look for exact string matches of country names from the [Gurnee and Tegmark \(2024\)](#) dataset in an extract of The Pile ([Gao et al., 2020](#)) containing 3.5 million samples<sup>1</sup>. We select this dataset as it was used to pretrain the models from the Pythia suite ([Biderman et al., 2023a](#)) we evaluate in this section. We find 15 million matches, covering 98% of the countries of the dataset.

We do not count occurrences of location names directly, as matching locations on the basis of their names does not account for named entity ambiguity. An example of ambiguous location name is *Fully*, which is a town in Switzerland. An exact match strategy overestimates by large margins the occurrence count of this location, because of the corresponding English word *fully*. Disambiguation techniques have been designed ([Hoffart et al., 2011](#); [Orr et al., 2020](#)), but we prefer to avoid the risk of bias propagation and the cost of using such methods on a large corpus.

We display Pearson correlations between each of the aforementioned factors and the entity-level MSE for each model size in [Figure 5.6](#). As in [Figure 5.1a](#), we observe that the error on coordinates prediction is negatively correlated with the latitude, i.e. southern locations are less accurately identified. This correlation slowly decays as the model size increases. Meanwhile, longitude seems to be mildly correlated with the probe performance.

Interestingly, the population count is not correlated with the error level. The occurrence count of the location country is negatively correlated with the error level, thus showing that the more country names appear in the training dataset, the more the probes are able to recover coordinates

<sup>1</sup>[https://huggingface.co/datasets/ola13/small-the\\_pile](https://huggingface.co/datasets/ola13/small-the_pile)

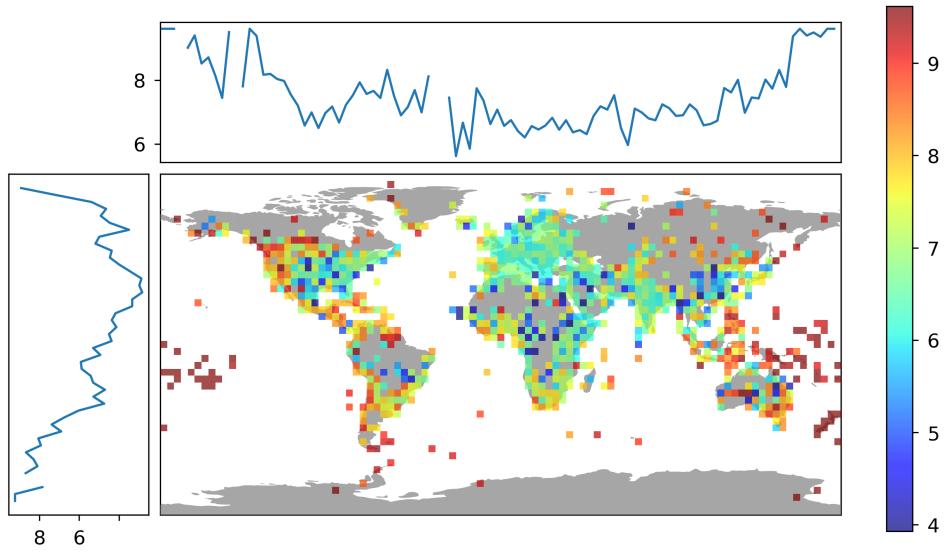


Figure 5.5: Test log-MSE for Pythia-1B as plotted on a World map.

from locations in these countries. However, this correlation is mild and even below the significance threshold for the smallest model.

We also measure the correlation between country occurrences and other metrics to account for the bias inherent to the data. We observe that country name occurrences are positively correlated with latitude with a p-value of 0.06, and not correlated with the longitude. More importantly, the population count of a country and the count of this country name in the data are heavily correlated (factor of +0.52 and p-value of 3e-23). Thus, even though the data seems guided by demographic factors, this is not the case of the model’s representations.

### 5.3 DISCUSSION

We believe that quantifying sociocultural bias in representations of language models and pretraining datasets allows to better understand the roots of the biases that can be observed during generation.

[Bender et al. \(2021\)](#) discuss the relevance of scaling models to ever larger magnitudes, with regard to environmental and financial costs. Our study shows that scale can also increase language modeling bias when it comes to geographical representation, given a pretraining dataset. We advocate in favor of measuring and mitigating bias in pretraining datasets to avoid scaling bias along with performance.

### CONCLUSION

In this chapter, we show that a wide variety of language models, varying in architecture and sizes, implicitly embed geographical data to some extent. As we consider larger models, the performance of geographical probes consistently increases towards levels shown in [Gurnee and Tegmark \(2024\)](#).

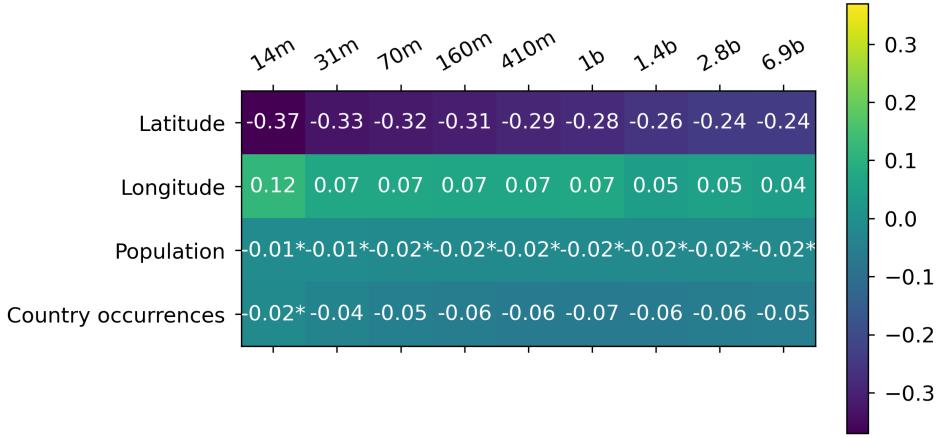


Figure 5.6: Pearson correlation coefficients of various factors with location-wise MSE, for several Pythia model sizes. \*: Tests that yielded p-values above 0.05.

We show numerically that the geographical probe performance is correlated with latitude across model sizes, but also with the number of occurrence of corresponding country names in the pretraining data. Conversely, the population count of the location seems uncorrelated with the probe performance. This indicates that a minority of people benefit from better geographical understanding when using language models, which does not maximize the social utility of these systems.

While it may initially seem that this performance increase mitigates heterogeneity between Southern and Northern countries, we actually show that larger models tend to be more biased according to the Gini coefficient taken on prediction error. This tends to show that scaling language models can amplify discrepancies in their geographical knowledge.

This chapter shows that scaling up the size of language models does not reduce their dependency to the data distribution, and can even lead to more biased behaviours. As a result, this incentivizes research towards training paradigms that reduce the dependency to the data distribution, so that the model size can be scaled harmlessly and so that the model can intrinsically be aligned with ethical guidelines easily.



# 6 STUDYING LANGUAGE MODEL SATURATION VIA THE SOFTMAX BOTTLENECK

In Chapter 5, we explored the knowledge bias phenomenon as the size of the models increases. Zhou et al. (2021b) describe the connection between knowledge bias and frequency-based distortions such as anisotropy (see Section 4.3). In this chapter, we explore these geometrical distortions for various model sizes, with a particular focus on training dynamics.

In Section 4.3, we discussed the representation degeneration phenomenon from the various perspectives offered by the literature. However, the observations made in the aforementioned works were mostly made on relatively small-scale models of dimensions comparable to BERT (Devlin et al., 2019) or models from the GPT-2 suite (Radford et al., 2019).

We recall that language models are usually composed of a neural network  $f_\theta$  that takes sequences of tokens  $(\mathbf{w}_{)}) \in [1, V]^{t-1}$  as inputs and produces a relatively low-dimensional contextual representation in  $\mathbb{R}^{d_m}$ , where  $d_m$  is the *hidden dimension* of the model. They then rely on a *language modeling head* that produces logits for contextual token probabilities. A common choice for the language modeling head is a linear layer with parameter  $W \in \mathbb{R}^{V \times d_m}$ , where  $V$  is the vocabulary size. The resulting next-token probability distribution is then given by:

$$p(w_t) = \phi_\theta(\mathbf{w}_{)}) = \sigma(W f_\theta(\mathbf{w}_{$$

where  $\sigma$  is the softmax function.

As discussed in Section 2.3.7, the current trend consists in scaling up the generative pretraining approach introduced with GPT-2, which implies training neural models made of several billions of parameters on gigantic web-mined text corpora (Brown et al., 2020b; Touvron et al., 2023; Almazrouei et al., 2023; Jiang et al., 2023). However, training and serving such highly parameterized models raises energy and hardware-related problematics, which motivates for looking into achieving similar performance levels with smaller models (Sardana and Frankle, 2023).

Nevertheless, the evaluation of the Pythia model suite (Biderman et al., 2023b) has shown that training small models on very large corpora could lead to *saturation*, in the form of a performance degradation in late pretraining. In this chapter, we explore this saturation phenomenon through the lens of representation degeneration, and find that both phenomena strongly correlate. We further demonstrate that representation degeneration strongly occurs in the language modeling head of small models, and we theoretically and empirically show how a linear language modeling head can represent a performance bottleneck for architectures based on small hidden dimensions.

Overall, our contributions can be summarized as:

- We characterize the performance saturation of small language models through evaluation and extrapolation of the scaling laws;
- We find that the representations of smaller models degenerate concurrently with this saturation. We shed light on *rank saturation*, i.e. the explosion of the entropy of singular value distributions of small LM prediction heads;
- We empirically verify that the rank of the target contextual distribution is usually high. Moreover, we observe that regardless of the expressiveness of the output representations of a model, a linear head  $W$  substantially affects performance when  $\text{rank}(W) < 1000$  roughly;
- We theoretically quantify the performance limitation induced by a low-rank linear language modeling head.

## 6.1 LANGUAGE MODEL SATURATION

We first verify that we can indeed observe and quantify performance saturation for the Pythia checkpoints, as they are the only released intermediate checkpoints for a wide range of model sizes. We measure the cross-entropy of Pythia checkpoints on 50k tokens randomly sampled from their pretraining dataset, i.e. The Pile (Gao et al., 2020).

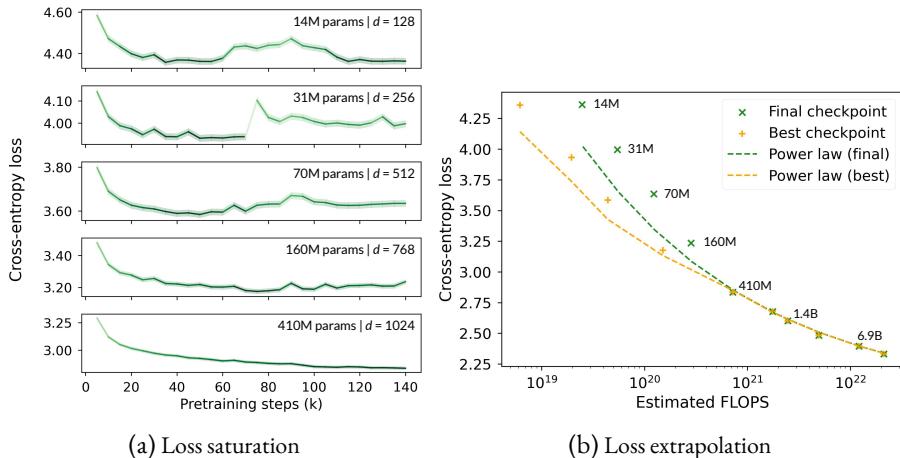


Figure 6.1: Performance of Pythia models on the Pile. On the left, we compare training dynamics of models from 14M (top) to 410M (bottom) parameters, displaying darker shades as we approach the minimal value. On the right, we fit a power law on larger models and find that final checkpoints of smaller models underperform compared to predictions.

In Figure 6.1a, we clearly see that models up to 410M parameters suffer from the saturation phenomenon, characterized as an increase of the in-domain loss in advanced training stages.

In Figure 6.1b, we fit a scaling law in the style of Hoffmann et al. (2022) on data points from models ranging from 410M parameters, only optimizing for model-related constants ( $A$  and  $\alpha$ )

while reusing all other values ( $B = 410.7$ ,  $\beta = 0.28$ ,  $E = 1.69$ ). We recall the relation between parameter count  $N$  and token count  $T$  given in [Hoffmann et al. \(2022\)](#):

$$L(N, T) = \frac{A}{N^\alpha} + \frac{B}{T^\beta} + E$$

We find that optimal parameters are  $A = 119.09$  and  $\alpha = 0.246$ . We display the fitted curves for token counts that correspond to best and final checkpoints. We observe that the final checkpoints underperform the extrapolation by 8% in average. The loss-minimizing (*best*) checkpoints, which are expected to fall short of the extrapolation due to their incomplete learning rate cooldown, only underperform it by roughly 4%.

We report a similar performance saturation on datasets used for evaluation using the LM Evaluation Harness ([Gao et al., 2023](#)), as shown in [Table 6.1](#).

Checkpoint	Lambada (ppl.) ↓	Lambada ↑	StoryCloze ↑	WikiText (ppl.) ↓	SciQ ↑	ARC-e ↑
Best	<b>24.6</b>	<b>40.3</b>	<b>59.6</b>	<b>30.47</b>	<b>79.6</b>	<b>46.5</b>
Final	32.9	38	57.2	33.4	73.4	43.2

Table 6.1: Zero-shot performance of Pythia-160M best and final checkpoints on evaluation datasets. Unless specified, we report accuracy for all tasks.

## 6.2 PERFORMANCE SATURATION IS RANK SATURATION

### 6.2.1 ANISOTROPY AT SCALE

Given that most research on degeneration was conducted on smaller models, it remains unclear whether anisotropy affects models with over 1 billion parameters. In order to address this question, we compute average cosine-similarity of intermediate representations across layers in suites of models; namely GPT-2 ([Radford et al., 2019](#)), OPT ([Zhang et al., 2022](#)), Pythia ([Biderman et al., 2023b](#)), and Gemma ([Team et al., 2024](#)). We use a subsample of The Pile ([Gao et al., 2020](#)), as we hypothesize that the domain of this dataset includes or matches the domain of the pretraining datasets used in these suites.

In [Figure 6.2](#), we observe that most layers of Transformers models are anisotropic to some extent, regardless of the scale. Nevertheless, there seems to be a dichotomy in the last layer, where models are either nearly isotropic or highly anisotropic. Interestingly, we notice that the dichotomy aligns with the one of the saturation phenomenon for the Pythia suite, where only models containing 160M or fewer parameters seem affected by last-layer anisotropy. In this chapter, we will focus on last-layer anisotropy, and we refer to [Chapter 7](#) for a study of degeneration in intermediate layers.

We thus decide to study the training dynamics of anisotropy for the Pythia suite, and compare them with the saturation phenomenon in [Figure 6.3](#).

[Figure 6.3](#) illustrates a neat correlation between the emergence of the performance saturation phenomenon and the appearance of anisotropy in the last-layer representations of the models. It also shows that anisotropy increases abruptly around the saturation point during training.

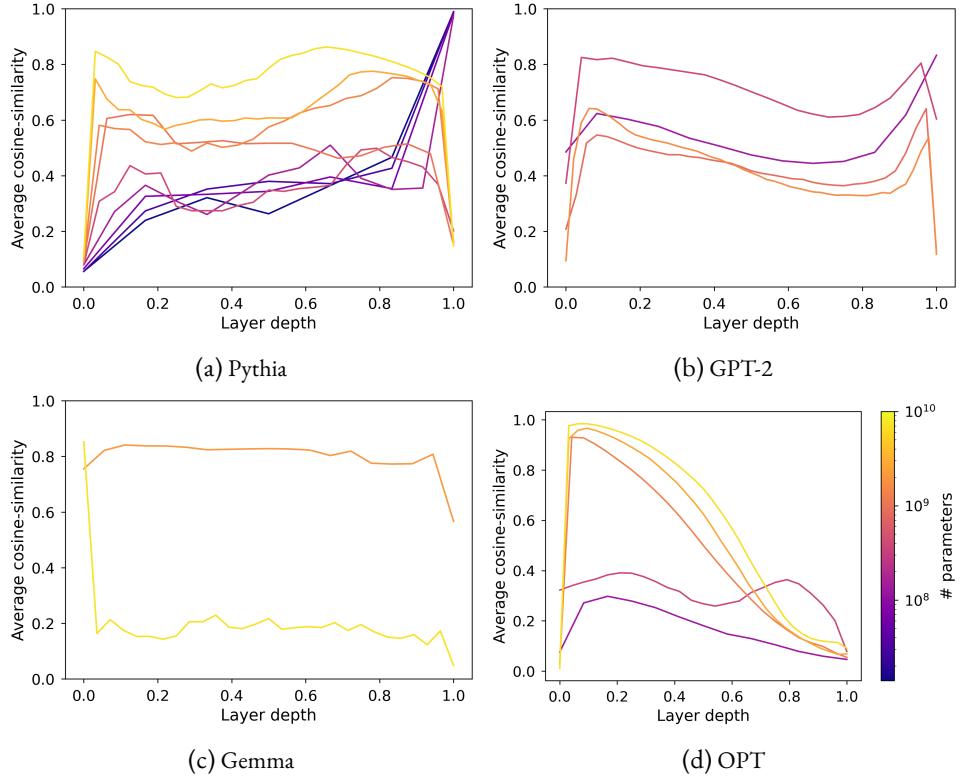


Figure 6.2: Anisotropy in function of layer depth (i.e. order in the forward pass).

Moreover, we see here that on a specific in-domain corpus, the models quickly lose performance at saturation and never seem to fully recover from this explosion.

### 6.2.2 SINGULAR VALUES SATURATION

Average cosine-similarity is a valuable measure of the uniformity of a distribution, but including other metrics can help to better capture the complexity of some manifolds (Rudman et al., 2022). Moreover, it only focuses on the output embeddings of the language models, and not on their weights. In this section, we extend our analysis by studying the singular value distributions of the language modeling heads, to link our empirical observations to our theoretical findings. In Figure 6.4, we display the singular value distributions of the final predictive layer weights  $W$  along training.

Figure 6.4 sheds light on a specific pattern of spectral saturation, roughly co-occurring with the performance saturation phenomenon. It shows that the singular value distribution progressively flattens during training, and nearly reaches uniformity before abruptly evolving towards a spiked distribution with a high maximal singular value, relatively to the other ones.

In order to quantify this behavior more accurately, we use a *singular entropy metric*, computed as the Kullback-Leibler divergence between the normalized singular value distribution and the uniform distribution.

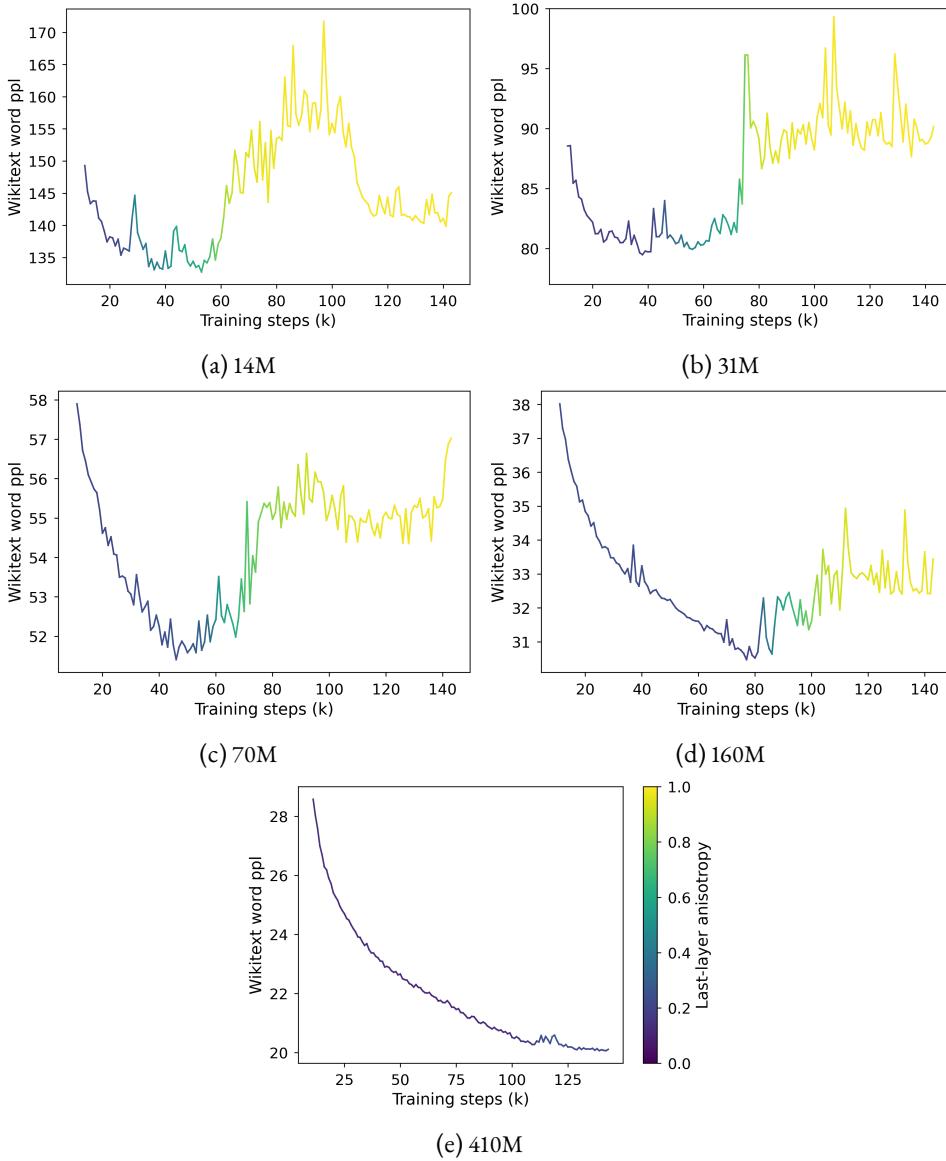


Figure 6.3: Evolution of the language modeling performance on the Wikipedia test set from the LM Evaluation Harness (Gao et al., 2023) and last-layer anisotropy of Pythia models along training (color).

[Figure 6.5](#) shows that singular distributions evolve differently for models using less than 410M parameters than for the larger ones. The heads of small models see their singular value distributions become increasingly uniform, up to a point where they degenerate abruptly, which again correlates with the LM performance drop. The singular value distributions of larger models tend to be more stable, and do not display clear monotonic patterns throughout training.

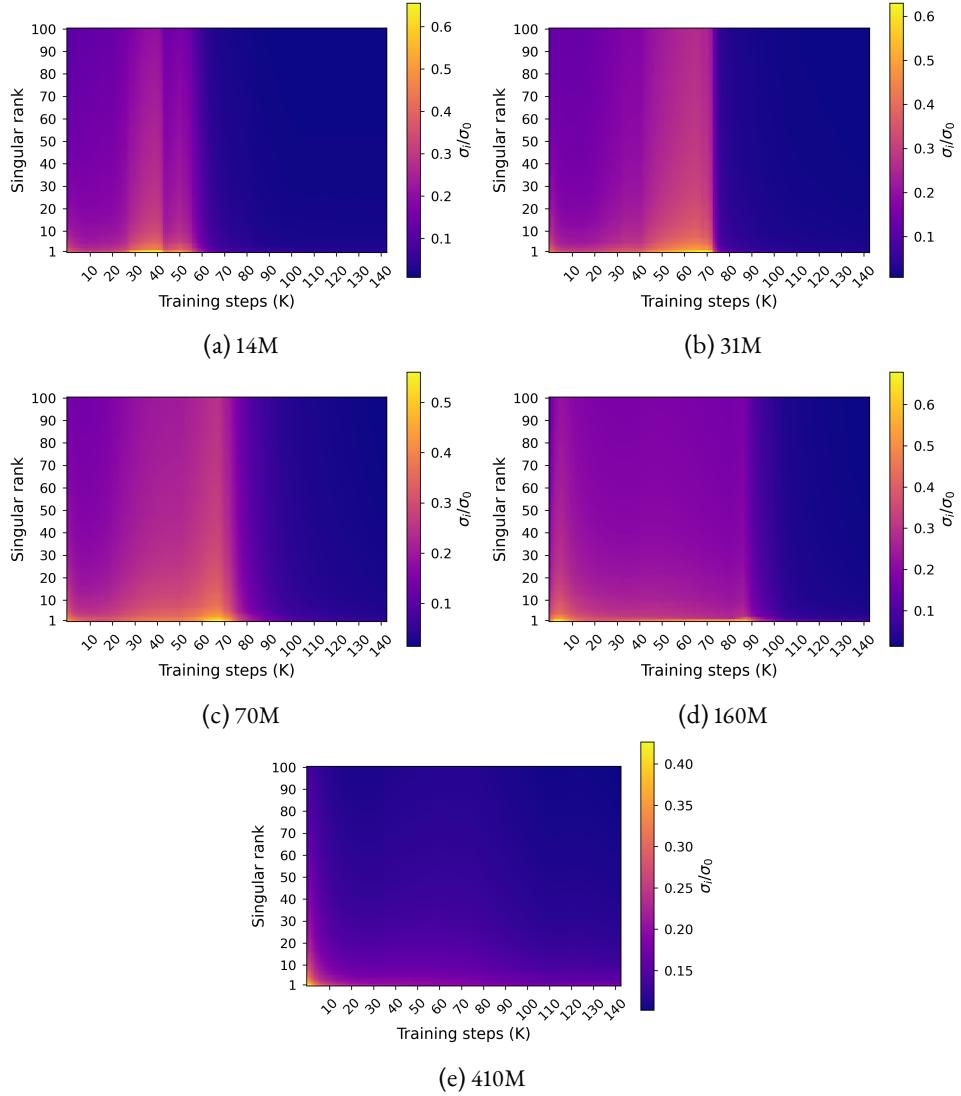


Figure 6.4: Evolution of the singular value distributions of the LM heads of Pythia models during training, normalized by the maximum singular value.

### 6.3 THE SOFTMAX BOTTLENECK & LANGUAGE DIMENSIONALITY

#### 6.3.1 INHERENT DIMENSIONALITY OF NATURAL LANGUAGE

Intuitively, the saturation of the singular values distribution observed only for smaller models in [Section 6.2.2](#) questions the dimensionalities involved in the optimization of the LM head. In this section, we propose to empirically measure a critical value for the rank of the LM head, and to estimate the dimensionality of the contextual probability distribution the head's outputs are supposed to match.

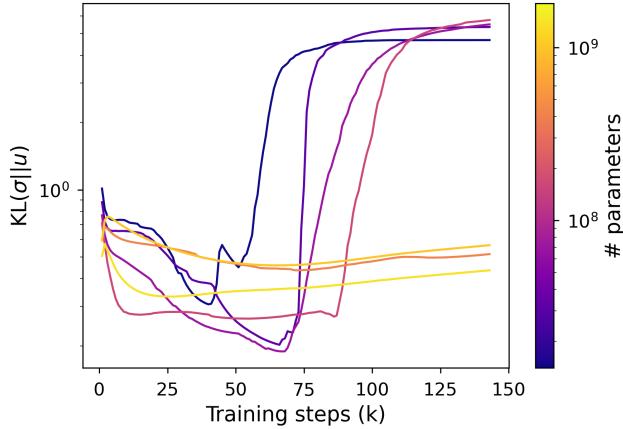


Figure 6.5: Training dynamics of the singular entropy, for different Pythia models.

In order to empirically measure the effect of the rank of the linear head, we propose to train rank-constrained heads on pretrained contextual representations from highly-parameterized language models. In order to control the maximum rank  $r$ , we consider heads of the form  $W = AB \in \mathbb{R}^{V \times d_m}$ , where the coefficients of  $A \in \mathbb{R}^{V \times r}$  and  $B \in \mathbb{R}^{r \times d_m}$  are drawn from  $\mathcal{N}(0, 1)$  ( $d_m$  being the hidden dimension of the model). The rank of such  $W$  matrices is limited by the parameter  $r \in [1, d_m]$ , which we sweep over a wide range of values.

We freeze the language models and train the rank-constrained heads on their output representations on roughly 150M tokens, while adjusting the learning rate to the trainable parameter count.

More precisely, we freeze the pretrained weights in the Transformer layers, and we train each rank-constrained head (i.e. in the form  $W = AB$  with  $r$  as the inner dimension of the matrix product) for various values of  $r$  on 150M tokens sampled from The Pile using 4 V100 GPUs for the Pythia models and 4 A100 GPUs for Llama-7B. We use the hyperparameters from [Biderman et al. \(2023b\)](#), except for the batch size which we set to 256 as it fits our hardware setup better. As the trainable parameter count evolves with  $r$ , we search for the best-performing learning rates among values ranging from  $1 \cdot 10^{-3}$  to  $5 \cdot 10^{-2}$ .

We report the chosen learning rates in [Figure 6.6](#).

In [Figure 6.7](#), we observe that perplexity starts to noticeably decrease when the rank of the language modeling head  $W$  is roughly inferior to 1000, *regardless of the model size*. This hints that the head is not a major performance bottleneck for models with greater hidden dimensions, but that it may hurt performance for models with smaller ones independently of the quality of the output representations.

Another interesting factor to estimate is the dimensionality inherent to the data itself. To avoid possible effects related to specific inductive biases, we train naive 5-gram language models on several datasets of varying coverage (IMDb ([Maas et al., 2011](#)), WikiText ([Merity et al., 2016](#)), and The Pile ([Gao et al., 2020](#))), using two tokenizers of varying vocabulary sizes (30k tokens for Llama-2 and 50k tokens for Pythia). Given  $C$  observed 5-grams, we consider the matrices  $W \in \mathbb{R}^{C \times V}$  where each row is a probability distribution over possible tokens in a given 4-token context, and compute

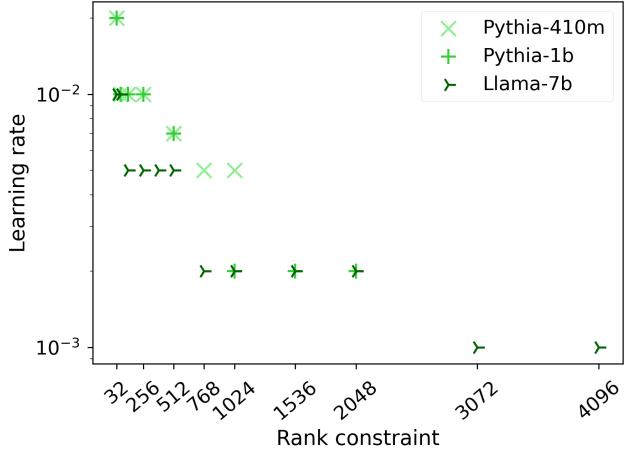


Figure 6.6: Chosen peak learning rates used for the rank-constrained head experiments for each model.

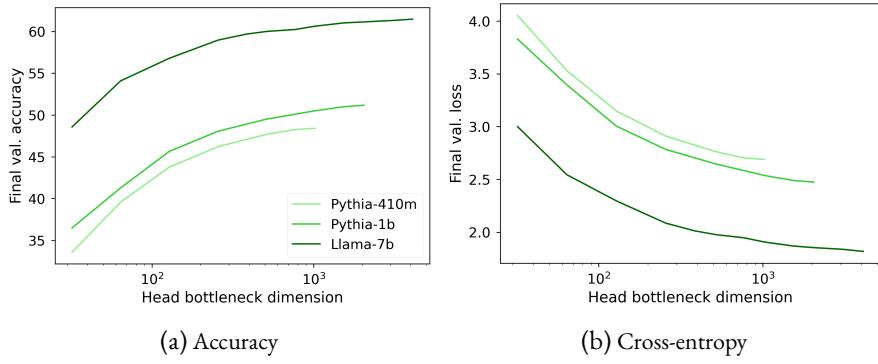


Figure 6.7: Performance of several models as the bottleneck dimension of the head increases.

their singular value distributions, as in [Terashima et al. \(2003\)](#). In [Figure 6.8](#), we report *W-error*, the minimal approximation error on  $W$  for a matrix of rank  $d$  as predicted by the Eckart-Young-Mirsky theorem (see [Lemma 6.3.2](#)), normalized by the Frobenius norm of  $W$ :

$$W\text{-error}(d) = \frac{\|\sigma_{d+1:}\|_2}{\|W\|_F}$$

We find that the estimated rank of  $W$  is non-negligible with respect to the usual magnitude of hidden dimensions. In the next section, we analyze the connection between the dimensionality of an ideal linear language modeling head and performance from a theoretical perspective.

### 6.3.2 A THEORETICAL BOTTLENECK

In this section, we aim at identifying a formal link between the inherent dimensionality of the contextual distribution and the performance bottleneck that can be attributed to the lower di-

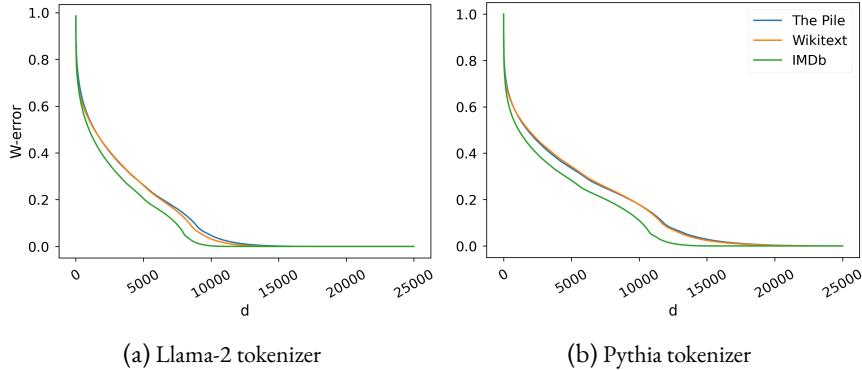


Figure 6.8:  $W$ -error as  $d_m$  increases, for different tokenizers and datasets. We observe that while  $W$ -error can be halved using 1000 or 2000 dimensions, it only becomes negligible after 10,000-15,000 dimensions.

dimensionality of the output representations of a language model. To that end, we conceptualize a language modeling head optimized on *ideal* contextual representations, and we explore the relationship between its spectral properties and the performance gap induced when training a low-rank head on the same representations.

Let's consider a set  $\mathcal{T}$  of sequences  $(\mathbf{w}^i)_{i \in [1, |\mathcal{T}|]}$  of elements taken from a vocabulary of size  $V$ , representing the pretraining data. We consider a function  $\phi^*$  that *perfectly* (e.g. in a bijective way) represents a given context  $\mathbf{w}_{<t}^i$  as a single real vector of *infinite* dimension. As we do not focus on  $\phi^*$ , we can simplify the notations by introducing the contextual representations  $h_{i,t}^* = f^*(\mathbf{w}_{<t}^i)$ .

The task of the linear language modeling head can be formalized as an optimization problem on the matrix  $W$ :

$$W^* = \arg \min_{W \in \mathbb{R}^{V \times \infty}} \sum_{i=1}^{|\mathcal{T}|} \sum_{t=1}^{|\mathbf{w}^i|} \mathcal{L}(W, h_{i,t}^*, w_t^i) \quad (6.1)$$

where  $\mathcal{L}$  is the cross-entropy objective defined using the softmax function  $\sigma$  as:

$$\mathcal{L}(W, h, w) = -\log(\sigma(Wh)_w)$$

In practice, a neural language model  $f_\theta$  produces contextual representations  $h_{i,t} = f_\theta(\mathbf{w}_{<t}^i)$  of dimension  $d_m \in \mathbb{N}^*$ . The linear language modeling head  $W_\theta \in \mathbb{R}^{V \times d_m}$  is trained concurrently with  $f_\theta$  with the same objective as in [Equation 6.1](#).

We focus on the maximal expressiveness of a lower-dimensional head: when provided with *perfect* contextual representations  $h_{i,t}^*$ , what is the maximal performance level of a linear language modeling head of maximal rank  $d$ ? This question can be put in mathematical terms:

$$W_d^* = \arg \min_{W \in \mathbb{R}^{V \times \infty}} \sum_{i=1}^{|\mathcal{T}|} \sum_{t=1}^{|\mathbf{w}^i|} \mathcal{L}(W, h_{i,t}^*, w_t^i) \text{ s.t. } \text{rank}(W) \leq d \quad (6.2)$$

[Lemma 6.3.1](#) shows that by approaching  $W^*$  directly, we can asymptotically expect to close the performance gap.

**Lemma 6.3.1.** Let's consider  $W \in \mathbb{R}^{V \times \infty}$ ,  $M \in \mathcal{H}^{V \times \infty}$  the matrix unit sphere for the Frobenius norm  $\|\cdot\|_F$ , and  $\varepsilon \in \mathbb{R}_+^*$  such that  $W = W^* + \varepsilon M$ . When  $\varepsilon \rightarrow 0$ , for all  $h \in \mathbb{R}^d$  and  $w \in [1, V]$ :

$$|\mathcal{L}(W, h, w) - \mathcal{L}(W^*, h, w)| = O(\varepsilon)$$

*Proof.* The proof is mainly based on calculations and limited development:

$$\begin{aligned} & |\mathcal{L}(W, h, w) - \mathcal{L}(W^*, h, w)| \\ &= \left| -\log \frac{\exp((Wh)_w)}{\sum_{j \in V} \exp((Wh)_w)} + \log \frac{\exp((W^*h)_w)}{\sum_{j \in V} \exp((W^*h)_w)} \right| \\ &= \left| -(\varepsilon Mh)_w + \log \frac{\sum_{j \in V} \exp((W^*h)_j) \exp((\varepsilon Mh)_j)}{\sum_{j \in V} \exp((W^*h)_j)} \right| \\ &= \left| -\varepsilon(Mh)_w + \log \left( 1 + \frac{\sum_{j \in V} \varepsilon \exp((Mh)_j)}{\sum_{j \in V} \exp((W^*h)_j)} + o(\varepsilon) \right) \right| \\ &= \left| -\varepsilon(Mh)_w + \varepsilon \frac{\sum_{j \in V} \exp((Mh)_j)}{\sum_{j \in V} \exp((W^*h)_j)} \right| + o(\varepsilon) \\ &= \varepsilon \left| -(Mh)_w + \frac{\sum_{j \in V} \exp((Mh)_j)}{\sum_{j \in V} \exp((W^*h)_j)} \right| + o(\varepsilon) \end{aligned}$$

The continuous function  $M \rightarrow \left| -(Mh)_w + \frac{\sum_{j \in V} \exp((Mh)_j)}{\sum_{j \in V} \exp((W^*h)_j)} \right|$  is bounded on the compact matrix unit sphere (i.e. where  $\|M\|_F = 1$ ), which ends the proof.

**Remark :** This result could also be proven using a differentiability argument, but we prefer to display a more precise relation between the loss gap and the error on the  $W$  matrix approximation, stressing out its quasi-linear nature. This formulation will hopefully pave the way for further exploration of this relation in future works.

□

Hence, our problem is linked to a low-rank matrix approximation (Kumar and Schneider, 2017), which has direct connections with spectral theory. In our case, we can use the Eckart–Young–Mirsky theorem.

**Lemma 6.3.2.** (Eckart–Young–Mirsky theorem) Let's consider  $(\sigma_i)$  the singular values of  $W^*$  in decreasing order, and  $\mathcal{M}_d$  the set of matrices in  $\mathbb{R}^{V \times \infty}$  of rank  $d < V = \text{rank}(W^*)$ . Then:

$$\min_{W_d \in \mathcal{M}_d} \|W_d - W^*\|_F = \sqrt{\sum_{i=d+1}^V \sigma_i^2}$$

Combining all of the above yields [Theorem 6.3.3](#).

**Theorem 6.3.3.** Let's consider  $(\sigma_i)$  the singular values of  $W^*$  in decreasing order. Then, when  $d \rightarrow V$ , the loss gap induced by a  $d$ -dimensional bottleneck on the linear LM head follows:

$$\sum_{i=1}^{|\mathcal{T}|} \sum_{t=1}^{|\mathbf{w}^i|} \mathcal{L}(W_d^*, h_{i,t}^*, w_t^i) - \mathcal{L}(W^*, h_{i,t}^*, w_t^i) = O\left(\sqrt{\sum_{i=d+1}^V \sigma_i^2}\right)$$

*Proof.* Let us note  $W_d$  the best approximation of  $W^*$  of rank  $d$  with respect to the Frobenius norm. By the triangle inequality, we have that:

$$\left| \sum_{i=1}^{|\mathcal{T}|} \sum_{t=1}^{|\mathbf{w}^i|} \mathcal{L}(W_d^*, h_{i,t}^*, w_t^i) - \mathcal{L}(W^*, h_{i,t}^*, w_t^i) \right| \leq \sum_{i=1}^{|\mathcal{T}|} \sum_{t=1}^{|\mathbf{w}^i|} \left| \mathcal{L}(W_d, h_{i,t}^*, w_t^i) - \mathcal{L}(W^*, h_{i,t}^*, w_t^i) \right| \quad (6.3)$$

The Eckart-Young-Mirsky theorem tells us that when  $d \rightarrow V$ ,

$$\|W_d - W^*\|_F = \sqrt{\sum_{i=d+1}^V \sigma_i^2} \rightarrow 0$$

By defining  $\varepsilon = W_d - W^*$ , we can apply Lemma 6.3.1 and show that:

$$|\mathcal{L}(W_d, h_{i,t}^*, w_t^i) - \mathcal{L}(W^*, h_{i,t}^*, w_t^i)| = O(\|W_d - W^*\|_F) = O\left(\sqrt{\sum_{i=d+1}^V \sigma_i^2}\right)$$

From Equation (6.3), we have that:

$$\left| \sum_{i=1}^{|\mathcal{T}|} \sum_{t=1}^{|\mathbf{w}^i|} \mathcal{L}(W_d^*, h_{i,t}^*, w_t^i) - \mathcal{L}(W^*, h_{i,t}^*, w_t^i) \right| = O\left(\sqrt{\sum_{i=d+1}^V \sigma_i^2}\right)$$

By definition of  $W^*$  and  $W_d^*$ , we also have that:

$$0 \leq \sum_{i=1}^{|\mathcal{T}|} \sum_{t=1}^{|\mathbf{w}^i|} \mathcal{L}(W_d^*, h_{i,t}^*, w_t^i) - \mathcal{L}(W^*, h_{i,t}^*, w_t^i)$$

which ends the proof.  $\square$

**REMARK** The bound used in Equation (6.3) can be rather loose in practice. We can think of no particular reason why approaching  $W^*$  directly should be the optimal way to minimize the loss on  $\mathcal{T}$ . Hence, the presented result should be taken carefully, and we leave the refinement of such an analysis for future work.

$\square$

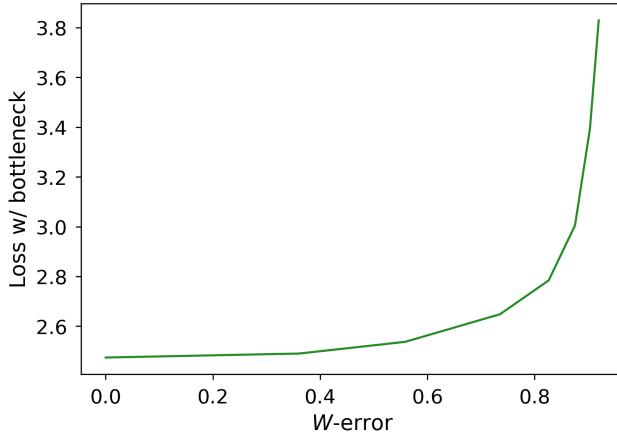


Figure 6.9: Final loss with trained rank-constrained heads (mimicking  $W_d^*$ ), as a function of the theoretical  $W$ -error for rank  $d$  on the head of the Pythia-1B model.

These properties shed light on how the dimensionality of the ideal language modeling head impacts the performance when the LM head is low-rank. However, the relation obtained in [Theorem 6.3.3](#) is not particularly strong, as discussed in the proof.

In [Figure 6.9](#), we compare the results of the head bottleneck experiment of the Pythia-1B model in [Section 6.3.1](#) to the  $W$ -error on the head of the same model as the bottleneck dimension  $d$  evolves. It shows that the loss gap grows slowly with the  $W$ -error, implying that even when the allowed rank would lead to a poor approximation of  $W$ , the performance can still remain acceptable. We notice that the performance starts decreasing when the  $W$ -error outgrows 0.6.

## 6.4 DISCUSSION

One way to address the problem at hand could be to train shallow small language models, increasing hidden dimension at the expense of other hyperparameters, such as layer count or feed-forward dimension. However, we believe that such research directions may not be promising in this context. Previous works have extensively explored and optimized the hyperparameter choices for various architecture sizes. The impact of width and depth has been extensively studied ([Merrill et al., 2022](#); [Tay et al., 2022](#); [Petty et al., 2023](#)), often showcasing the importance of depth in final performance and generalization capabilities.

Another possible way forward consists in implementing more expressive softmax alternatives ([Yang et al., 2018](#); [Chang and McCallum, 2022](#)) in the context of pretraining small language models on large datasets. We leave the exploration of such techniques for future work.

We also believe that further exploration of the specific nature of the singular components after the collapse we describe in [Section 6.2.2](#) could improve our understanding of LM saturation. We hypothesize that the resulting dominating components are correlated with token frequency, based on previous works that link anisotropy with token frequency ([Gao et al., 2019b](#); [Ethayarajh, 2019](#); [Bié et al., 2021](#)) and show the importance of token frequency in the LM head mechanism ([Meister et al., 2023](#)).

We argue that our work demonstrates that last-layer anisotropy is symptomatic of performance saturation, and is thus likely not a desirable property of language models. We also advocate that this work paves the way towards a better understanding of the structure of the contextual probability distribution, which could also enhance our interpretation of the scaling laws.

## CONCLUSION

Small language models can be affected by performance saturation during training. We find that this phenomenon can be explained by an inherent difficulty in mapping a low-dimensional output representation space to a high-rank contextual probability distribution through a linear language modeling head. Indeed, we show a theoretical link between the performance gap induced by a smaller hidden dimension and the spectral properties of the contextual probability distribution.

We empirically confirm that the rank of such a mapping can be expected to be relatively high compared to regular hidden dimension choices. Moreover, we conduct experiments to measure the impact of constraining the rank of the LM head on the performance of a large language model. Our results show that performance noticeably drops when using a hidden dimension smaller than roughly 1000. We further analyze the saturation phenomenon through the lens of spectral analysis and find that the emergence of last-layer anisotropy that only affects small models can be correlated with saturation. We also show that the LM heads of small models concurrently suffer from *spectral* saturation, i.e. a uniformization of singular values that leads to a degenerated state.

Our work paves the way for a better understanding of the consequences of the softmax bottleneck on language modeling, and for the conception of language models that better embrace the complexity of the target probability distribution.

## LIMITATIONS

The main limitation of this chapter is the relatively small amount of saturated language models we studied. As it is the only suite of language models trained in the range of interest to release an extensive amount of intermediate checkpoints, we could only observe the training dynamics of small Pythia models. Although we observe strong last-layer anisotropy for the smallest GPT-2 model, we cannot tell with certainty whether it suffered from saturation. The OPT-125m model does not display a strong last-layer anisotropy, which could indicate that it was not affected by the saturation phenomenon.

Nevertheless, we argue that we do not show that *all* small models should suffer from saturation, but rather that the saturation of small language models is symptomatic of a limitation that may affect language models that are based on a relatively small hidden dimension. Furthermore, we do not state that there is a causality relationship between degeneration and low hidden dimension choices, but rather expose a strong correlation between both phenomenon that can be explained through the prism of our softmax bottleneck analysis.

Another limitation of this work is the loose nature of the mathematical connection that we establish between the dimensionality of the ideal language modeling head and the rank-constrained performance (cf. [Theorem 6.3.3](#)). Moreover, it can also be argued that considering *ideal*  $h_{i,t}^*$  representations is an ill-defined notion. We argue that the reasoning behind [Theorem 6.3.3](#) could be

applied to any contextual representations, as the *ideal* nature of  $h_{i,t}^*$  is not necessary in the demonstrations. The word *ideal* reflects that our observations hold for  $h_{i,t}^*$  representations obtained from *any underlying model*, to an extent that depends on the structure that these representations impose on the  $W^*$  matrix for a given training set  $\mathcal{T}$ .

This chapter shows that language model representations can suffer not only from biases carried over by training data ([Chapter 5](#)), but also from limitations inherited from the complexity of language itself. Representing token contextual distributions using low-dimensional dense vectors inevitably restricts the performance of language models, and the magnitude of dimensionalities that are significantly affected by this phenomenon is empirically not negligible.

This effect is strongly correlated with last-layer anisotropy, but it is unclear whether this effect is sufficient to account for anisotropy in the other layers of language models. We explore the degeneration phenomenon in intermediate layers in [Chapter 7](#).

## 7

## ANISOTROPY IS INHERENT TO SELF-ATTENTION IN TRANSFORMERS

Anisotropy has been widely observed among self-supervised models based on Transformers, and literature suggests that it may be a consequence of optimizing the cross-entropy loss on long-tailed distributions of tokens, as discussed in [Chapter 6](#). However, this observation does not suffice to explain the anisotropy levels of other layers in pretrained language models, including those that seem to be affected less by these frequency-based distortions. For instance, the OPT models ([Zhang et al., 2022](#)) have isotropic last layers, but degenerated first layers (see [Figure 6.2d](#) in [Chapter 6](#)).

This raises the question of the effect of anisotropy on the inner workings of Transformer layers, but also of the effect of the inner workings of Transformers layers on the geometry of their output distributions.

In this paper, we investigate the anisotropy problem in depth, and we make several contributions:

- We demonstrate empirically that anisotropy can be observed in language models with character-aware architectures that should not suffer directly from the same consequences as token-based models. We extend our observations to Transformers trained on other modalities, such as image and audio data, and show that anisotropy cannot be explained solely based on linguistic properties;
- We provide empirical observations on the anisotropic properties of the Transformer block by studying untrained layers, and establish a relation between anisotropy and the general sharpness of the self-attention mechanism;
- We conduct an analysis of the representations used in self-attention (queries and keys) along training and show that anisotropy appears intrinsically in the self-attention mechanism, when training pushes for sharp patterns.

As mentioned in [Section 4.3.2](#), several works have established a connection between word frequency and distortions of the latent spaces ([Yu et al., 2022](#); [Puccetti et al., 2022](#); [Rajaee and Pilehvar, 2022](#)). [Biš et al. \(2021\)](#) have shown that anisotropy in LMs could be explained by a global *drift* of the representations in the same direction, thus unifying conclusions from [Ethayarajh \(2019\)](#) and [Gao et al. \(2019b\)](#). The authors propose that this drift is caused by the persistent updating of the representation of rare and unused tokens in a consistent direction, due to the nature of the softmax operation in the cross-entropy loss. They show that removing the average component to all representations leads to a nearly perfect isotropy.

Various methods have been proposed to reduce anisotropy in Transformer-based LMs at token-level ([Rajaee and Pilehvar, 2021](#); [Wang et al., 2020a](#)), or at sentence-level ([Gao et al., 2021](#); [Yan et al., 2021](#); [Su et al., 2021](#)) (see [Section 3.3](#)). They usually consist in post-processing the representations,

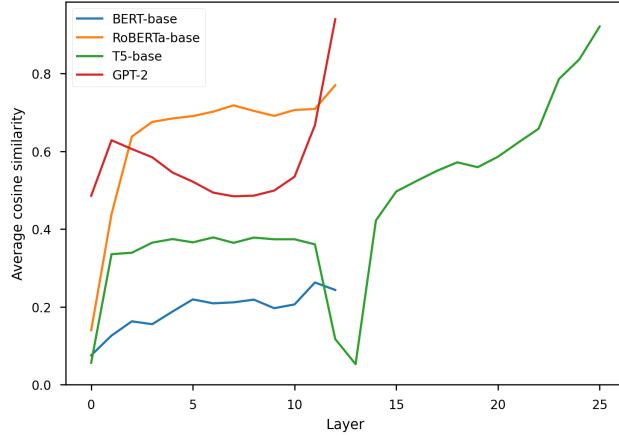


Figure 7.1: Average cosine-similarity between hidden representations across layers for token-level NLP models. For T5-base, we concatenate encoder and decoder results.

and lead to downstream performance boosts. We argue that these positive results are paving the way for the search of pre-training objectives that do not introduce anisotropy in the first place, in the hope that the resulting models will also perform better without any post-processing, and potentially be trained more efficiently. This motivates us to gain a deeper understanding of the underlying factors that induce anisotropy, whether they belong in data, architectures, or training procedures.

## 7.1 ANISOTROPY IN PRE-TRAINED TRANSFORMERS

### 7.1.1 CHARACTER-BASED NLP

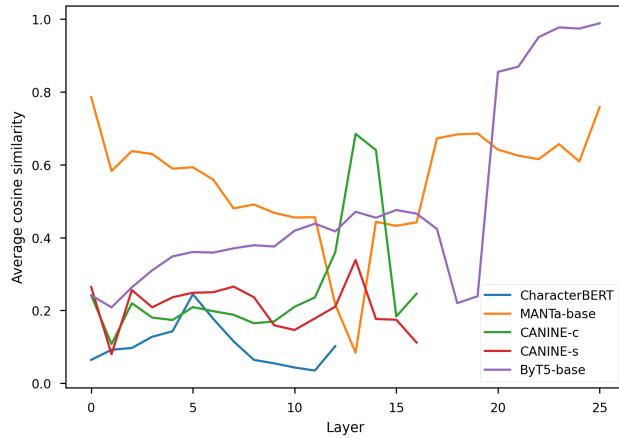


Figure 7.2: Average cosine-similarity between hidden representations across layers for character-level models.

To assert whether the cross-entropy objective applied on vocabularies containing rare tokens is the sole cause for the common drift issue, we explore anisotropy in character-based models. We study different architectures presented in [Section 2.4.4](#), and our character-level model:

- CharacterBERT ([El Boukkouri et al., 2020](#)) is constructing whole word representations from character embeddings put through convolutions and highway layers, before feeding them to a Transformers architecture.
- CANINE ([Clark et al., 2022b](#)) is downsampling contextualized character representations via a strided convolution before feeding them to a Transformers. It can be trained either with a subword-based objective (CANINE-s) or with a character-level one (CANINE-c).
- MANTa-LM (see [Chapter 9](#)) is based on a differentiable segmentation and embedding module added before an encoder-decoder model in the style of T5 ([Raffel et al., 2020a](#)). It takes bytes as inputs and outputs, but builds internal representations that are usually based on several bytes.
- ByT5 ([Xue et al., 2022a](#)) is a version of T5 that is trained at byte-level. To afford for more complex encoding, the authors resize the encoder-decoder architecture.

Neither of these architectures should suffer from out-of-vocabulary tokens in the process of creating representations. The models that predict at word or sub-word level (CharacterBERT and CANINE-s) could have the cross-entropy loss systematically pushing away rare item representations. However, it is rather unclear why this would imply an embedding drift for deeper layers. Hence, if anisotropy was only caused by the presence of unused or rare subwords, those character-level models should be much less prone to this issue.

To verify this hypothesis, we compute hidden representations for the validation set of the WikiText-103 corpus ([Merity et al., 2017](#)). We then compute the average cosine-similarity between two representations, uniformly taken in the whole validation corpus.

In fact, as shown in [Figure 7.2](#), those models all display significant levels of anisotropy in at least one of their layers. Interestingly, the models that are based solely on characters or bytes for input and prediction (ByT5, CANINE-c, and MANTA-LM) seem to display even higher levels of anisotropy. We note, as it is the case for the T5 model, that the ByT5 decoder displays extremely high levels of anisotropy.

### 7.1.2 OTHER MODALITIES

We have shown in the previous section that character-level language models suffer from anisotropy similarly to token-level ones, hinting that subword token distributions are not solely responsible for anisotropy. Still, it may be argued that anisotropy is related to linguistic properties inherent to textual data. Thus, we proceed to explore the anisotropy problem for Transformers-based models in other modalities, specifically speech and vision.

For speech models, we consider wav2Vec 2.0 ([Baevski et al., 2020a](#)), HuBERT ([Hsu et al., 2021](#)), and Whisper ([Radford et al., 2023](#)) with the Common Voice 11.0 dataset ([Ardila et al., 2020](#)). For vision models, we use ViT ([Wu et al., 2020](#)), BEiT ([Bao et al., 2022](#)), MiT ([Xie et al., 2021](#)), and DEiT ([Touvron et al., 2021](#)) on the ImageNet dataset ([Russakovsky et al., 2015](#)).

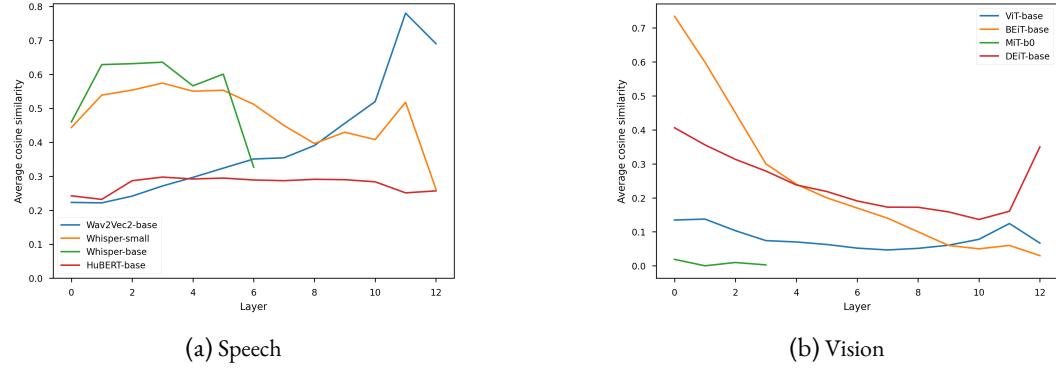


Figure 7.3: Average cosine-similarity between hidden representations across layers for Speech and Vision modalities. We observe that across both modalities, several models display significant levels of anisotropy.

As in [Section 7.1.1](#), we infer hidden representations on the validation sets for each modality. We then uniformly sample pairs of vectors to get cosine-similarity values for every layer of every model. The averaged results are displayed in [Figure 7.3](#).

Once again, almost every model shows a significant level of anisotropy on some of its layers. Notably, speech models seem to have very anisotropic representations, as every layer of every model outputs an average cosine-similarity of at least 0.2. We find some exceptions among vision models, since the MiT model seems to use isotropic representation spaces and the ViT model has a low average cosine-similarity for all its layers.

We also conduct the same experiment for convolution-based networks in the vision modality. The models at glance are ResNet ([He et al., 2016b](#)), EfficientNet ([Tan and Le, 2019](#)), CvT ([Wu et al., 2021](#)), ConvNeXt ([Liu et al., 2022](#)), and VAN ([Guo et al., 2023](#)). For these networks, we flatten convolution maps to vectors before computing the cosine-similarity.

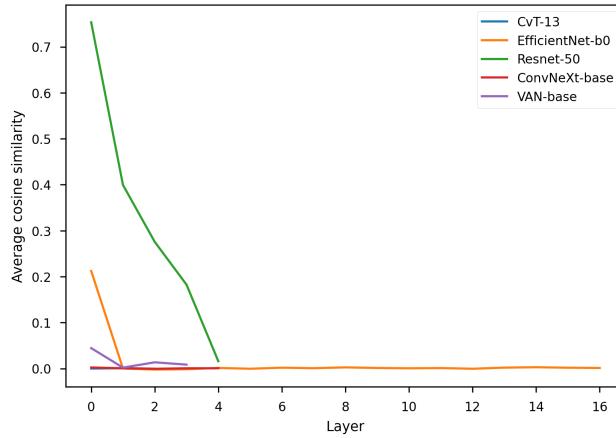
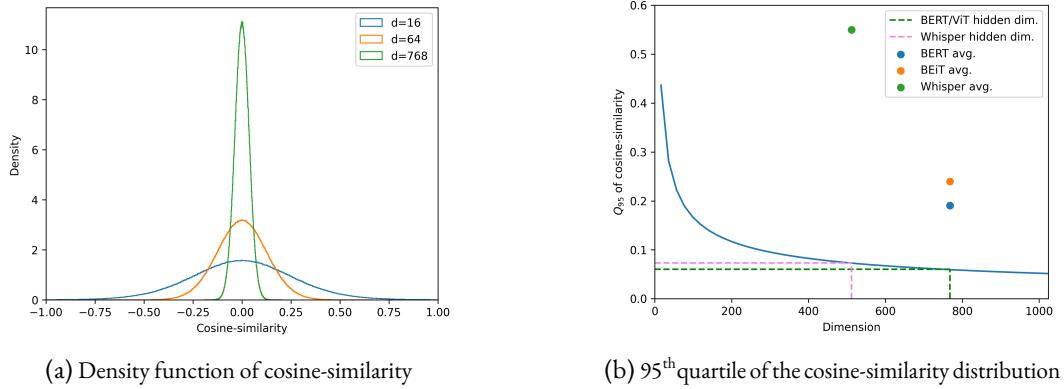


Figure 7.4: Average cosine-similarity between hidden representations across layers for convolution-based vision models.

We observe in [Figure 7.4](#) that most of the convolution-based models are isotropic. Interestingly, the only exception is ResNet-50, whose representations become more and more isotropic as one explores deeper layers. This could partially be explained by the fact that the batch normalization ([Ioffe and Szegedy, 2015](#)) used in some of these models mitigates *a posteriori* the drift effect by removing the mean component of the representations. However, the ConvNeXt model also seems to use isotropic representations while not using batch normalization, which shows that this is not the only factor in the isotropic behavior of these models.

### 7.1.3 WHEN IS ANISOTROPY “HIGH”?



[Figure 7.5](#): Anisotropy metrics on multi-dimensional normal distributions as the dimension increases. In [Figure 7.5b](#), we add points for the average cosine-similarity level of Transformers models for several modalities.

It can be argued that describing anisotropy as the observation of “high” cosine-similarity values is not a convincing definition. This section aims at showing which ranges of cosine-similarity values are characteristic of anisotropic distributions. In [Figure 7.5a](#), we show the density function of the cosine-similarity values obtained when drawing pairs of samples from isotropic normal distributions in  $\mathbb{R}^d$  as  $d$  increases.

For smaller dimensions ( $d = 16$ ), we see that the range of cosine-similarity values that are reached between isotropic distributions is relatively broad compared to the possible spectrum ( $[-1, 1]$ ). As  $d$  increases, the support of the observed distributions seems to become smaller, due to the curse of dimensionality.

We analyze this effect more in-depth in [Figure 7.5b](#), where we plot the 95th quantile of the cosine-similarity distribution in the isotropic scenario. We also add values for the layer-wise average cosine-similarity levels of typical models in several modalities for comparison. We can clearly observe that the levels of cosine-similarity observed in the representations of Transformers-based models are significantly unlikely to be observed in between samples drawn in isotropic normal distributions.

Nevertheless, as we go towards higher dimensional spaces for bigger models (e.g. Llama-65B from [Touvron et al. \(2023\)](#) has 8192 hidden dimensions), we believe that it may be relevant to

introduce isotropy metrics that are grounded to isotropic cosine-similarity distributions. We leave this question for future works.

#### 7.1.4 To DRIFT OR NOT TO DRIFT?

Related works (Bić et al., 2021; Gao et al., 2019b) show that anisotropy in subword-level language models is caused by a drift of the hidden representations in a shared direction. In this section, we try to extend this observation to other modalities.

We study the correlation between the uniformly measured cosine-similarity, and the norm of the average hidden representation  $\|\bar{h}\|_2$  for each layer. If anisotropy could be directly explained by the drift effect, we would expect a monotonic relation between  $\|\bar{h}\|_2$  and the average cosine-similarity. To verify this, we apply a Spearman correlation test on these two metrics for every model from Section 7.1.1 and Section 7.1.2, along with some token-level language models, namely T5 (Raffel et al., 2020a), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and GPT-2 (Radford et al., 2019).

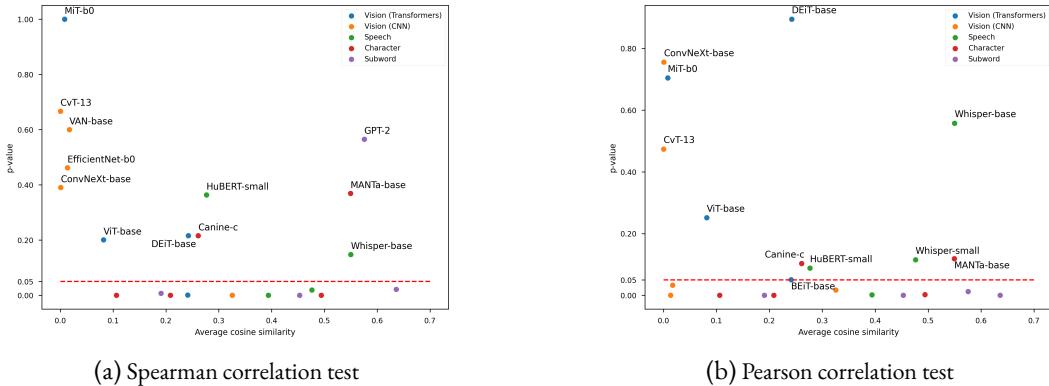


Figure 7.6: p-value of correlation tests between the norm of the average representation and the cosine-similarity averaged over all layers, across modalities. For models above the red dotted line, there is no significant ( $p > 0.05$ ) correlation between the drift effect and the anisotropy level.

In Figure 7.6, we observe that we can correlate the anisotropy level and the magnitude of the drift component across layers for several models. The anisotropy of subword-based models can generally be correlated with the drift effect using the Spearman correlation test, except for GPT-2 for which it may not be appropriate.

The Pearson test measures a linear correlation between random variables, while the Spearman test measures a monotonic correlation. As there is no specific argument in favor of a linear relationship between the measured distributions (average cosine-similarity and norm of the average representation), we decide to favour the Spearman correlation test in order to take into account more complex relation patterns.

Nevertheless, this metric is based on the rank of each observation, and is thus not robust to fluctuations due to sample variance, specifically when working with such small samples. This is reflected by the discrepancy between Pearson and Spearman p-values for some models (e.g. GPT-2). Hence, we report both metrics for completeness.

Interestingly, we notice that the anisotropy affecting most CNN-based vision models is generally not correlated with the drift effect, contrary to Transformers-based models in the same modality. Some speech models (HuBERT and Whisper-base) also display signs of anisotropy that cannot be correlated with the drift effect. Figure 7.6 also shows a correlation for all character-based models but Canine-C and MANTa-base.

## 7.2 EXPLORING THE REPRESENTATION DRIFT

In this section, we focus on some intrinsic properties of the Transformer block in a modality-agnostic fashion, i.e. with minimal assumptions on the data distribution, and without training. We analyze experimentally the behavior of the untrained Transformer block  $T$  when a common bias term  $b$  is added to untrained input representations  $\mathbf{h}$ . This allows us to mimic the common drift as mentioned in Bić et al. (2021) and to identify some properties induced by this artificial drift on the output representations.

### 7.2.1 EXPERIMENTAL SETUP

We consider an embedding lookup table  $E$  and a Transformer block  $T$  with weights initialized as in BERT (Devlin et al., 2019). We then draw 16 input embedding sequences  $\mathbf{h}$  of length 512 uniformly from  $E$ . To account for a drift component of norm  $\eta \in \mathbb{R}$ , we generate a vector  $b_u \sim \mathcal{N}(0, I_d)$ , which we normalize into  $b_\eta = \frac{b_u}{\|b_u\|_2} \cdot \eta$ . We finally compute  $T(\mathbf{h} + b)$  for every sequence  $\mathbf{h}$ , and study the resulting distributions.

Specifically, we study the average norm of the input representations  $\mathbb{E}(\|\mathbf{h} + b_\eta\|_2)$  against the average norm of the output representations  $\mathbb{E}(\|T(\mathbf{h} + b_\eta)\|_2)$  in Figure 7.7b. We also retrieve the self-attention scores before the softmax operation, namely  $\frac{Q^h K^{h^T}}{\sqrt{d_k}}$ , along with the corresponding  $Q^h$  and  $K^h$  matrices. We study some of their properties in Figure 7.8 and Figure 7.9.

### 7.2.2 INPUT VS. OUTPUT ANALYSIS

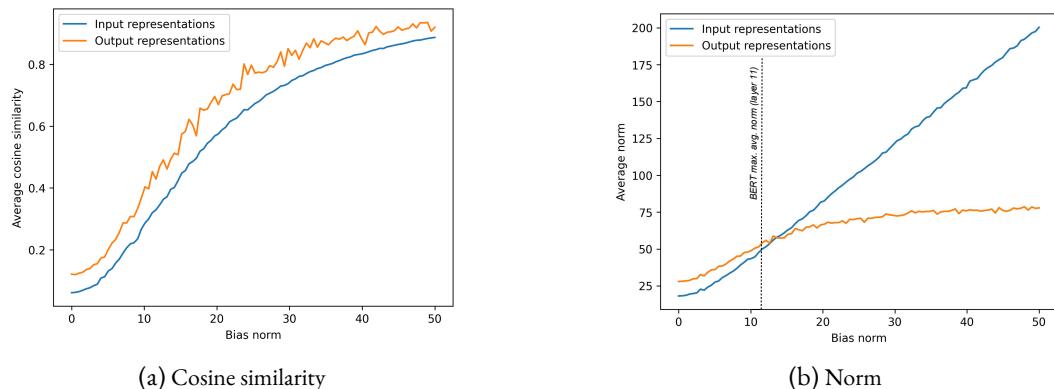


Figure 7.7: Input/Output comparison of a Transformer block from BERT-base as the bias norms increases.

In Figure 7.7a, we observe that the output representations have an average cosine-similarity value that is slightly higher than the one of the input representations, no matter the level of input bias. We also notice that while the norm of the average output representation increases with the bias norm, it seems to meet the corresponding input measure for a given bias norm.

Interestingly, this shows that there is a *fixed point* in terms of norm in the Transformers function with biased input. More formally, there seems to exist a bias norm  $\eta^* \in \mathbb{R}_+$  such that:

$$\mathbb{E}_{\mathbf{h}, b_{\eta^*}} (\|\mathbf{h} + b_{\eta^*}\|) = \mathbb{E}_{\mathbf{h}, b_{\eta^*}} (\|T(\mathbf{h} + b_{\eta^*})\|)$$

Moreover, this fixed point level  $\eta^*$  is in the order of magnitude of the average hidden state norms of the layers of the trained BERT model. This hints that the model’s representations stabilize when their norm is close to this fixed point. We leave a more thorough analysis of this hypothesis for future work.

### 7.2.3 EXPLORING THE TRANSFORMER BLOCK

To understand the effect of the drift effect on the inner workings of the Transformer layer, we take a closer look at the self-attention operation as the average input representation drifts away.

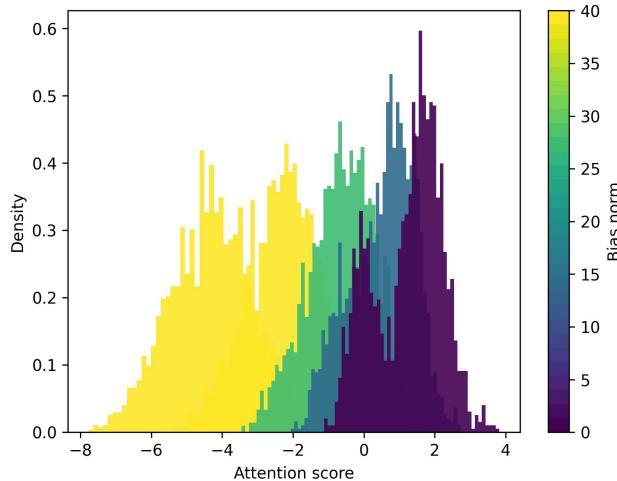
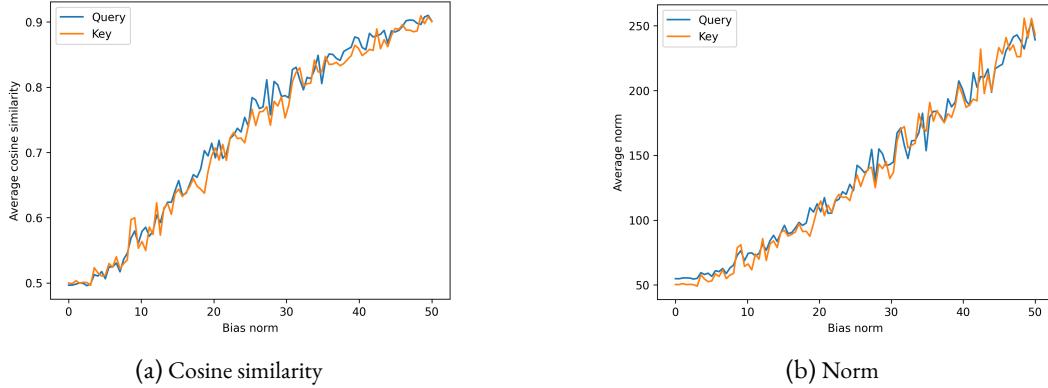


Figure 7.8: Histograms of the pre-softmax attention scores as the input bias norm increases. Other initializations of the layer and of the bias direction  $b_u$  led to a general *increase* of the attention scores instead.

Figure 7.8 shows that the attention scores tend to move away from zero as the input bias norm increases. Indeed, as the norm of the average  $\bar{\mathbf{h}}$  of the input embeddings increases, we can expect the query and key vectors  $Q^h$  and  $K^h$  to also display signs of anisotropy. Actually, for each self-attention head, and for all position  $i \in [1, L]$ , we have:

$$\begin{cases} \mathbb{E}_{\mathbf{h}}(Q_i^h) = W_{Q^h} \bar{\mathbf{h}} + b_{Q^h} \\ \mathbb{E}_{\mathbf{h}}(K_i^h) = W_{K^h} \bar{\mathbf{h}} + b_{K^h} \end{cases} \quad (7.1)$$

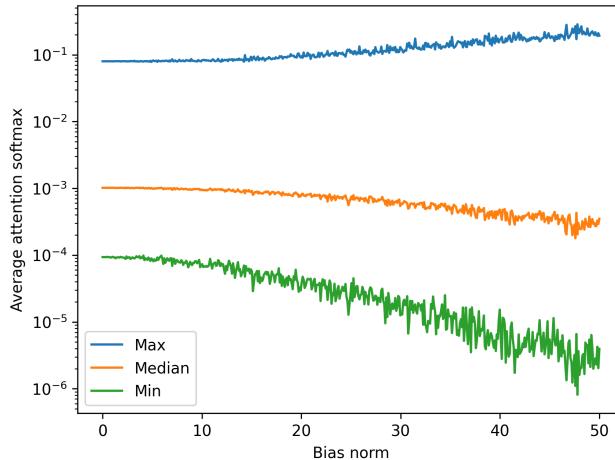
We can observe in [Figure 7.9](#) that query and key representations indeed increase in norm with the input bias norm. We also notice that the corresponding distributions are anisotropic even when no bias is added, which may be a consequence of BERT’s initialization parameters.



[Figure 7.9](#): Analysis of the self-attention query and key distributions

#### 7.2.4 IMPACT OF THE DRIFT

After exploring the consequences of the drift of input representations on the query-key product in self-attention, we identify in this section the implications of this drift at a more explainable level, by observing the resulting post-softmax distributions.



[Figure 7.10](#): Evolution of the self-attention softmax values as the input bias norm increases.

In [Figure 7.10](#), we retrieve softmax values in the self-attention block and for each position, we extract the maximum, the median and the minimum. We then average these values over the whole batch, and repeat for various input bias norm levels. We notice that as the input bias norm increases, the self-attention softmax distributions tend to become less entropic, evolving towards higher

maximal probabilities and lower minimal probabilities. In the following analysis, we'll use the term *sharpness* to discuss entropy levels of the self-attention distributions.

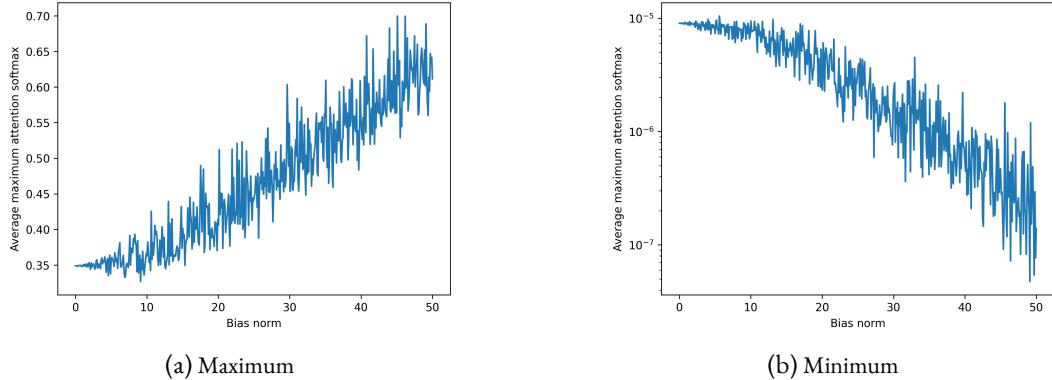


Figure 7.11: Comparison of the extreme values of each sequence averaged over the batch as the bias norm increases.

This sharpening effect of the attention distributions becomes even clearer if we consider the maximum and minimum values over the whole sequences, as in Figure 7.11.

However, at low anisotropy levels, i.e. when the bias norm is low, we see that the effect is not very important. Figure 7.10 and Figure 7.11 only hint at the fact that the drift of embeddings may help the self-attention to be sharper. Another explanation could be that training favors sharp self-attention patterns, as has been pointed out in previous works (Clark et al., 2019b), which in turn induces a drift in the models' representations. In order to account for that, we need to study the evolution of latent spaces at the self-attention level along training.

## 7.3 QUERIES AND KEYS: TRAINING DYNAMICS

### 7.3.1 PARALLEL DRIFT OF QUERIES AND KEYS

We have established that manually pushing for drift-based anisotropy on *untrained* Transformers models leads to sharper (i.e. low-entropy) self-attention patterns. In this section, we show that this evolution of self-attention values actually takes place during training, and we explore the mechanism behind their appearance. As pointed out in Section 7.2, the self-attention scores result from the  $Q^h(K^h)^T$  operation, which computes scalar products between query and key representations corresponding to each pair of positions. Thus, in this section, we study the evolution of these query and key representations *along training*, and explore the mechanism behind the increase of the scalar products leading to self-attention scores.

We use the MultiBERT checkpoints (Sellam et al., 2022) with seed 0 to retrieve  $Q^h$  and  $K^h$  distributions at different pretraining steps, and we use 128 samples from Wikitext-103 as input data. Along this section,  $Q_s^h$  and  $K_s^h$  refer to query and key representations extracted at a specific layer and head at a given step  $s$ , and  $\hat{Q}_s^h$  and  $\hat{K}_s^h$  are the average representations, taken over all tokens in the sampled batch. By studying  $Q_s^h$  and  $K_s^h$ , we aim at exploring the common (or context-agnostic) drifts of keys and queries distributions.

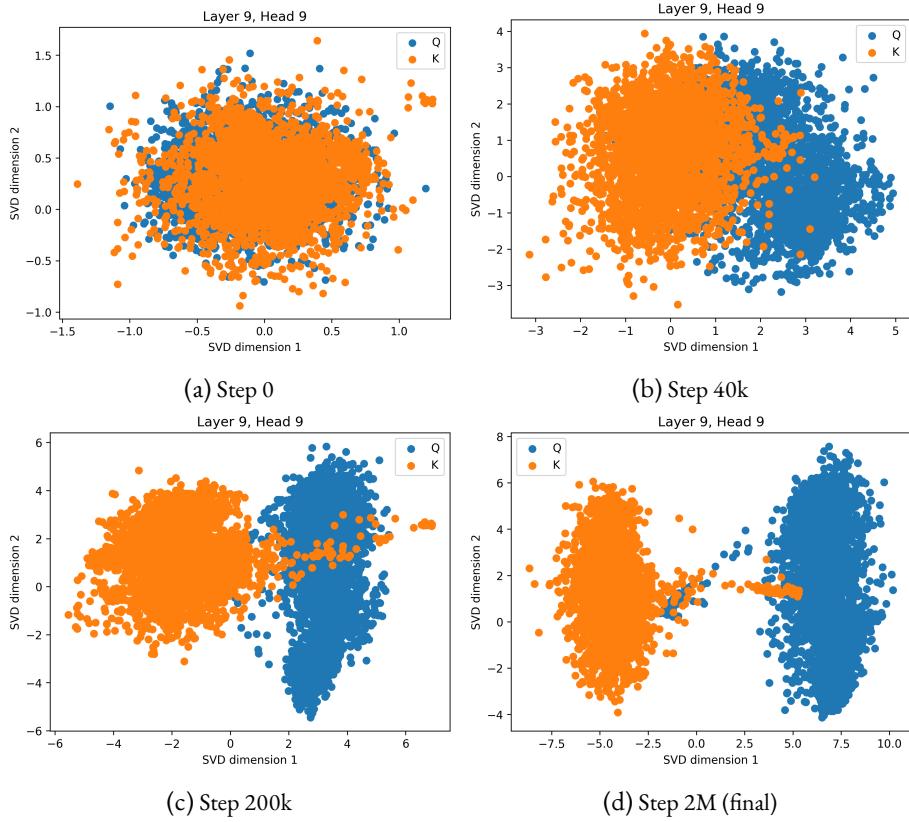


Figure 7.12: Evolution of  $Q_s^h$  and  $K_s^h$  distributions along training (on layer 9 and head  $h = 9$ ). Vectors are projected using a common SVD.

In Figure 7.12 and Figure 7.13, we compute a SVD of the union of  $Q_s^h$  and  $K_s^h$  for all steps  $s$ , so that the projection makes sense for both distributions across steps for visualization purposes<sup>1</sup>. As shown in our selected examples, we observe that the dynamics of  $\bar{Q}_s^h$  and  $\bar{K}_s^h$  tend to align along training, making the average of the distributions drift in either similar or opposite directions. The first dimension of the SVD seems to describe this common drift. Note that in  $\mathbb{R}^{d_h}$  ( $d_h = 64$  being the head dimension), such an alignment is very unlikely to happen randomly. Interestingly, Figure 7.13a shows that the common direction dynamics appear in the first few steps, while the opposite direction dynamics of Figure 7.13b only starts after 8% of the total training steps.

<sup>1</sup>We actually uniformly sample 20% of the whole set of representations to compute the SVD under reasonable memory constraints.

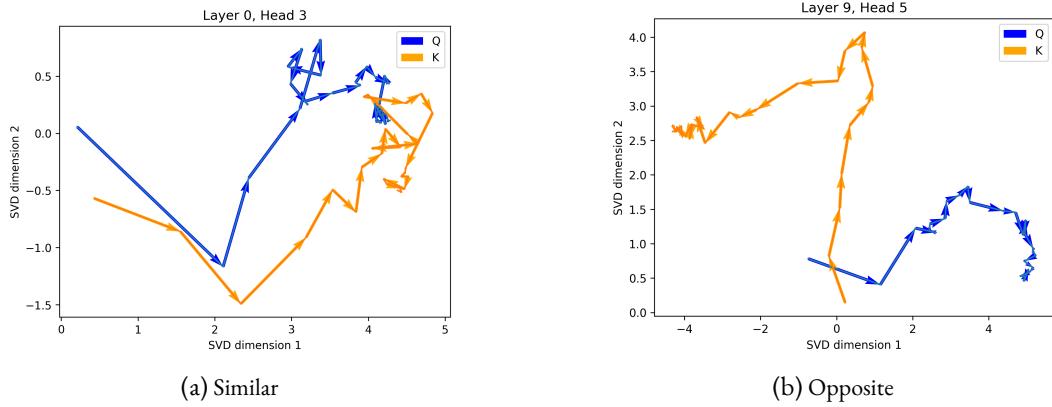


Figure 7.13: Evolution of  $\bar{Q}_s^h$  and  $\bar{K}_s^h$  along training for two different heads in the network, projected via common SVD. Each arrow represents a checkpoint in the MultiBERT suite. We display typical examples of dynamics in same/opposite direction.

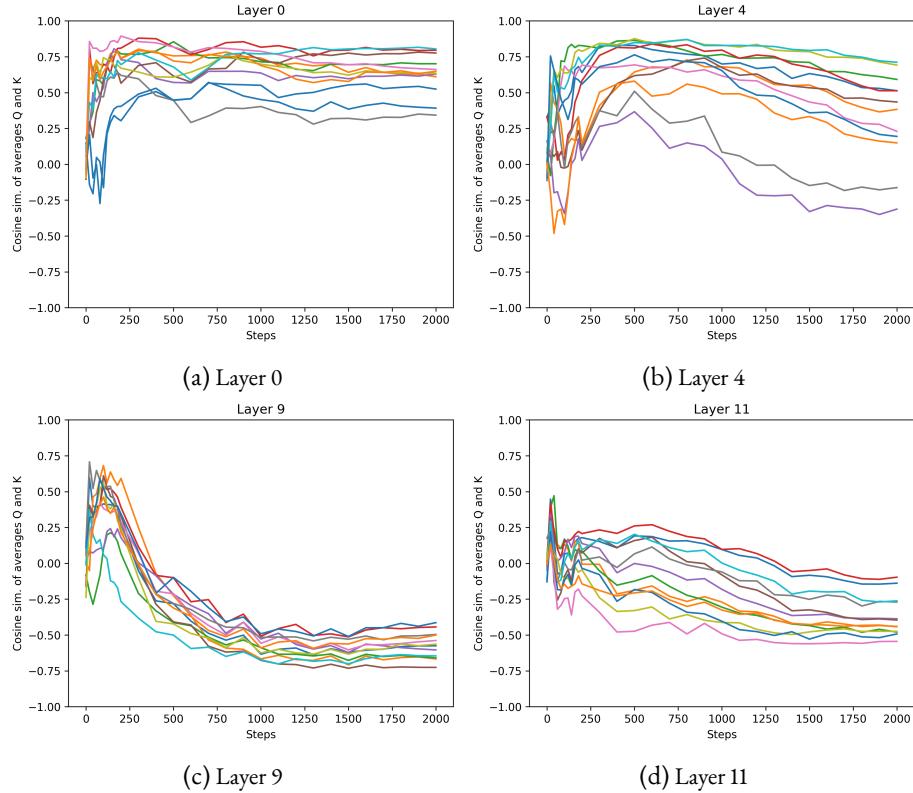


Figure 7.14: Evolution of cosine-similarity between  $\bar{Q}_s^h$  and  $\bar{K}_s^h$  along training. Each color represents one self-attention head. Steps are counted in thousands. We generally observe that almost all heads see  $\bar{Q}_s^h$  and  $\bar{K}_s^h$  align in common or opposite directions along training. In other words, the average components of keys and queries representations tend to align in self-attention heads, which maximizes the magnitude of the scalar product between two average representations. We run a similar experiment on all MultiBERT seeds in Figure 5, and obtain comparable results.

To consolidate our observations, we compute the evolution of the cosine-similarity between  $\bar{Q}_s^h$  and  $\bar{K}_s^h$  along training in Figure 7.14. We also display some projected  $Q_s^h$  and  $K_s^h$  distributions for several  $s$  steps in Figure 7.12.

Figure 7.14 shows that the first layers display a common direction dynamic, as the cosine-similarity tends to increase, thus showing that **the key and query distributions drift along a similar direction** in average. The last layers seem to adopt an opposite direction dynamic, as the cosine-similarity between their mean key and query representations gets negative along training.

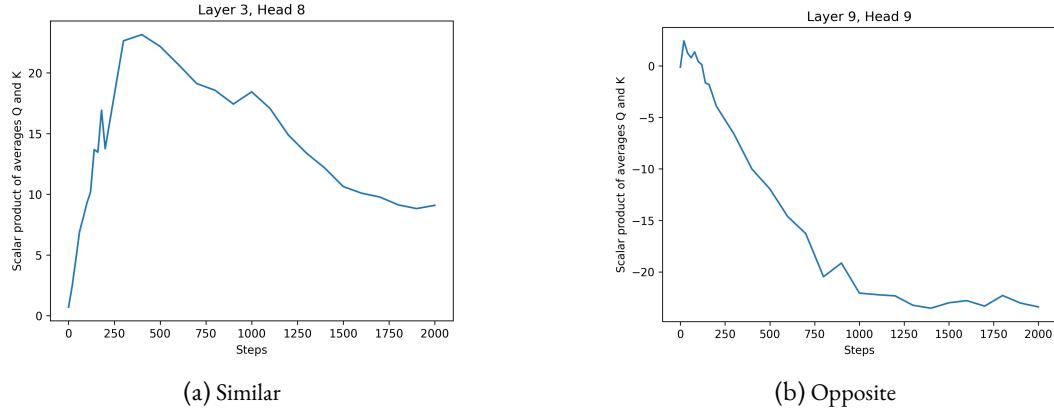


Figure 7.15: Evolution of the scalar product between  $\bar{Q}_s^h$  and  $\bar{K}_s^h$  along training. Steps are in thousands.

As shown in Figure 7.15, this drift induces an increase in the magnitude of scalar products obtained in the self-attention  $Q^h K^{hT}$  operation, thus facilitating the emergence of sharp patterns where attention focuses on specific tokens.

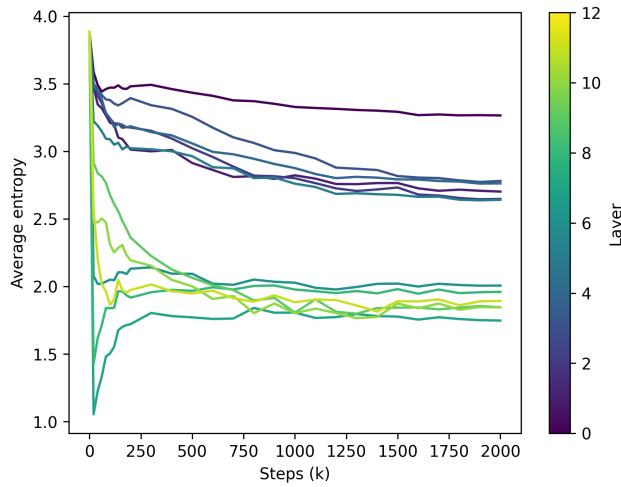


Figure 7.16: Average entropy of the probability distributions corresponding to self-attention rows along training. Each curve corresponds to one layer.

Finally, Figure 7.16 describes the evolution of the average entropy in self-attention distributions. We observe that training induces an overall decay of the entropy for all layers, with different dynamics. This corresponds to sharper self-attention distributions. It is interesting to notice that the distributions in the first layers remain sharper than the ones in the last layers.

### 7.3.2 RELATION BETWEEN SPARSITY AND ANISOTROPY

Although there seems to be a clear correlation between parallel drift and sparsity in practice, we did not expose a clear causation between the coupled degeneration of queries and keys and the observed sparsity of attention maps. As a matter of fact, this parallel drift should only cause a bias in the attention values, which in turn has no incidence on the subsequent softmax-normalized attention map.

To better understand this phenomenon, we need to precisely characterize the structure of the sparsity in attention maps. In Figure 7.17, we compute the attention maps of TinyLlama (Zhang et al., 2024), a 1B parameter causal language model based on the Llama suite (Touvron et al., 2023).

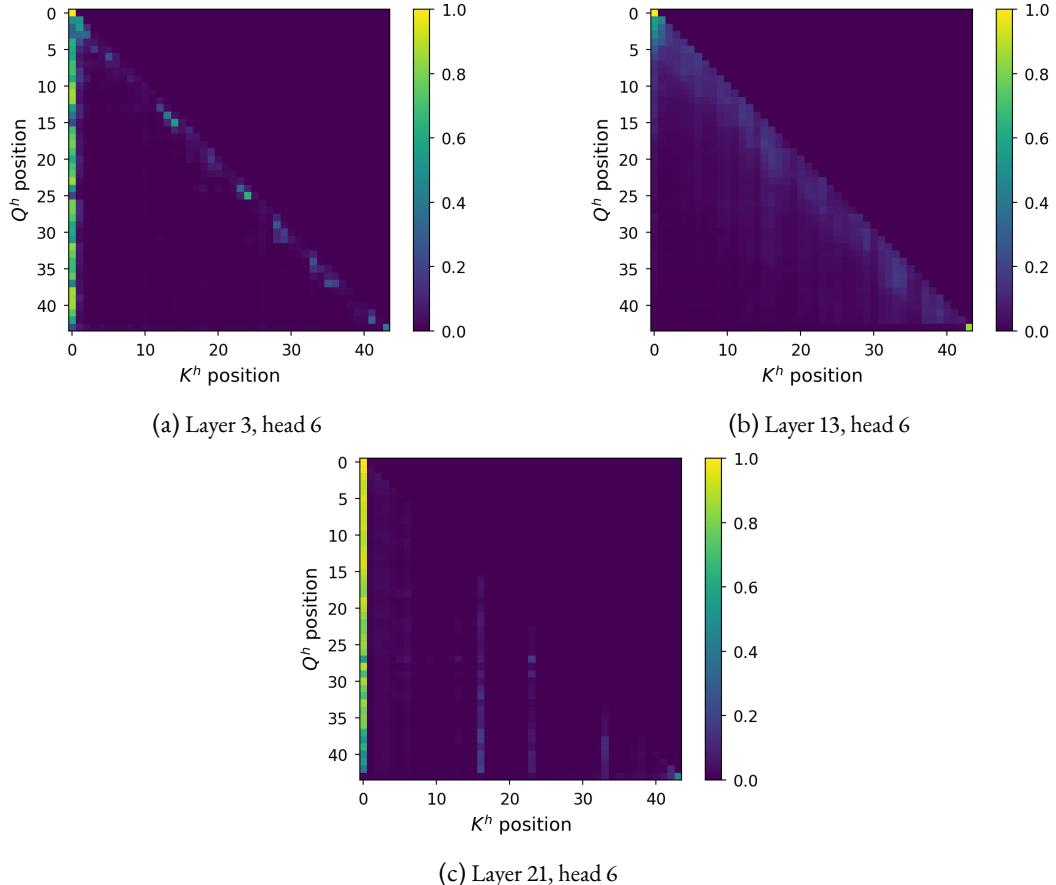
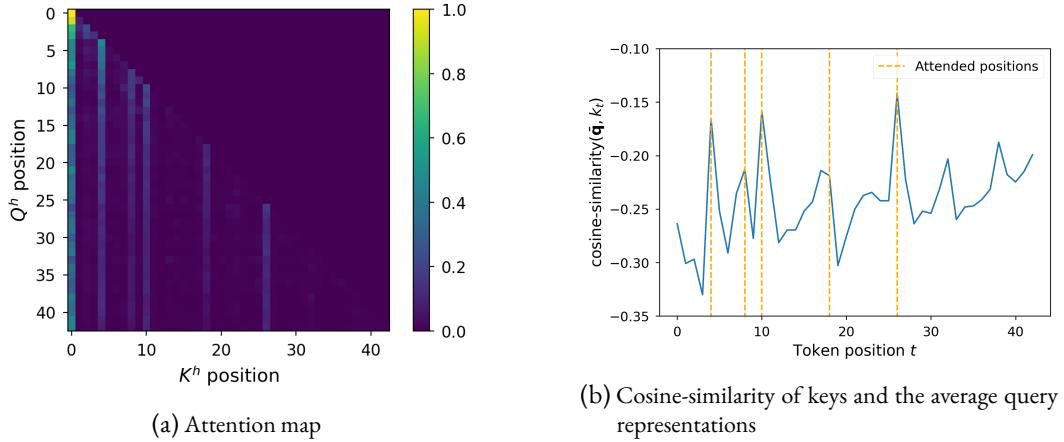


Figure 7.17: TinyLlama’s attention maps for different layers and heads.

As shown in [Figure 7.17](#) attention maps are mostly sparse, following patterns where self-interaction dominate ([Figure 7.17a](#)), where interactions are mostly active in a short context window before the current position ([Figure 7.17b](#)), or where some positions are attended by all future ones ([Figure 7.17c](#)).

We start by analyzing this latter case through the anisotropy lens. In [Figure 7.18](#), we measure cosine-similarity between the average query representation  $\bar{Q}^h$  and the key representations  $K_t^h$ , for an attention head that behaves accordingly.



[Figure 7.18](#): Analysis of TinyLlama’s attention map for layer 2 and head 9.

[Figure 7.18](#) illustrates how the cosine-similarity of keys with the average query component allows to differentiate between attended and ignored tokens. In other words, the norm of the projections of key vectors  $K_t^h$  in the common drift direction represented by  $\bar{Q}^h$  allows to separate between discardable positions and important ones. Most interactions being ignored by the attention pattern, we observe that most of the measured cosine-similarities are negative.

For attention patterns that resemble [Figure 7.17b](#), we need to use a different approach as the attention value seems to depend on both key and query indices. In [Figure 7.19](#), we propose to select a several fixed query indices and to measure the cosine-similarity with past keys.

In [Figure 7.19](#), we remark a similar pattern than in [Figure 7.18](#), where all interactions correspond to negative cosine-similarity, and where dominant interactions correspond to higher cosine-similarity levels. As most interactions are apparently discarded in this attention pattern,  $\bar{Q}^h$  representations need to have negative cosine-similarity with an increasing number of  $K_t^h$  as we move further into the sequence, which reinforces an overall alignment of these representations.

Hence, the parallel drift of queries and keys representations facilitates the emergence of a common direction that encodes a selective process in the self-attention patterns, where discardable interactions lead to lower cosine-similarity values.

Overall, this section shows that drift anisotropy emerges in the query and key representations during the training of MultiBERT, as self-attention distributions become sharper. The drifts of queries and keys tend to align, thus increasing the magnitude of scalar products, and the general sharpness of self-attention.

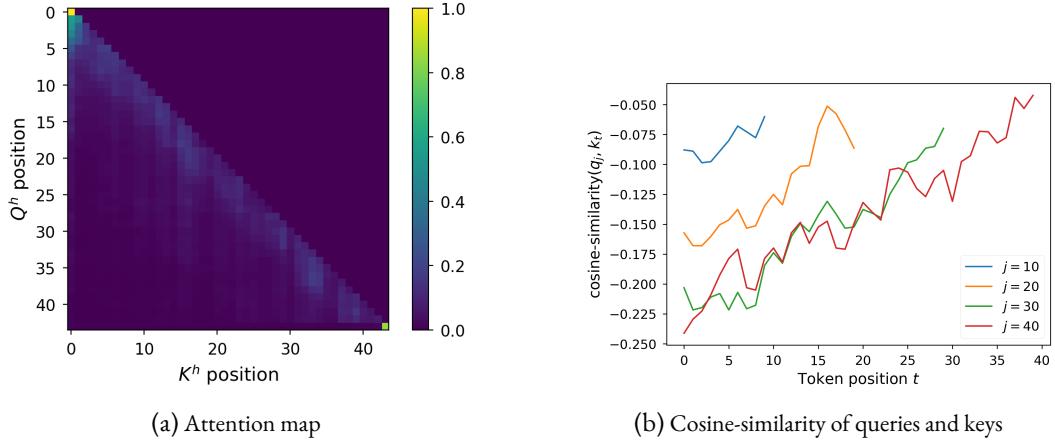


Figure 7.19: Analysis of TinyLlama’s attention map for layer 13 and head 6.

Although this section focuses on the case of token-based NLP, we believe that strong attention patterns may be required when training Transformers across all modalities, potentially generating distortions in query and key distributions that account for the final observed anisotropy of the models. However, we could not extend experiments to other modalities due to the lack of released intermediate checkpoints, to the best of our knowledge.

## 7.4 DISCUSSION

In this work, we argue that the nature of data distributions is not solely responsible for the anisotropy observed in most hidden representations of Transformers-based models across modalities. As Section 7.2 shows, untrained Transformers layers display a tendency towards anisotropy. Biased inputs tend to increase the variance of the attention scores and thus facilitate the emergence of sharp patterns in the self-attention mechanisms. We also show in Section 7.3 that along training, query and key distributions drift in parallel directions, which increases anisotropy in the inner representations of the Transformer layers, while allowing sharper attention patterns. As discussed in Puccetti et al. (2022), outlier dimensions in Transformers are also involved in the emergence of strong attention patterns.

**CONSISTENCY OF THE SVD** In Section 7.3, we use an SVD on the *union* of  $Q_s^h$  and  $K_s^h$  for visualization purposes (see Figure 7.12 and Figure 7.13). It may be argued that this approach favors the emergence of a discriminative singular direction, that helps distinguish between keys and queries, thus supporting the findings in a less convincing way. To address this concern, we display alternative projections in Appendix A.1, where we compute the SVD on  $Q_s^h$  or  $K_s^h$  only, and then project all representations using this SVD. Our observations show that our findings are consistent for these alternative projections.

**HARMFULNESS OF ANISOTROPY** Even though anisotropy has not been shown to be an issue in language modeling, previous works have advocated that removing anisotropy in output repre-

sentations leads to better sense disambiguation abilities (Bihani and Rayz, 2021; Biš et al., 2021). Isotropic models could also improve cross-lingual alignment in multilingual language models (Hämmerl et al., 2023b). Nevertheless, concurrent works have suggested that anisotropy may not hurt the quality of the representations (Ait-Saada and Nadif, 2023; Rudman and Eickhoff, 2024b). We argue that anisotropy in the Transformer architecture may actually help models by allowing sharp attention patterns, but we also believe that our work can pave the way for new architectures that can easily use sharp attention patterns without inducing anisotropy.

## CONCLUSION

In this paper, we investigated the anisotropy problem through the lens of the drift effect, and made several contributions to the understanding of this phenomenon. We demonstrated that anisotropy can be observed in language models with character-aware architectures, extended our observations to Transformers trained on other modalities, and studied anisotropy in untrained Transformers layers. We finally explored the training dynamics of the query and key distributions, and found that they drift along a shared direction hence maximizing  $Q^h K^{hT}$  scalar products in absolute value, allowing stronger attention patterns as a result.

We conclude that anisotropy almost systematically affects Transformers on all modalities, in a way that is not always correlated with the drift of the representations. We also provide empirical evidence that anisotropy appears as an inherent property of latent distributions used in the self-attention mechanism when modeling sharp attention patterns. We hypothesize that a revision of the self-attention operation could help reduce anisotropy by facilitating the emergence of sharp attention softmax distributions without distorting the geometry of the hidden representations.

## LIMITATIONS

As mentioned in Section 7.4, we acknowledge that Section 7.2 does not take into account the training dynamics, and only exposes some properties of the Transformer layer at initialization.

Moreover, we are aware that our approach is not theoretically rigorous in some aspects. For instance, we don't prove that sharp self-attention patterns *cannot* emerge without anisotropy in keys and queries representations. In other words, this article is focusing on exposing and *correlating* factors that explain anisotropy, but we do not demonstrate theoretical properties that would help identify the *causes* of anisotropy. Nevertheless, we believe that our work can pave the way for such theoretical exploration in the future.

In this section, we show that representation degeneration happens in the self-attention layers and co-occurs with the sparsification of attention patterns, regardless of the data modality. This incentivizes the analysis of representation geometry as a way to better understand these implicit biases and paths towards how to improve them.



# CONCLUSION

Overall, studying distortions and biases in the representation space has allowed us to shed light on bottlenecks and limitations that are inherent to the classical language modeling framework. It also provided insights about the architecture of modern language models, from the dimensionality and sparsity perspectives.

Beyond the scope of usual interpretability frameworks, that are designed to explain predictions from targeted observations, we advocate for tools that allow analyzing global behaviors of the inner states of language models, in order to provide guidelines towards better paradigm for learning models of natural language.

Finally, we underline that our work, especially [Chapter 5](#) and [Chapter 6](#), shows that representation degeneration and frequency-related biases hurt the quality of affected language models, either by degrading their performance or incorporating knowledge bias. In [Part III](#), we propose several methods aimed at avoiding degeneration, reducing frequency dependency and mitigating sparsity in language models, in the hope that these methods will indirectly mitigate the identified limitations that are correlated with these phenomena.



## PART III

### EXTENSIONS OF THE LANGUAGE MODELING PARADIGM

In this part, we leverage the insights provided in [Part II](#) in various ways and derive methods that mitigate the identifies limitations of the classical language modeling framework. Our approaches yield several benefits in robustness, performance, and efficiency.



# 8

## HEADLESS LANGUAGE MODELS: LEARNING WITHOUT PREDICTING WITH CONTRASTIVE WEIGHT TYING

Natural Language Processing (NLP) has seen tremendous progress in recent years thanks to the development of large-scale neural language models. These models have been shown to be effective in a wide range of NLP tasks such as text classification, question answering, and machine translation, either in fine-tuning, few-shot and zero-shot settings. These approaches usually involve a self-supervised pre-training step, based on tasks requiring predictions of contextual probability distributions over a large vocabulary of tokens.

However, the need for a language modeling projection head can be a limitation as it requires additional memory, slows down training and impedes scaling up to large token vocabularies. In this chapter, we propose a novel pretraining approach called Headless Language Modeling, which removes the need to predict probability distributions and rather focuses on leveraging contrastive learning to reconstruct sequences of input embeddings. Instead of adding a projection head towards a high-dimensional vocabulary space in order to make a prediction about a given token, we teach those models to contrastively output static embeddings corresponding to this token. The static embeddings we use for this are the model’s own input embeddings. Due to its resemblance with the well-established weight-tying trick (Press and Wolf, 2017; He et al., 2023), we call this pre-training technique *Contrastive Weight Tying* (CWT).

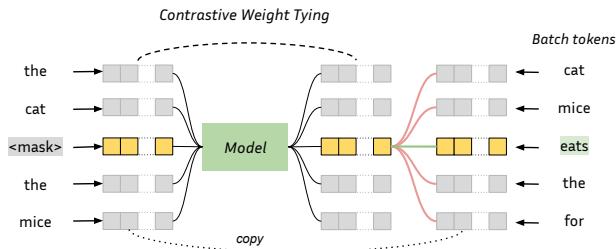


Figure 8.1: Masked Headless Language Modeling (HLM) using Contrastive Weight Tying. The CWT objective aims to contrastively predict masked input representations using in-batch negative examples.

We find that our approach outperforms usual language modeling counterparts in several aspects and by substantial margins. First, it drastically speeds up training by freeing up GPU memory and avoiding the costly language modeling projection, thus allowing up to  $2\times$  acceleration of the training throughput, and up to  $20\times$  less compute requirements to achieve similar performance.

Moreover, given the same amount of training tokens, headless language models (HLMs) significantly outperform their classical counterparts on downstream tasks, as shown by a 2.7 gain in LAMBADA accuracy for our headless generative model. Finally, given similar compute budgets, HLMs bring substantial gains for NLU tasks, with our BERT reproduction scoring 1.6 points above its classical counterpart on the GLUE benchmark. We also show that headless models can benefit from larger token vocabularies at a much more reasonable cost than classical models.

In terms of implementation<sup>1</sup>, our approach can be used as a drop-in replacement in usual pretraining codebases, as it only requires a change in the loss computation that can be applied to any kind of language model.

Overall, we make several contributions in this chapter:

- We introduce a pretraining objective that replaces cross-entropy, thus removing the need to project on the vocabulary high-dimensional space and instead learning to contrastively predict latent representations of tokens;
- Using this technique, we pretrain encoder and decoder models for English, and a multilingual encoder model;
- We show the various benefits of headless training in terms of data-efficiency, compute-efficiency, and performance;
- We explore the effects of micro-batch size and vocabulary size on downstream performance, and provide an ablation study of our contrastive objective.

## 8.1 RELATED WORK: TOKEN-LEVEL CONTRASTIVE LEARNING

**CONTRASTIVE APPROACHES IN NLP** The idea of relieving language models of the need to predict probabilities over the whole token vocabulary has been explored in the importance sampling literature (Bengio and Senecal, 2003; Mnih and Teh, 2012; Jean et al., 2015; Ma and Collins, 2018). These methods approximate the denominator of the softmax by using only a subset of the possible tokens. Those approaches usually rely on variants of the Noise-Contrastive Estimation objective (Gutmann and Hyvärinen, 2010) that use unique negative samples, contrary to our approach that samples representations uniformly from the batch. Kumar and Tsvetkov (2019) and Tokarchuk and Niculae (2022) use contrastive objectives based on cosine-similarity to match pre-trained static embeddings for Machine Translation. We instead use the model’s input embeddings as trainable target representations.

**CONTRASTIVE SELF-SUPERVISED LEARNING** The Contrastive Predictive Coding loss (van den Oord et al., 2019b) initiated the use of pretraining approaches based on a contrastive learning objective, an idea that has obtained success in many modalities over the years (Sermanet et al., 2018; Schneider et al., 2019; Baevski et al., 2020b; Algayres et al., 2022). In NLP, contrastive learning has proven efficient in the training of sentence-level models (Gao et al., 2021; Yan et al., 2021; Klein and Nabi, 2023). Token-level approaches rely on contrastive auxiliary objectives that are added

---

<sup>1</sup>Our pretraining and fine-tuning code is published in <https://github.com/NathanGodey/headless-lm>

to the usual cross-entropy loss. SimCTG (Su et al., 2022a) introduces a token-level contrastive objective using in-batch output representations as negative samples, and adds this objective to a sentence-level contrastive loss and a regular causal LM loss. TaCL (Su et al., 2022b) relies on a similar technique for encoder models, where a teacher model is used to produce negative samples. ContraCLM (Jain et al., 2023) uses an auxiliary contrastive loss for code generation.

Our work differ by using a token-level contrastive objective not as an approximation of cross-entropy as in importance sampling, or as a regularizing auxilary loss as in Su et al. (2022a), but rather as a standalone token-level representation learning objective. The resulting representation models can then be fine-tuned for downstream classification and regression tasks, or for language modeling.

## 8.2 METHOD

### 8.2.1 CLASSICAL FRAMEWORK

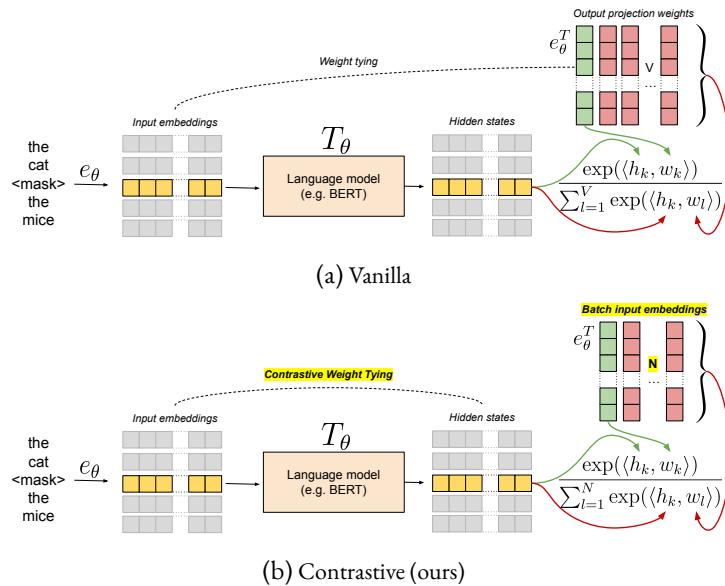


Figure 8.2: Schematic comparison of the classical weight tying approach and the Contrastive Weight Tying loss. The green embedding corresponds to the masked token.

We consider a batch  $\mathcal{B} = (w_t^i)_{i \in [1, N], t \in [1, L]}$  of  $N$  token sequences of length  $L$ . We also produce a slightly altered version of these sequences  $\tilde{\mathcal{B}} = (\tilde{w}_t^i)_{i \in [1, N], t \in [1, \tilde{L}]}$ , optionally using masking or random replacement for instance, as some pretraining objectives require. We introduce an embedding matrix  $e_\theta \in \mathbb{R}^{V \times d_m}$  where  $V$  is the token vocabulary size and  $D$  is the hidden dimension, and a sequence-to-sequence model  $T_\theta : \mathbb{R}^{N \times L \times d_m} \rightarrow \mathbb{R}^{N \times L \times d_m}$  both based on a set of parameters  $\theta \in \mathbb{R}^P$ .

A classical language modeling approach consists in selecting a subset of tokens  $\mathcal{B}_S = (w_t^i)_{i, t \in S}$ , and then estimating a probability distribution over the token vocabulary for these tokens from

the  $(w_t^i)$  sequences, using  $e_\theta$  and  $T_\theta$ . Learning occurs as  $\mathcal{B}_S$  is partially altered in  $(\tilde{w}_t^i)$  (e.g. in Masked Language Modeling) or internally in  $T_\theta$  (e.g. decoder models), and contextual information is essential for  $e_\theta$  and  $T_\theta$  to accurately estimate the tokens in  $\mathcal{B}_S$ .

A trick that has been used in many such approaches relies on using  $e_\theta$ 's transpose ( $e_\theta^T$ ) as a projection from the output space of  $T_\theta$  to  $\mathbb{R}^V$ . This approach, called weight tying (Press and Wolf, 2017), can be written for a given sequence at index  $i \in [1, N]$  as:

$$\hat{p}_t^i = \sigma\left(e_\theta^T \left(T_\theta(e_\theta(\tilde{\mathbf{w}}^i))_j\right)\right)$$

where  $\hat{p}_t^i$  is the estimated distribution for the  $t$ -th token of the  $i$ -th sequence. Weight tying has been shown to improve performance while reducing the number of parameters (Clark et al., 2020). Cross-entropy loss is then used as an objective function:

$$\mathcal{L}(\theta, \mathcal{B}, \tilde{\mathcal{B}}) = -\frac{1}{|\mathcal{S}|} \sum_{i, t \in \mathcal{S}} \mathbf{1}_{w_t^i} \cdot \log(\hat{p}_t^i)$$

### 8.2.2 HEADLESS MODELING

While weight tying does not use additional parameters, the projection  $e_\theta^T$  actually has a non-negligible computational cost, which increases as the token vocabulary grows. Like Gao et al. (2019b), we advocate that the weight tying approach tends to maximize the scalar product between the input embedding of the original token  $e_\theta(w_t^i)$  and the output representation at the same position  $o_t^{\theta, i} = T_\theta(e_\theta(\tilde{\mathbf{w}}^i))_t$ , under the contrastive regularization of the softmax function.

Based on this understanding, we design an objective that directly optimizes this scalar product while not requiring the computation of the  $e_\theta^T$  projection. As we do not use this projection, we cannot rely on softmax regularization anymore, and instead introduce a contrastive loss using the in-batch samples from  $\mathcal{S}$  as negatives. All in all, our contrastive loss can be written as:

$$\mathcal{L}_c(\theta, \mathcal{B}, \tilde{\mathcal{B}}) = -\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \log \sum_{t \in \mathcal{S}^i} \frac{e^{o_t^{\theta, i} \cdot e_\theta(w_t^i)}}{\sum_{k, l \in \mathcal{S}} e^{o_t^{\theta, i} \cdot e_\theta(w_t^k)}}$$

We call this objective *Contrastive Weight Tying* (CWT), as weight sharing is not used *per se* but is set as a contrastive objective. Across the chapter, we *do not combine* this loss function with the classical cross-entropy objective as in Su et al. (2022a), and rather use it as the only pretraining objective. To the best of our knowledge, this work stands as the first attempt to pretrain language models in a self-supervised fashion using an explicit contrastive loss as the sole objective.

### 8.2.3 THE CASE OF DECODERS: CAUSAL FINE-TUNING

We can easily adapt the Causal Language Modeling (CLM) objective using the Contrastive Weight Tying approach. Negative samples correspond to every input embedding at a different position in the batch. However, the resulting model is not directly able to generate text, as it has no projection head towards  $\mathbb{R}^V$ . A way to retrieve language generation capacities is to use the input embedding matrix transpose  $e_\theta^T$  as a projection head (Kumar and Tsvetkov, 2019; Tokarchuk

and Niculae, 2022). Nevertheless, we observe that this approach yields poor performance (see Table 8.4). Instead, we fine-tune the headless model and a language modeling head initialized with  $e_\theta^T$  using the predictive CLM objective on a small portion (<2%) of the pre-training dataset. This method allows recovering an effective language model.

#### 8.2.4 THEORETICAL CONSIDERATIONS

In terms of time and memory complexity, Headless Language Models (HLMs) are more efficient than classical language models under usual conditions. If we focus on the computation of the loss *on a single device* from  $|\mathcal{S}| = K$  output representations, a neural probabilistic LM requires  $O(Kd_mV)$  operations while our headless approach performs  $O(K^2d_m)$  operations<sup>2</sup>. Hence, when  $K < V$ , which is very common for micro-batch sizes that fit on one device, our CWT loss is more computationally efficient than cross-entropy. With regard to memory requirements, our CWT loss is also more efficient than its classical counterpart. On the one hand, the cross-entropy loss with weight tying stores the outputs of the  $e_\theta^T$  projection of dimension  $K \times V$  in the forward pass. On the other hand, our CWT loss stores the scalar product matrix of dimension  $K \times N$ , which is again smaller when  $K < V$ .

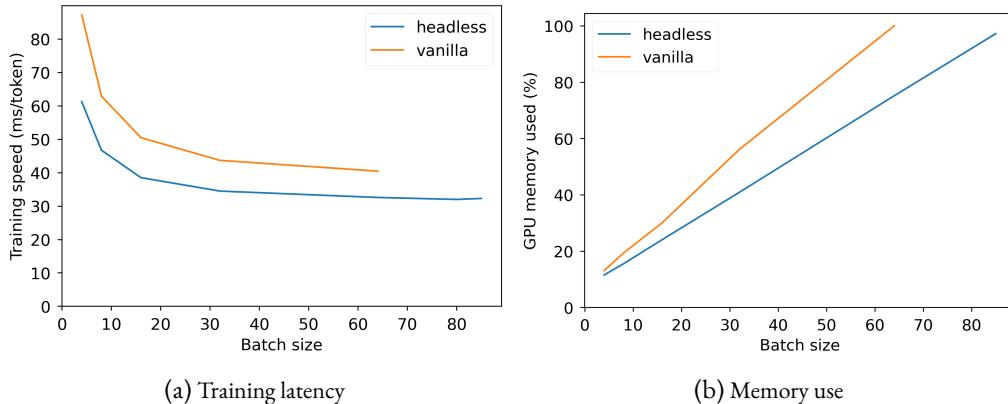


Figure 8.3: Comparison of time and memory complexities of a BERT-base model on a single RTX 8000 GPU.

In Figure 8.3, we provide a preliminary empirical analysis of the speed and memory improvements when training a BERT-base model using original hyperparameters, i.e. sequences of 512 tokens and 15% masking. We use HuggingFace’s implementation for the Transformers blocks, and run experiments on a single RTX 8000 GPU. We observe that training latency is significantly reduced by roughly 25% for all batch sizes, and that the engine can handle a larger batch size due to the improvement in memory consumption.

<sup>2</sup>One could extend our CWT loss by picking a separate set  $\mathcal{S}_N$  of negative samples. This allows to tune the number of negative samples, which is important in Contrastive Learning. However, for the sake of simplicity, and to avoid extensive hyperparameter tuning, we set  $\mathcal{S}_N = \mathcal{S}$ .

## 8.3 EXPERIMENTS

We use the Contrastive Weight Tying objective for medium-scale pre-training experiments in different contexts. We focus on monolingual encoder and decoder architectures, but we also train one multilingual encoder as we believe the uniformity brought by our contrastive objective may improve cross-lingual alignment. We compare our HLMs with classical language models that we pretrain on the same data with roughly similar compute budgets.

### 8.3.1 HEADLESS MONOLINGUAL ENCODER

We pretrain BERT-base architectures (110M parameters) for English on the OpenWebText2 dataset extracted from The Pile (Gao et al., 2020). We use the tokenizer from the Pythia suite (Biderman et al., 2023b), which was trained on The Pile and uses a 50k tokens vocabulary. We mostly use hyperparameters from BERT (Devlin et al., 2019), although we remove the NSP objective as in RoBERTa (Liu et al., 2019b). For the sake of simplicity, we use a sequence length of 128 for the whole training. We give a detailed overview of the hyperparameters in Appendix B.1.

We pretrain all models using 8 A100 GPUs, with a budget of roughly 1,000 hours each. To optimize training, we use memory-efficient self-attention as implemented in xFormers (Lefauze et al., 2022) for all experiments. For the vanilla MLM, we set a micro-batch size of 32 for each A100 GPU, then accumulate to the original 256 batch size at optimization level, and train on 1 million batches. For our headless approach, we observed that we could remain within compute budget when using a micro-batch size of 64. Hence, we use an effective batch size of 512 for the headless MLM (HMLM). Although the HMLM uses more pretraining sequences, it does not gain additional information compared to the vanilla MLM as both models perform several epochs on the OpenWebText2 dataset.

We evaluate on the GLUE benchmark, where we exclude the RTE dataset due to high standard deviations in the obtained scores. We fine-tune our models for 10 epochs on every dataset, and compute validation metrics once every fine-tuning epoch. We use the AdamW optimizer with a learning rate of  $10^{-5}$ , a weight decay of 0.01 and a balanced cross-entropy loss objective.

As a matter of fact, we noticed that using balanced cross-entropy loss for fine-tuning could further improve the performance of all our monolingual encoders, and increase the gap between headless models and their vanilla counterparts. We also noticed empirically that it helped stabilize results for smaller datasets such as MRPC and COLA.

Let's consider a classification problem where the class distribution is described by frequencies  $(f_c)_{c \in [1, C]}$ . We can group the cross entropy loss  $\mathcal{L}_{ce}$  as such:

$$\mathcal{L}_{ce}(X, Y) = \sum_{c=1}^C \mathcal{L}_c(X, Y)$$

where

$$\mathcal{L}_c(X, Y) = \sum_{i=1}^N \mathbf{1}_{y_i=c} \cdot \mathcal{L}_{ce}(x_i, y_i)$$

Using this notation, the *balanced cross-entropy loss* can be defined as:

$$\mathcal{L}_{bce}(X, Y) = \sum_{c=1}^C \frac{\mathcal{L}_c(X, Y)}{f_c}$$

In practice, we approximate the ( $f_c$ ) using the batch labels. The purpose of the balanced cross-entropy loss is to mitigate general and in-batch class imbalance.

We reproduce fine-tuning experiments with the more usual categorical cross-entropy loss only, and using moderately optimized hyperparameters for this loss (see Table 6).

MLM type	Balanced loss (FT)	GLUE Avg.
Vanilla	✗	81.5 ( $\pm 0.14$ )
Vanilla	✓	82.43 ( $\pm 0.12$ )
Headless	✗	83.19 ( $\pm 0.09$ )
Headless	✓	<b>84.08</b> ( $\pm 0.14$ )

Table 8.1: Results of Masked Language Models (MLMs) on the dev sets of the GLUE benchmark for different fine-tuning strategies. Results are averaged over 3 runs.

In Table 8.2, we compare our headless MLM with the classical MLM on the GLUE benchmark. To ensure fair comparison, we display evaluations at similar amounts of tokens seen during pre-training, and at similar training durations on the same hardware. In both cases, the headless MLM outperforms the vanilla MLM by significant margins, showing that our CWT loss is both more data-efficient and compute-efficient in this setup. We extend this analysis at various intervals along pretraining, and plot results in Figure 8.4. It shows that the headless MLM outperforms the downstream performance of its vanilla counterpart after using 25% of its training compute. We notice that the performance gap is near constant across pretraining steps.

### 8.3.2 HEADLESS MONOLINGUAL DECODER

We pretrain Pythia-70M architectures for English, sticking to the Pythia procedure (Biderman et al., 2023b) as much as possible. We use OpenWebText2 as a pretraining dataset. We train on 143,000 batches of 1,024 sequences of length 2,048 split over 16 V100 GPUs. We use exactly the same hyperparameters as in the Pythia suite. The micro-batch size is set to 32 in both cases.

MLM type	Tokens (B)	GPU hours	MRPC	COLA	STS-B	SST2	QNLI	QQP	MNLI	Avg.
Vanilla	4.1	989	85.87	54.66	83.7	92.45	88.38	89.57	82.4	82.43 ( $\pm 0.12$ )
Headless	4.1	444	85.31	<u>58.35</u>	84.54	<b>93.23</b>	<u>89.49</u>	<u>89.62</u>	82.54	<u>83.29</u> ( $\pm 0.15$ )
Headless	8.2	888	<b>86.89</b>	<b>60.72</b>	<b>85.98</b>	<u>92.56</u>	<b>89.75</b>	<b>89.81</b>	<b>82.87</b>	<b>84.08</b> ( $\pm 0.14$ )

Table 8.2: Results of Masked Language Models (MLMs) on the dev sets of the GLUE benchmark. Best results are **bold** and second best are underlined. We report Matthews' correlation for COLA, Spearman correlation for STS-B, and accuracy elsewhere. MNLI validation datasets are concatenated. All scores are averaged over 3 different seeds.

MLM type	BoolQ	CB	COPA	WiC	Avg.
Vanilla	68.8	<b>77.8</b>	60.2	64.9	67.9 ( $\pm 0.4$ )
Headless	<b>69.8</b>	74.7	<b>62.7</b>	<b>67.2</b>	<b>68.6</b> ( $\pm 0.6$ )

Table 8.3: Results of Masked Language Models (MLMs) on the dev sets of datasets from the SuperGLUE benchmark. We report accuracy for all tasks. Scores are averaged over 10 fine-tuning runs.

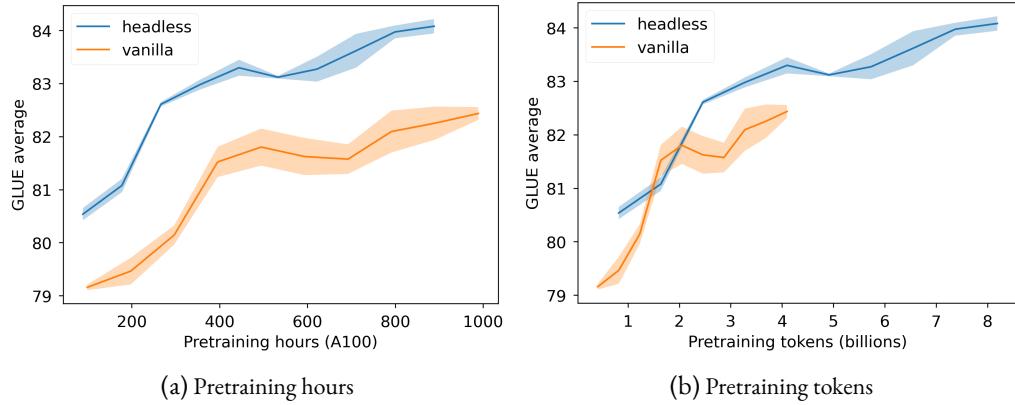


Figure 8.4: Comparison of GLUE average scores along pretraining.

As mentioned in [Section 8.2.3](#), we fine-tune our headless models for CLM with an LM head initialized with  $e_\theta^T$  for 10000 steps using an effective batch size of 256 ( $4 \times$  smaller than during pretraining), a learning rate of  $10^{-4}$ , and a constant learning rate schedule with 2000 linear warm-up steps. All other hyperparameters are kept similar to pretraining. We evaluate our models on the LAMBADA dataset and report accuracy and perplexity for zero-shot generation in [Figure 8.5](#).

We find that the HLM fine-tuned for predictive language modeling outperforms the vanilla model by a significant margin along training. We report language generation results in [Table 8.4](#). We observe that despite having a higher validation perplexity even after fine-tuning, the HLM is improving the zero-shot perplexity on the LAMBADA dataset.

We also study the zero-shot performance of the causal models on datasets taken from the LM Evaluation Harness. At this model scale, many tasks are not relevant and thus discarded, as the results do not always significantly outperform a random baseline. We also discarded tasks where

LM type	Validation		LAMBADA	
	Ppl.	Ppl.	Acc.	
Vanilla	<b>3.143</b>	170.23	19.52	
Headless	-	524.44	18.26	
Headless + FT	3.283	<b>153.5</b>	<b>22.2</b>	

Table 8.4: Results of the causal language models on the validation set after training, and on the LAMBADA dataset.

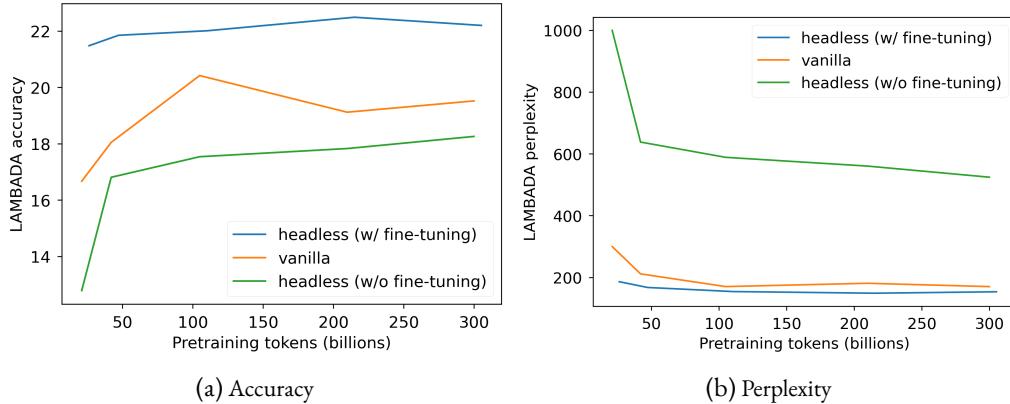


Figure 8.5: Comparison of LAMBADA metrics along pretraining. We display results for vanilla causal language modeling and headless models before and after causal LM fine-tuning. The pretraining token count for the fine-tuned HLM takes fine-tuning tokens into account.

LM type	GPU hours	BoolQ	CrowS-Pairs ↓	RACE	SciQ	PubMedQA	QASPER
Vanilla	1712 (·)	47.8 ( $\pm 0.9$ )	57.3 ( $\pm 1.2$ )	23.7 ( $\pm 1.3$ )	<b>66.4</b> ( $\pm 1.5$ )	43.8 ( $\pm 1.6$ )	41.9 ( $\pm 4.8$ )
HLM + FT	1052 (61%)	<b>53.0</b> $\dagger$ ( $\pm 0.9$ )	<b>56.0</b> ( $\pm 1.2$ )	<b>26.0</b> ( $\pm 1.4$ )	64.5 ( $\pm 1.5$ )	<b>47.5</b> $\dagger$ ( $\pm 1.6$ )	<b>66.0</b> $\dagger$ ( $\pm 3.1$ )

Table 8.5: Zero-shot evaluation of monolingual causal language models on datasets from the LM Evaluation Harness. We report the stereotype percentage for CrowS-Pairs and accuracy elsewhere.  $\dagger$ : best scores that are significantly better than the second best score according to a one-tailed t-test with power 0.95.

the sample size was below 1000 or where comparison was not meaningful due to low performance gaps compared to the variance level. Hence, only a subset of the tasks is shown in Table 8.5.

In Table 8.5, we find that the fine-tuned HLM outperforms the vanilla causal model by significant margins on BoolQ (Clark et al., 2019a), PubMedQA (Jin et al., 2019) and QASPER (Dasigi et al., 2021). Although we observe less statistically significant gaps for the other datasets, we still note that our HLM performs at least comparably to the vanilla baseline. We also note that the HLM seems slightly less prone to stereotypes as measured by the CrowS-Pairs benchmark (Nangia et al., 2020).

Overall, using the Contrastive Weight Tying loss in the context of causal LM allows obtaining models on par with vanilla counterparts at a lower compute cost. We notice that the resulting models can get surprisingly good results in challenging datasets, hence showing language understanding capabilities, while being outclassed in language generation benchmarks (before predictive fine-tuning). We believe that this work shows that language generation needs to be considered as a *downstream task* for HLMs, as they are designed to generate representations instead of words.

## 8.4 MULTILINGUAL ENCODER

In this section, we pretrain small multilingual MLMs and evaluate their performance on the XNLI dataset (Conneau et al., 2018b). Due to compute limitations, we consider architectures similar to

the distilled multilingual BERT<sup>3</sup> trained by Sanh et al. (2019). This model has 137M parameters, and uses a vocabulary of 119k tokens. As in Section 8.3.1, we train a vanilla MLM and a headless counterpart. However, we share training hyperparameters such as batch size and total number of steps between both models, without compute considerations. For both experiments, we pretrain our models on 400k batches of 64 sequences of 128 tokens taken from the multilingual Wikipedia dataset using a single RTX8000 GPU. We select 90 million entries from 10 languages (Arabic, German, English, Spanish, French, Hindi, Italian, Japanese, Korean, and Chinese). Training hyperparameters can be found in Appendix B.3.

Models are then fine-tuned on the XNLI dataset, for both cross-lingual zero-shot transfer from English and target language fine-tuning. Fine-tuning hyperparameters can be found in Appendix C.4.

MLM type	ar	de	en	es	fr	hi	zh	Avg.
<i>Fine-tuned on English only</i>								
Vanilla	46.83	56.71	71.66	59.93	58.34	43.16	50.99	55.37 ( $\pm 0.11$ )
Headless	<b>48.06</b>	<b>57.32</b>	<b>74.03</b>	<b>62.72</b>	<b>62</b>	<b>45.25</b>	<b>52.15</b>	<b>57.36</b> ( $\pm 0.2$ )
<i>Fine-tuned on target language</i>								
Vanilla	51.32	64.09	70.4	66.98	65.88	55.95	64.63	62.87 ( $\pm 0.2$ )
Headless	<b>54.25</b>	<b>66.95</b>	<b>73.96</b>	<b>69.14</b>	<b>67.22</b>	<b>60.04</b>	<b>67.22</b>	<b>65.54</b> ( $\pm 0.22$ )

Table 8.6: Evaluation of multilingual models on the XNLI benchmark. We report dev accuracy, averaged over 3 runs.

We display final results in Figure 8.6. We find that the headless approach leads to significantly better performance for every language in both cross-lingual transfer and language-specific fine-tuning. In average, the headless MLM outperforms its vanilla counterpart by 2 accuracy points in the cross-lingual scenario, and by 2.7 points in the language-specific fine-tuning experiments.

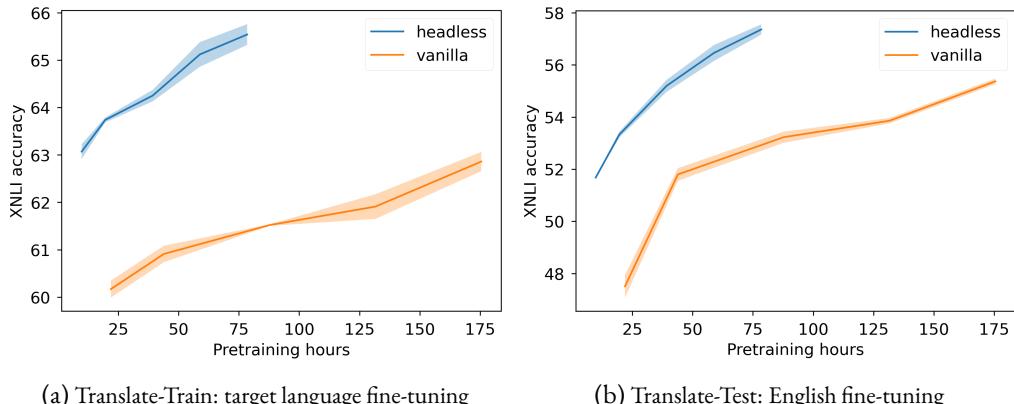


Figure 8.6: Comparison of XNLI average scores along pretraining for different setups. Models are fine-tuned/evaluated in Arabic, German, English, Spanish, French, Hindi and Chinese.

<sup>3</sup>Available at <https://huggingface.co/distilbert-base-multilingual-cased>

In Figure 8.6, we evaluate the models at intermediate pretraining checkpoints and plot the XNLI average score as a function of used GPU hours. We observe that our HLM finishes training within 45% of the time required by the vanilla mode, and that its performance level outperforms the fully trained vanilla model after only using 5% as much compute in Figure 8.6a, and 22% in Figure 8.6b.

## 8.5 DISCUSSION

**TOKEN VOCABULARY** Training language models without output vocabulary projection makes using large vocabularies more affordable in terms of compute. As a matter of fact, the time complexity of HLMs during training is theoretically constant as we increase the vocabulary size. With input embedding lookup tables that do not require fully loading the  $e_\theta$  weights, the memory complexity can also be kept constant with respect to the size of the vocabulary. This property could be useful for multilingual models relying on considerable vocabulary sizes, such as XLM-V (Liang et al., 2023).

To verify this hypothesis, we pretrain models for different vocabulary sizes using the BERT-Small architecture from Turc et al. (2019) and the CC-News dataset (Hamborg et al., 2017). Hyperparameter details can be found in Appendix B.4. For each vocabulary size, we train a BPE tokenizer similar to the BERT tokenizer, and pretrain a vanilla MLM and a headless MLM. We then compare average GLUE results, excluding RTE, MRPC and COLA, due to high variance at that model scale.

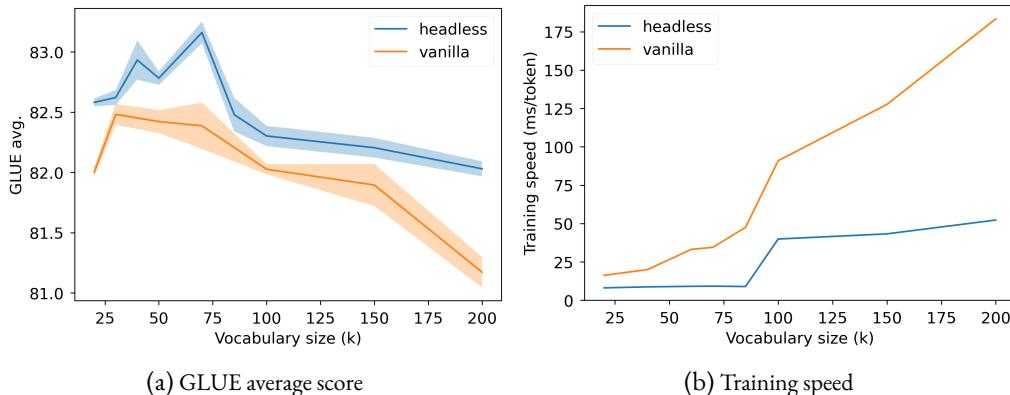


Figure 8.7: Comparison of downstream performance and training speed for small models trained using different token vocabulary sizes.

Figure 8.7 shows that HLMs can actually benefit from larger token vocabularies up to a certain extent, and that they outperform their vanilla counterparts for every vocabulary size. Figure 8.7b demonstrates that increasing the vocabulary size comes at almost no decrease in training speed for the HLMs, contrary to vanilla MLMs. However, we observe a sudden throughput increase between 85k and 100k tokens vocabularies for both vanilla and headless models, which we attribute to a different handling of GPU memory and operations as the models get bigger.

**BATCH SIZE** As discussed in [Section 8.2.4](#), the micro-batch size used to compute the CWT loss is important as it impacts the training complexity by increasing the number of negative samples. Recent work on Contrastive Learning shows that there usually exists an optimal number of negative samples in terms of model performance ([Awasthi et al., 2022](#); [Ash et al., 2022](#)). As a consequence, increasing the batch size when using CWT may not always be beneficial.

To study the impact of batch size on downstream performance, we pretrain small decoder models using different batch sizes. Our models are inspired from the smallest architecture of GPT2 ([Radford et al., 2019](#)) where many hyperparameters are divided by 4. More details about the pretraining procedure of these models can be found in [Appendix B.5](#). HLMs are fine-tuned similarly to [Section 8.3.2](#).

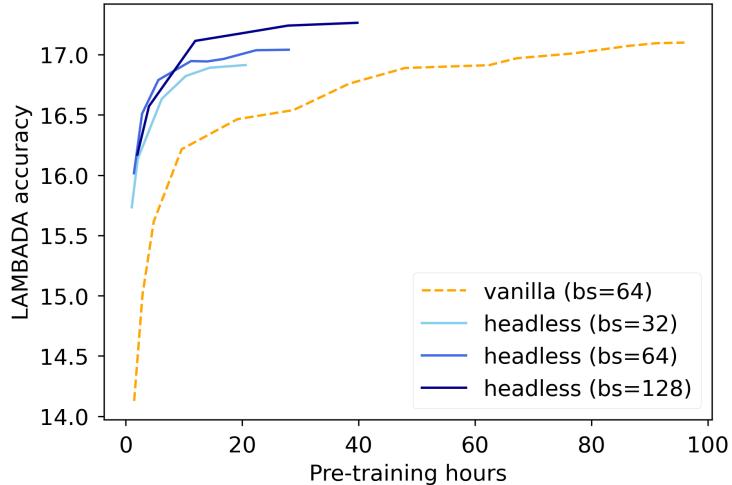


Figure 8.8: LAMBADA accuracy along pretraining for different batch sizes.

In [Figure 8.8](#), we observe that increasing batch size leads to better performance for our HLMs. While smaller batch sizes train even faster, the headless model with the greatest batch size (128) is the only one that is able to significantly outperform its vanilla counterpart at the end of training.

**ABLATION STUDY** In [Table 8.7](#), we conduct an ablation study by training small models using the hyperparameters described in [Appendix B.4](#) for different objectives. We observe that adding Cross-Entropy to CWT leads to slightly worse performance, at the cost of reduced throughput. We also notice that using a contrastive objective without using input embeddings as targets decreases performance, despite adding parameters during training. This shows the relevance of our weight tying approach.

## 8.6 MODELING CONSIDERATIONS

### 8.6.1 TOKEN-LEVEL SIMILARITY & FREQUENCY-RELATED ISSUES

From a linguistic point of view, we hypothesize that an important difference between our approach and classical predictive modeling is the fact that *headless modeling mostly pushes for discrimination*

Objective	Parameters	Throughput ↑	GLUE avg.
Cross-Entropy	x1	x1	82.45
Cross-Entropy + CWT	x1	x0.87	82.93
NCE (wo/ WT)	x1.57	<b>x2.47</b>	82.91
CWT	x1	<b>x2.13</b>	<b>83.37</b>

Table 8.7: Ablation study using variants of the CWT objective. In CWT + Cross-Entropy, we add the objectives without specific weights. In NCE (wo/ WT), we adapt our CWT objective with an additional static embedding matrix instead of the model’s input embeddings, which resembles [Ma and Collins \(2018\)](#).

*between co-occurring tokens*, instead of imposing a contextual hierarchy over the whole vocabulary. For instance, in the case of synonyms A and B, each occurrence of A (or B) is pushing the input representations of A and B apart for predictive modeling, due to weight tying. For headless modeling, an occurrence of A will only push the representations apart if B appears in the same batch. Hence, the CWT objective could let models identify A and B as synonyms more easily. This argument is already mentioned in [Jean et al. \(2015\)](#).

To provide empirical evidence of this behavior, we study the representation similarity for pairs of synonyms for classical and headless models. We use WordNet ([Fellbaum, 1998](#)) to extract synonym pairs and we then compute the cosine-similarity between the input embeddings corresponding to the two synonyms. Resulting cosine-similarity distributions are displayed in [Figure 8.9](#).

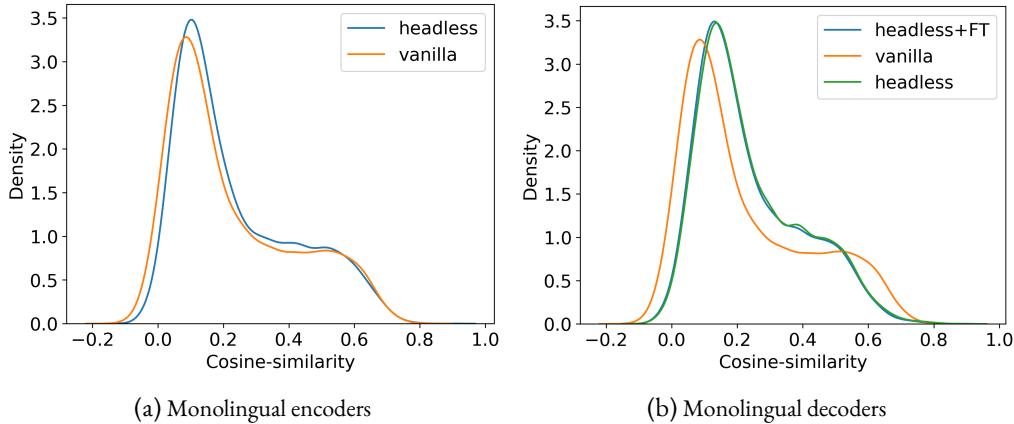


Figure 8.9: Cosine-similarity distributions for pairs of WordNet synonyms.

In [Figure 8.9](#), we observe that HLMs tend to generally represent synonyms in a more similar way than vanilla LMs, as cosine-similarity distributions slightly drift towards higher values. In average, cosine-similarity between synonyms is 1.4 points higher for the encoder and roughly 7 points higher for both the original HLM decoder and its fine-tuned version.

However, we do not observe a radical difference between HLMs and classical LMs in this analysis of the input representations. A more thorough analysis of the latent spaces of both types of models

could be relevant. For instance, comparing contextual representations of similar words across examples could help clarify this matter. We leave such analyses for future work.

Another advantage of pushing discrimination between co-occurring tokens only may be an improved feedback quality, as we expect distinguishing between co-occurring tokens to be more linguistically relevant than distinguishing between all tokens.

Finally, we believe that our method avoids the issue of cross-entropy regarding rare and unused tokens. Gao et al. (2019a) prove that cross-entropy pushes the representations of rare and unused tokens in a shared direction, thus distorting the resulting embedding space. The CWT objective only updates these embeddings when they appear in the negative samples, which should result in more meaningful representations.

### 8.6.2 ANISOTROPY MITIGATION

An implicit consequence of using a contrastive objective with in-batch negative samples is a uniformization of the representational space (Wang et al., 2020a). As a result, it can be expected that the CWT objective leads to more isotropic representations than vanilla cross-entropy, at least on the last layers of trained models. Moreover, using in-batch negative samples reduces the negative update issues mentioned in Gao et al. (2019b) and Biš et al. (2021), i.e. the distortions that are caused by some rare tokens being trained to be dissimilar to frequent ones. Indeed, for CLMs, as tokens used negative samples also appear in the batch, their representations are not only updated to contrast them with positive samples as they are also trained to be aligned with their corresponding ground-truth token representation.

To verify the effect of the CWT loss on anisotropy, we report layer-wise average cosine-similarity (see Section 4.3.3) in Figure 8.10.

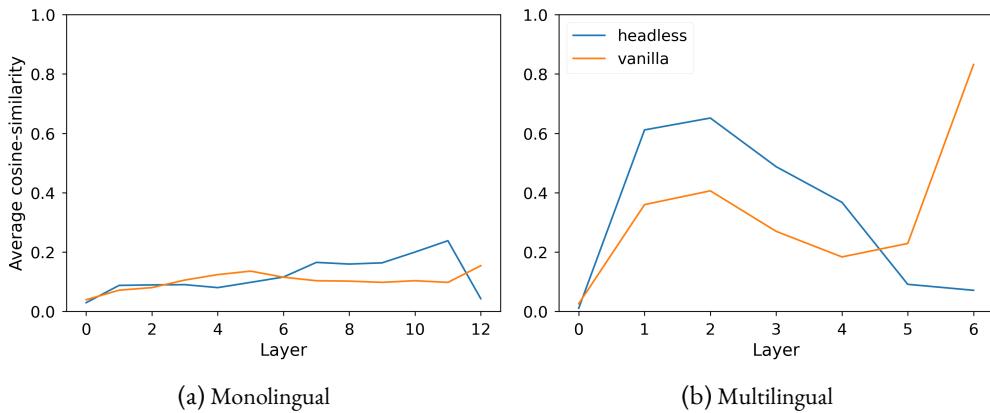


Figure 8.10: Layer-wise anisotropy for monolingual and multilingual MLMs trained with the headless and vanilla objectives.

Figure 8.10 shows that monolingual and multilingual headless models are nearly isotropic on their last layer, as opposed to models based on cross-entropy. Interestingly, both our monolingual pretrained encoder models have rather low anisotropy levels compared to the observations mentioned in Chapter 7. Moreover, in both cases, the anisotropy level in intermediate layers is actually

higher for headless models, which proves that the effect of the loss on last-layer anisotropy is not always propagated to these layers. Hence, the representation degeneration in intermediate layers cannot be safely analyzed only from the loss perspective, as detailed in [Chapter 7](#).

In our experiments, the Contrastive Weight Tying objective successfully mitigates last-layer anisotropy, which may further explain the performance and efficiency gap between headless and vanilla models.

## CONCLUSION

In this chapter, we present a new pretraining approach called headless language modeling, that removes the need to predict probability distributions over token vocabulary spaces and instead focuses on learning to reconstruct representations in a contrastive fashion. Our method only relies on changing the objective function, allowing for straightforward adaptations of classical language modeling pretraining objectives.

Using our contrastive objective, we pretrain headless monolingual and multilingual encoders, and a headless monolingual decoder. We demonstrate that headless pretraining is significantly more compute-efficient, data-efficient, and performant than classical predictive methods. We provide explanations for this improved performance based on representation analysis, as these models better embed semantic similarity and mitigate representation degeneration for the output layer.

A major advantage of our approach is that it enables the use of very large token vocabularies at virtually no increased cost. We believe that this work paves the way for the exploration of contrastive techniques as a replacement of cross-entropy based pretraining objectives for NLP.

## 8.7 LIMITATIONS

One key limitation of this work is the scale of the used architectures. In recent months, the dawn of Large Language Models using billions of parameters reshaped the language modeling paradigm. The research process that led to this work is empirical and required extensive experimentation that could not be done at large scale in our academic compute budget. We believe that the results presented here are still sufficiently promising to be communicated and useful to the community. We leave the scaling of these techniques to future work.

It could be opposed to this work that as architectures grow in size, the proportion of compute that is associated with the output vocabulary projection shrinks. While we acknowledge that this effect may reduce the advantage of HLMs in terms of training throughput, our experiments show that HLMs are more performant for a given number of pretraining steps.

We chose not to compare with other efficient encoder architectures such as ELECTRA or DeBERTa in this chapter. We also chose not to apply our method to encoder-decoder architectures, or to subtle masking methods such as SpanBERT ([Joshi et al., 2020](#)). As a matter of fact, we argue that our work could be combined to these methods, and we thus believe that comparison is not relevant as these works are orthogonal to ours. We leave the intersection of these approaches for future work.

Finally, we decided to pick English for all monolingual experiments. Different behaviors could be observed for other languages, although our multilingual experiments gave no sign of such discrepancies.

This chapter illustrates the effectiveness of pretraining NLP models in a pure representation learning paradigm, aiming at avoiding the distortion and bias phenomena observed in classical language models (see [Part II](#)). In this paradigm, language modeling can be seen as a downstream task, which lets us frame token-level cross-entropy as a fine-tuning objective.

Hence, the resulting language models are still briefly trained using cross-entropy on token distributions, which is prone to the limitations and bottleneck notably mentioned in [Chapter 6](#). In the next chapter, we propose to abstract from the token-level paradigm that implicitly causes frequency-based and dimensional issues in the usual framework.

# 9 EFFICIENT GRADIENT-BASED TOKENIZATION

[Chapter 4](#) and [Chapter 6](#) present several issues that emerge when training language models on contextual token distributions. The choice of a tokenization scheme plays a significant role in the nature of these distributions: longer tokens tend to lead to larger vocabularies  $\mathcal{V}$  and to high-rank contextual log-probability spaces (cf. [Figure 6.8](#) in [Chapter 6](#)), while shorter tokens (e.g. characters) lead to short vocabularies but can drastically increase sequence lengths and the inherent difficulty of the language modeling task.

Hence, designing *efficient* character-level models can be an interesting way forward to avoid phenomena such as the softmax bottleneck described in [Chapter 6](#). In this chapter, we propose a gradient-based pooling module for language models based on character-level tokens, that reduces the overhead of processing longer sequences

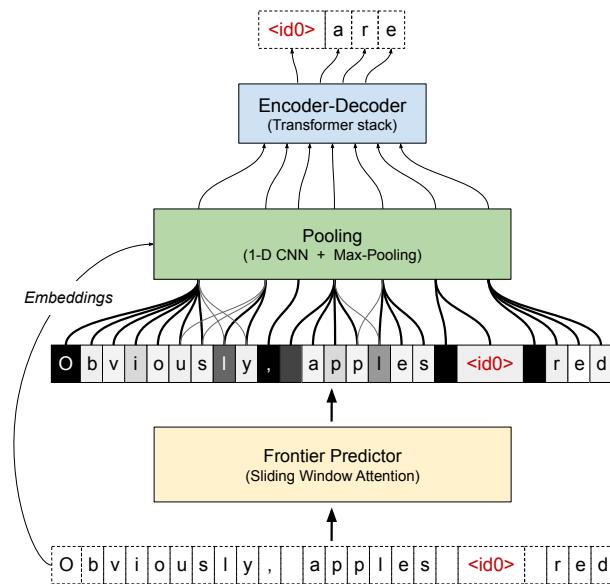


Figure 9.1: The differentiable tokenization scheme of MANTa-LM. Input bytes are first assigned a *separation probability* using a Sliding Window Attention Transformer. These probabilities are used to compute the contribution of each byte embedding in the pooled representations of the *blocks*. The block embeddings are fed to the Encoder-Decoding layers which predict the masked bytes. All the components are optimized with the LM objective.

To overcome the inefficiency of character-level language modeling, *tokenization-free* models (Clark et al., 2022a; Tay et al., 2021) compress sequences using specialized modules that rely on a static segmentation strategy (see Section 2.4.4).

We argue that learning a subword-level neural tokenization scheme together with input representations in an end-to-end fashion is beneficial for language modeling. In this work, we introduce MANTa, a gradient-based tokenizer and embedding module. It can easily be plugged-in to replace the classical combination of fixed tokenizers and trainable subword embedding matrices existing in most encoder-decoder models, without any increase in the total number of trainable parameters. We also introduce MANTa-LM, a Transformer encoder-decoder that incorporates MANTa and that is trained end-to-end. By learning a soft, adaptive segmentation of input sequences jointly with the LM pre-training objective, MANTa-LM produces byte-based representations with sequence lengths similar to those produced by static subword tokenizers. Additionally, by propagating gradients through our soft segmentation module during fine-tuning as well, we are able to adapt the segmentation to new domains, removing the limitations imposed by static subword tokenization.

Moreover, we show that MANTa-LM is robust to noisy text data and able to adapt to new domains while being significantly faster than byte-level models. Interestingly, MANTa learns a simple but explainable segmentation using only the LM objective while effectively reducing the length of byte sequences.

In summary, the contributions of this paper are the following:

- We introduce MANTa, a gradient-based tokenization and pooling module that can learn jointly with an encoder-decoder LM;
- We train MANTa-LM on English data and we evaluate its robustness to synthetic and natural variation and its ability to adapt to new domains compared to byte-level models.

## 9.1 MANTA

### 9.1.1 DIFFERENTIABLE TOKENIZATION

Our main contribution is the introduction of an end-to-end differentiable tokenization architecture that consists in softly aggregating input bytes into what we refer to as *blocks*. As an analogy with hard tokenization schemes, blocks can be compared to tokens with smooth borders.

We decompose the tokenization process into several differentiable operations, ensuring that our model can be trained end-to-end. Our approach consists in predicting a segmentation, and then combining byte embeddings according to this segmentation. MANTa can be divided in three different parts:

- Predicting block frontiers using a parameterized layer to assign a probability  $p_{F_t}$  to each input byte<sup>1</sup>  $b_t$  of being a frontier;<sup>2</sup>
- Building a byte-block unnormalized joint distribution using the frontier probabilities  $(p_{F_t})_{t \in [1, L]}$  corresponding to a soft assignment from bytes to blocks;

---

<sup>1</sup>Throughout this chapter, we use the notation  $b_t$  instead of  $w_t$  to improve clarity, as tokens do not correspond to word or subword level strings but rather to bytes.

<sup>2</sup> $F$  in  $p_{F_t}$  stands for *Frontier*.

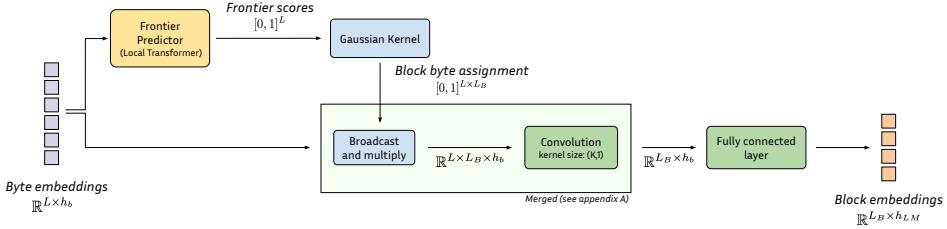


Figure 9.2: A detailed view of the MANTa module. We denote by  $h_b$  the dimension of the byte embeddings, by  $h_{LM}$  the dimension of the block embeddings that will be fed to the encoder-decoder model,  $L$  the length of the input sequence and  $L_B$  the length of the block sequence. We omit batch sizes for simplicity.

- Pooling byte representations for each block  $B_i$  weighted by the probability of each byte to belong in the current block  $P(b_t \in B_i)$ .

This process results in a sequence of embeddings that can be given directly to the encoder-decoder model. We provide an overview of the entire model in Figure 9.1, and we summarize the process of mapping byte embeddings to block embeddings in Figure 9.2.

### 9.1.2 PREDICTING SUBWORD FRONTIERS

Our frontier predictor consists in a parameterized module mapping each byte  $b_t$  to the probability of being a block frontier  $p_{F_t}$ . In a first part, we embed each byte  $b_t$  to an embedding  $e_{b_t}$ . Working with bytes instead of characters allows modeling a larger array of symbols while having very small embedding matrices with  $256 \cdot d_m$  parameters, where  $d_m$  is the hidden dimension of the model. Since the input sequences fed to the frontier predictor may be particularly long, we use a Transformer with sliding window attention (Beltagy et al., 2020). This layer achieves a linear complexity with respect to sequence length by computing attention using only a local context. This reduced context forces the model to focus on local surface features rather than long-range dependencies which may be hard to model at the byte level.

We make the assumption that long-range dependencies are not relevant for segmentation and that this reduced context window should not harm the quality of the tokenization.

### 9.1.3 MODELING THE BYTE-BLOCK ASSIGNMENT

Once the frontier probabilities  $(p_{F_t})_{t \in [1, L]}$  are predicted for the whole sequence, we use them to model an assignment between bytes and block slots. Each byte is given a probability distribution over the available block slots, and the expected block position of a byte in the block sequence increases along the byte sequence (i.e. the next byte is always more likely to be assigned to the next block).

Let us introduce  $(B, b_t)$ , the slot random variables for each byte  $b_t$ , describing the position of the block containing  $b_t$  in the block sequence. In other words, the event  $(B = i, b_t)$  describes the fact that the  $t$ -th byte belongs in the  $i$ -th block. These variables can only take values in  $[1, L]$ , as there cannot be more blocks than there are bytes. We can model the  $(B, b_t)$  as a *cumulative sum*

of the random variables  $F_t$ : the position of the block in which a byte belongs is exactly the number of frontier bytes before this one.

Since  $F_t \sim \mathcal{B}(p_{F_t})$ , we can model the block index variables  $B$  depending on the index of the bytes  $b$  using the Poisson Binomial distribution  $\mathcal{PB}$  which models the cumulative sum of Bernoulli variables:  $(B, b_t) \sim \mathcal{PB}((p_{F_k})_{k \leq t})$ . There exists no closed form for this distribution's mass function, but some fast methods have been developed to avoid exploring the whole event tree (Biscarri et al., 2018; Zhang et al., 2017). However, to reduce computational cost, we use a truncated Gaussian kernel  $G$  with the same mean and variance to approximate the  $(B, b_t)$  probability mass function:

$$\forall i \in [1, L_B], P(B = i, b_t) \simeq P_{i,t} \triangleq \frac{1}{Z} G_{\mu_t, \sigma_t}(i)$$

where  $Z = \sum_{1 \leq i \leq L_B} G_{\mu_t, \sigma_t}(i)$  is a normalization term, and:

$$\begin{cases} L_B = \min(L, (\mu_L + 3\sigma_L)) \\ \mu_t = \sum_{k=1}^t p_{F_k} \\ \sigma_t = \sqrt{\sum_{k=1}^t p_{F_k}(1 - p_{F_k})} \end{cases} \quad (9.1)$$

We denote by  $P_{i,t}$  the approximation of the probability of membership of the byte  $t$  to block  $i$ . We display an example of this map at different steps during training in Figure 9.3. We truncate the block sequences after  $(\mu_L + 3\sigma_L)$  since all the probabilities beyond this position are negligible.

#### 9.1.4 POOLING BLOCK EMBEDDINGS

At this point in the forward pass, we have estimated the position of the block in which each input byte belongs, along with the block sequence maximum plausible length  $L_B$ . In order to provide block embeddings to the LM, we now focus on the contribution of each byte to the block given by the block-byte assignment map. For each block position  $i \in [1, L_B]$ , this map actually provides an unnormalized contribution  $(P_{i,t})_{t \in [1, L]}$  of each byte in this block. We can then use the byte embeddings  $e_b$  from the frontier predictor described in Section 9.1.2 and, for the  $i$ -th block, build a block embedding where each byte  $b_t$  contributes based on its probability of being in this block  $P_{i,t}$ .

To build  $e_i^B$ , the embedding of block  $B_i$  in  $i$ -th position, we first compute the weighted byte embeddings  $(P_{i,t} \cdot e_t^b)_{t \in [1, L]} \in \mathbb{R}^{d_b}$ , with  $d_b$  the hidden size of the byte embeddings. To make the block embeddings aware of the ordering of the bytes (e.g. so that *ape* and *pea* can have different representations), we proceed to a depthwise 1-D convolution along the dimension of the bytes after weighting. This convolution also improves the expressiveness of the block embeddings.

Applying the 1-D convolution requires computing and storing  $\mathcal{O}(L_B \times L \times d_b)$  parameters since we apply the 1D-convolution on every row of the weighted embedding map  $P(e^b)^T$ . Therefore, this operation may be particularly costly, especially if the frontier predictor outputs a high number of blocks. However, we can use the fact that the weighted embedding map has a special form to reduce the memory load when computing the convolution. Let  $K$  be the convolution kernel size,

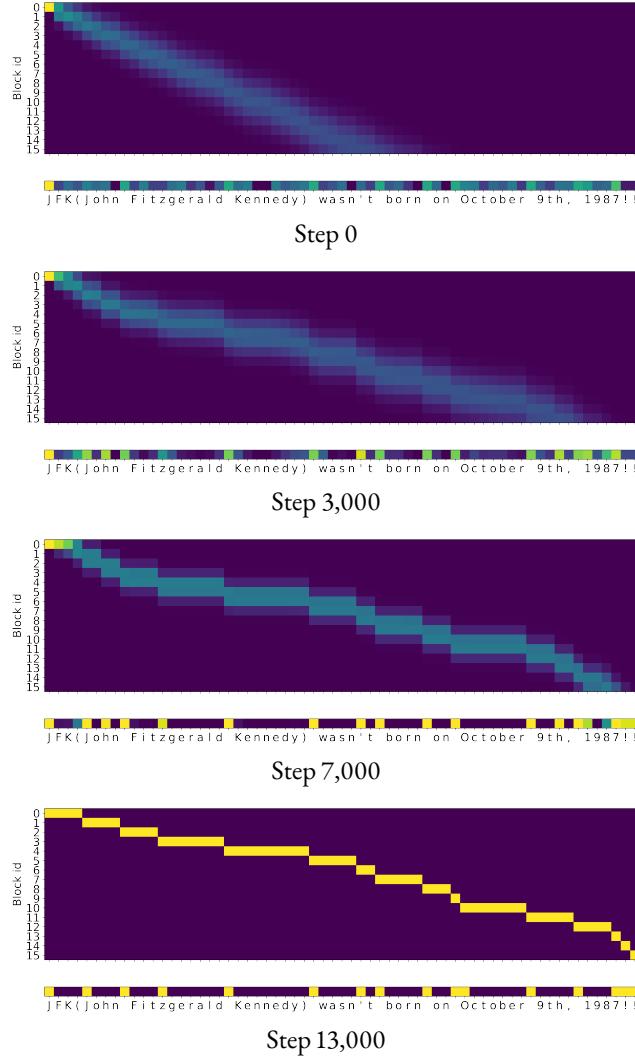


Figure 9.3: The block-byte assignment  $P$  during the first pre-training steps. MANTa learns to downsample input sequences so that no information is lost through truncation, but also converges towards a sharp segmentation.

$(C_j)_{j \in [1, K]} \in \mathbb{R}^{K \times d_b}$  the convolution filters and “.” denote the element-wise product. Then, omitting padding and biases :

$$\begin{aligned} e_i^B &= \max_{t \in [1, L]} \sum_{j=1}^K C_j \cdot (P_{i,t+j} \cdot e_{i+j}^b) \\ &= \max_{t \in [1, L]} \sum_{j=1}^K P_{i,t+j} \cdot (C_j \cdot e_{i+j}^b) \end{aligned}$$

Model	$ \theta $	MNLI	QNLI	MRPC	SST-2	QQP	STSB	COLA	AVG
T5 <sub>Small</sub>	60M	<b>79.7/79.7</b>	<b>85.7</b>	80.2/86.2	89.0	<b>90.2/86.6</b>	80.0	30.3	76.6
MANTa-LM <sub>Small</sub> (ours)	57M	79.2/78.6	84.5	<b>82.3/87.2</b>	<b>89.6</b>	89.9/ <b>86.5</b>	<b>81.4</b>	<b>32.0</b>	<b>77.1</b>

Table 9.1: Results on dev sets for the GLUE benchmark for small models following our pre-training procedure.

Notice how the product between the convolution filters and the byte embeddings  $C_j \cdot e_{t+j}^b \in \mathbb{R}^{d_b}$  does not depend on the block anymore. We cache this computation, storing  $\mathcal{O}(K \times L \times d_b)$  parameters and only later apply the convolution per block by summing these products with the block-byte membership map  $P$ . Caching greatly lowers the speed and memory requirements of MANTa, allowing to save  $L_B - 1$  element-wise products.<sup>3</sup>  $K$  is usually small, so the products can be stored easily.

We finally apply a max-pooling operation on the contextualized weighted byte embeddings for each block. This yields one embedding per block, with the same dimension as the byte embeddings. We use a linear layer to map the block embeddings to the right input dimension for the encoder-decoder model, i.e. its hidden size.

The final step consists in truncating the block embedding sequence to a fixed length  $\hat{L} = \min(L_B, L/K)$  with  $K \in \mathbb{N}^*$  a fixed *truncation factor*. This simple heuristic ensures that all sequences fed to the encoder-decoder have a length at least  $K$  times shorter than the input byte sequence length. We choose  $K = 4$  throughout the paper which is in average the number of bytes in an English BPE token. Most importantly, this truncation incentivizes the frontier predictor to produce sufficiently long blocks. We discuss the influence of this mechanism in more depth in Section 9.4.1.

### 9.1.5 MODEL TRAINING

We obtain from the differentiable tokenizer and pooling module a sequence of block embeddings that can be used exactly like subword embeddings. Thus, we use an encoder-decoder architecture identical to T5 (Raffel et al., 2020b). Nevertheless, since we do not have a fixed subword vocabulary, our decoder operates at the byte level similarly to ByT5 (Xue et al., 2022b).

### 9.1.6 PRE-TRAINING DETAILS

**OBJECTIVE** Our objective is identical to the one used in ByT5. We mask 15% of bytes randomly and choose a number of spans such that each has an average length of 20 bytes. Each span is then replaced by an `<extra_id_i>` token with  $i$  identifying the order of the span in the sequence. On the decoder side, the model has to predict in an autoregressive way the span identifier and the masked bytes.

**DATA** We pre-train our model on English text data using C4 (Raffel et al., 2020b), a large corpus scraped from the Internet. This corpus is particularly suited to our pre-training due to its diversity in terms of content and linguistic variations. In addition, it enables a better comparison with other

---

<sup>3</sup>This caching would be exactly similar if the convolution was not depthwise.

Model	$ \theta $	MNLI	QNLI	MRPC	SST-2	QQP	STSB	COLA	AVG
BERT <sup>†</sup> <sub>Base</sub>	110M	<b>84.4 / -</b>	88.4	86.7/-	<b>92.7</b>	-	-	-	-
T5 <sup>†</sup> <sub>Base</sub>	220M	84.2/ <b>84.6</b>	90.5	<b>88.9/92.1</b>	<b>92.7</b>	<b>91.6/88.7</b>	<b>88.0</b>	53.8	84.3
CharBERT <sup>§</sup> <sub>Base</sub>	125M	-	<b>91.7</b>	87.8/-	-	91/-	-	<b>59.1</b>	-
Byte-level T5 <sup>†</sup> <sub>Base</sub>	200M	82.5/82.7	88.7	87.3/91.0	91.6	90.9/87.7	84.3	45.1	81.5
Charformer <sup>†</sup> <sub>Base</sub>	203M	82.6/82.7	89.0	87.3/91.1	91.6	91.2/88.1	85.3	42.6	81.4
MANTa-LM <sub>Base</sub> (ours)	200M	77.5/78.8	88.2	82.4/88.2	91.3	90.8/87.7	79.2	51.0	80.3

Table 9.2: Results on dev sets for the GLUE benchmark.  $\dagger$  indicates results obtained by [Tay et al. \(2021\)](#), which are very similar to our models in terms of compute, but use a smaller batch size which may enhance their performance.  $\S$  indicates results obtained by [Ma et al. \(2020\)](#). The top section concerns model trained using a subword tokenizer.

tokenizer-free models trained using it such as Charformer. Since this dataset is not available publicly, we use the English split of the mC4 distributed by AllenAI. We filter long documents containing more than  $2^{15}$  bytes, which is a simple proxy to remove important quantities of unwanted code data.

**HYPERPARAMETERS** We pre-train two versions of our model: MANTa-LM<sub>Small</sub> and MANTa-LM<sub>Base</sub>. Each of them stacks a MANTa<sub>Small</sub> (resp. MANTa<sub>Base</sub>) tokenizer and embedding module and a T5<sub>Small</sub> (resp. T5<sub>Base</sub>) encoder-decoder model stripped of its tokenizer and subword embedding matrix. Details about MANTa hyperparameters can be found in [Appendix D](#).

Following T5 and ByT5, we use the Adafactor optimizer with a learning rate of  $10^{-2}$  for the encoder-decoder model, parameter scaling for the whole system and no weight decay. However, to maintain stability of our differentiable tokenizer, we use a learning rate of  $10^{-3}$  for the parameters of the byte embeddings, the frontier predictor, and the pooling module. We also use a triangular learning rate schedule with 1000 (resp. 5000) warm-up steps for batch size 1024 (resp. 64).

**TRAINING** We train T5<sub>Small</sub>, MANTa-LM<sub>Small</sub>, and MANTa-LM<sub>Base</sub> for 65k steps with a batch size of 1024. Sequence lengths are respectively 1024 for *Small* models and 2048 for the *Base* model. Thus, the models are trained on roughly the same amount of bytes as in [Tay et al. \(2021\)](#), where a batch size of 64 is used for 1M steps.

We also train a ByT5<sub>Small</sub> model on the same data, using a batch size of 64 and a sequence length of 1024. We consider the “Scaled” architecture which provides the encoder with more layers than the decoder ([Xue et al., 2022b](#)). To avoid prohibitive computation costs and ensure fairness in terms of available resources between models, we limit its training time to the one of MANTa-LM<sub>Small</sub>. Hence, our ByT5<sub>Small</sub> is only trained for 200k steps.

Model	Accuracy
BERT <sup>‡</sup> <sub>Base</sub>	77.7
CharacterBERT <sup>‡</sup> <sub>Base</sub>	77.9
T5 <sub>Small</sub>	75.3
MANTa-LM <sub>Small</sub> (ours)	75.6

Table 9.3: Results on MedNLI. ‡ indicates results from [El Boukkouri et al. \(2020\)](#), who use a different pre-training corpus than C4. All other results are from models trained with our codebase.

## 9.2 EXPERIMENTS AND RESULTS

### 9.2.1 EVALUATION ON GLUE

To ensure that our model is competitive with existing language models exploiting subword tokenization algorithms, we evaluate it on several English datasets and compare it with other baseline models.

**SETUP** We use GLUE ([Wang et al., 2018](#)), a Natural Language Understanding benchmark consisting of 7 tasks, to evaluate our model. Similarly to T5, we cast the classification tasks as generation tasks where the model has to predict autoregressively the bytes forming the answer.

We compare our model to an encoder-decoder model with subword tokenization (pre-trained with the same denoising objective as T5) and a fully byte-level encoder-decoder, similar to ByT5. We compare *Small* models with our pre-trained versions, and *Base* models with results mentioned in [Tay et al. \(2021\)](#). We report the number of parameters given in [Tay et al. \(2021\)](#) for Byte-level T5<sub>Base</sub>, and gather from its low value that their implementation corresponds to a T5<sub>Base</sub> architecture trained on byte-level inputs.

**RESULTS** Results can be found on Tables 9.1 and 9.2. Overall, MANTa-LM exhibits a performance slightly below Charformer but stays within a small margin on average (1.1 points below). Nonetheless, the main objective of our method is to balance decent performance with robustness and speed which we show in the following sections.

### 9.2.2 ROBUSTNESS TO DOMAIN CHANGE

Static subword tokenizers tend to show important limitations when used with texts originating from a domain unseen during training. For instance, [El Boukkouri et al. \(2020\)](#) show that tokenizing medical texts with a tokenizer trained on Wikipedia data often results in an over-segmentation of technical terms which in turn affects the downstream performance. By removing this static bottleneck in MANTa-LM, we hope that it should be able to adapt more easily to new domains. To test this hypothesis, we finetune it on a medical Natural Language Inference dataset.

**SETUP** We finetune MANTa-LM on MEDNLI ([Romanov and Shivade, 2018](#)), a dataset consisting of 14,049 sentence pairs extracted from clinical notes. We follow the same finetuning setup than for the GLUE Benchmark i.e. use the same batch size and learning rate. We compare our

Model	Accuracy
$T5_{Base}^{\dagger}$	91.5
$Charformer_{Base}^{\dagger}$	92.7
MANTa-LM <sub>Base</sub> (ours)	<b>93.2</b>

Table 9.4: Results on the ToxicComments dataset. Results indicated by  $\dagger$  are from [Tay et al. \(2021\)](#).

results to the ones obtained by [El Boukkouri et al. \(2020\)](#) with models pretrained on the general domain.

**RESULTS** We present our results on [Table 9.3](#). Although we notice a significant drop in performance compared to the encoder models trained by [El Boukkouri et al. \(2020\)](#), we believe this drop may be due to the different pretraining data used—CharacterBERT uses splits of Wikipedia, which may be helpful to learn some technical terms related to the clinical domain—, and the different model sizes—CharacterBERT uses all of its parameters to encode example, while we keep half of the parameters in the decoder. Nonetheless, we note that MANTa-LM reaches a better performance than its subword tokenization counterpart T5.

### 9.2.3 ROBUSTNESS TO NOISY DATA

Although LMs may learn complex patterns even from noisy input texts, this ability is conditioned by how the tokenizer segments character sequences. Since MANTa is not static and can be finetuned on non-standard data, we expect it should be able to learn to be more robust to variation/noise compared to a subword tokenizer paired with a LM. To evaluate this hypothesis, we study how MANTa-LM behaves on both naturally occurring text variation and multiple levels of synthetic noise.

### 9.2.4 NATURALLY OCCURRING NOISE

**SETUP** Similarly to [Tay et al. \(2021\)](#), we test our model on a toxicity detection task constructed with user generated data. We use the ToxicComments dataset ([Wulczyn et al., 2017](#)) which contains 223,549 sentences annotated with a binary label indicating whether each sentence can be classified as toxic or not. We also use the same finetuning setup here as the one used for evaluating on the GLUE benchmark.

**RESULTS** We present our results in [Table 9.4](#) and compare them to the ones reported in [Tay et al. \(2021\)](#). As expected, noisy user generated data is particularly harmful for models using subword tokenization. On the other hand, constructing sentence representations with byte-level information helps and our model is more accurate than Charformer. This gain may be due to a better segmentation of specific terms encountered in the data.

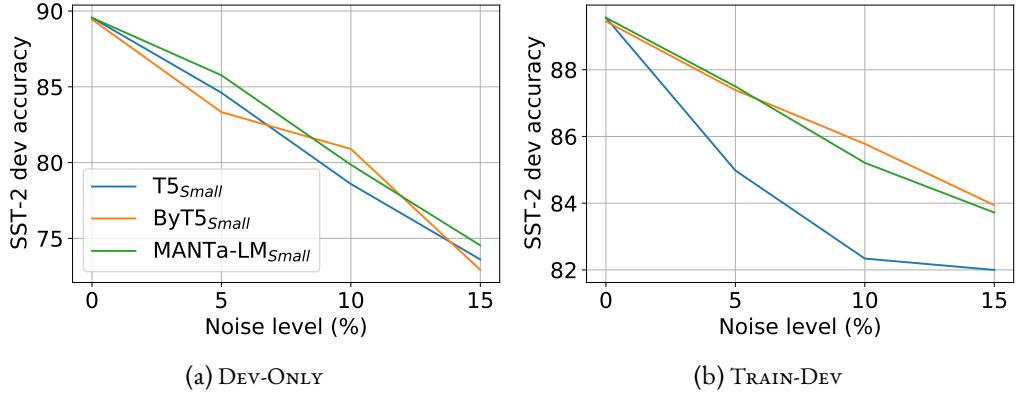


Figure 9.4: Best accuracy on the SST-2 development set as the noise level increases. The TRAIN-DEV setting corresponds to models finetuned on noisy data while models in the DEV-ONLY setting have been finetuned on clean data.

### 9.2.5 SYNTHETIC NOISE

**SETUP** We also compare T5 and ByT5 with our approach when facing different levels of noise. This study pictures how these models react to unseen noise at evaluation time (DEV-ONLY setup) and how they adapt to a given noise via fine-tuning (TRAIN-DEV setup). We apply synthetic noise at different levels  $\tau \in \{0.05, 0.10, 0.15\}$  by picking randomly  $\tau \times L$  positions in the byte sequences and equiprobably deleting, replacing or inserting bytes at these positions.

**RESULTS** The results can be found in Figure 9.4. We found that models performed similarly for the different noise levels in the DEV-ONLY setting. On the contrary, in the TRAIN-DEV setting, MANTa-LM can be finetuned as well as ByT5 for all levels of noise, while the performance of T5 quickly degrades.

## 9.3 TRAINING SPEEDUPS

In terms of speed, we compare our model to MANTa-LM<sub>Small</sub> to T5<sub>Small</sub> counterparts: one that is trained at the classical subword-level, and one trained at byte-level, hence using sequences that are roughly 4 times longer. We also report the speed of the larger ByT5<sub>Small</sub> architecture as described in Xue et al. (2022b). The results are presented in Table 9.5.

MANTa-LM is approximately 4 times faster than Byte-level T5<sub>Small</sub>, and 5 times faster than ByT5<sub>Small</sub>, which can be explained by the reduced sequence length we use in the encoder-decoder model. MANTa-LM is only 2.3 times slower than T5<sub>Small</sub> which furthermore benefits from already tokenized sequences at training time.

Model	$ \theta $	Seconds/step
Byte-level T5 <sub>Small</sub>	57M	9.06 ( $\times 8.0$ )
MANTa-LM <sub>Small</sub>	57M	2.61 ( $\times 2.3$ )
T5 <sub>Small</sub>	60M	1.13 ( $\times 1$ )

Table 9.5: Comparison of training speeds. All the experiments were run on 16 NVIDIA V100 GPUs using a batch size of 1024 and a sequence length of 1024 bytes or 256 tokens

<b>Original</b>	Oh, it's me vandalising?xD See here. Greetings,
<b>MANTa</b>	Oh,  it 's  me  vandalising?xD  See  here.  Greetings ,
<b>T5 tokenizer</b>	Oh,  it 's  me  van dal i sing?xD  See  here.   Greetings ,
<b>Original</b>	The patient was started on Levophed at 0.01mcg/kg/min.
<b>MANTa</b>	The  patient  was  started  on  Levophed  at  0.01mcg/kg/min .
<b>T5 tokenizer</b>	The  patient  was  started  on  Le v o p h e d  at  0.01mcg/kg/min .

Table 9.6: Examples of segmentations produced by our module (pre-trained only) and by T5’s BPE tokenizer. The sentences are samples from ToxicComments and MEDNLI.

## 9.4 DISCUSSION

### 9.4.1 TRUNCATING EMBEDDING SEQUENCES

Once we obtain block embeddings, the final step in MANTa consists in truncating sequences to a length 4 times smaller than the original byte sequence, as described in Section 9.1.4. This is essential to make MANTa-LM work.

First, it increases the control over the encoder-decoder’s computation cost. Without this bottleneck, the Transformer can receive sequences varying from a single block containing the whole sequence ( $L_B = 1$ ) to one block per byte in the sequence ( $L_B = L$ ). In the latter case, which mimics ByT5’s input segmentation, the computation becomes extremely slow due to the quadratic cost of the attention with respect to the sequence length. Using the bottleneck ensures that we can control the worst case complexity of the encoder Transformer and keep it similar to that of a subword-based encoder model.

Second, it serves as a kind of regularization for the block segmentations. We noted that training our module without the bottleneck often led to block sequences as long as byte sequences ( $L_B = L$ ). This may be due to the beginning of training where having very local information helps - for instance bytes to the left and right of masked spans. However, such a segmentation degrades the model speed and performance later in training. Truncating the sequence forces the model to construct larger blocks in order to “fit” all the information from the input sequence.

### 9.4.2 LEARNT BLOCK SEGMENTATION

Segmentation examples can be found in Table 9.6. For each byte, we retrieve the expected block position produced by MANTa and approximate it with the closest integer to mimic hard tokenization. We found that MANTa is not keen to produce subword level segmentations. Most of

the key symbols for word separation have been identified as block delimiters during pre-training. As expected, MANTa is less prone to over-segmentation of unknown words like named entities. We also found that a trained MANTa produced spiked separation probabilities, meaning that it converged towards a “hard” segmentation. This can also be observed by monitoring the value  $\min(p_{F_t}, 1 - p_{F_t})$  which always converges towards values of magnitude  $10^{-5}$ .

#### 9.4.3 GRADIENT-BASED SEGMENTATION

We employ a radically different downsampling approach compared to other gradient-based tokenization methods such as CANINE (Clark et al., 2022a) or Charformer (Tay et al., 2021). While CANINE downsamples sequences using a fixed rate after byte contextualization and Charformer’s GBST (Gradient Based Subword Tokenizer) pools representations created using various downsampling rates, MANTa only applies downsampling right before the LM to limit the length of block sequences. Hence, our model is able to build word-level representations of *arbitrary length* as long as it divides the whole byte sequence length by a fixed factor.

We also argue that our method yields more explainable pooled representations as the segmentation can be explicitly derived from the outputs of MANTa. Indeed, contrary to CANINE and Charformer, MANTa disentangles the segmentation of blocks from their representations, allowing to study each part separately.

#### 9.4.4 MAIN HYPERPARAMETERS

We discuss here some of the major hyperparameters of our method. Constrained by limited computational resources, we were unable to assess their exact importance on MANTa’s performance. We try to give some intuitions on their influence.

**FRONTIER PREDICTOR** We used a small Transformer network with sliding window attention for this module. A much larger network would be slower and may not bring significant improvements to the overall performance of the model, since it is only used for predicting the block byte assignment but does not “expand” the overall expressivity of the model.

**CONVOLUTION KERNEL APPLIED ON BYTE EMBEDDINGS** This kernel adds positional information to the byte embeddings and expressivity when constructing the block embeddings. Using a larger kernel or a concatenation of kernels might help for better block representations. However, our experiments did not show any significant difference in the pretraining performance.

**BLOCK EMBEDDING SEQUENCE TRUNCATION FACTOR** Trimming block sequences was instrumental to produce meaningful input segmentations and blocks containing more than a single byte. We settled for a factor of 4 since other values led to minor degradations early in training. This factor roughly corresponds to the average number of bytes in a subword created by an English tokenizer.

We believe that a more thorough hyperparameter search could improve the performance of our model. We leave this for future work due to computational limitations.

## CONCLUSION

In this work, we present MANTa, a fully differentiable module that learns to segment input byte sequences into blocks of arbitrary lengths, and constructs a robust representation for these blocks. We train this module jointly with an encoder-decoder LM on a span denoising objective to obtain MANTa-LM. We then show that MANTa-LM is more robust when applied to noisy or out-of-domain data than models using static subword tokenizers. At the same time, it performs on par with fully byte-level models on these setups while operating with a much reduced computational cost.

Beyond the noisy and out-of-domain settings, we believe that our approach could lead to interesting results for a number of languages, especially those whose writing system do not use whitespace separators, such as Chinese.

Finally, tokenizers are hypothesized to be an important limiting factor when segmenting multilingual data (Rust et al., 2021). We believe MANTa could be used in the multilingual setting to ensure a more balanced segmentation between languages.

## LIMITATIONS

Although MANTa can help alleviate some of the inherent issues accompanying subword tokenizers, it also suffers some flaws that we believe could be addressed in future work.

Contrary to encoder-decoder models that can decode long sequences efficiently, our model has to decode sequences byte-per-byte (similarly to Clark et al. (2022a); Xue et al. (2022b); Tay et al. (2021)) which adds an important computational overhead at generation time. Previous works have attempted to reduce this computational cost by decreasing the size of the decoder layers compared to the encoder (Xue et al., 2022b) or by projecting embeddings to a smaller latent space (Jaegle et al., 2021) for the decoding.

Finally, we presented in this work a proof of concept of adaptive segmentation algorithms on relatively small models, ranging from 50M to 200M parameters. Although we hypothesize that our model would scale relatively well since it keeps most of the encoder-decoder architecture untouched, this hypothesis should be tested in a future work.

The MANTa module allows us to design efficient byte-level language models. Nevertheless, as discussed in Chapter 7, we actually find that, on every layer including the last, the anisotropy level of MANTa-LM is actually comparable with the T5 subword-level model. This demonstrates that subword frequency-related issues are not the only cause of anisotropy even on the last layer of Transformer models.

For instance, byte-level modeling does not address the sparsity of attention maps which we find to be correlated with anisotropy in Chapter 7. In Chapter 10, we provide insights towards addressing the sparsity of self-attention, by using pooling strategies similar to the MANTa layer at the self-attention level.



# 10

## TOWARDS GRADIENT-BASED DYNAMIC KV-CACHE COMPRESSION

*This chapter describes work that is still in progress, and that was done during an academic visit with Pr Edoardo Ponti at the University of Edinburgh.*

Chapter 7 shows that studying the representation degeneration phenomenon at self-attention level sheds light on the sparse nature of most attention maps. It depicts a connection between anisotropy and sparsity in the attention maps of language models, as the identified common drift direction is used to encode the selective process of a specific head.

This observation raises questions about the efficiency of the attention operation, as most of the attention values are actually not relevant in the production of the output of the self-attention layer. It also questions the effect of such sparsity requirements on the upstream latent spaces of these language models, and on the possible harmfulness of the distortions that affect them.

As mentioned in Section 2.4.5, several works have explored ways to reduce the sparsity of self-attention maps, by filtering the  $(i, j)$  index pairs where attention should be computed during training and inference, in order to reduce the time and memory complexity of self-attention. Another line-of-work proposes solutions that focus on post-training efficiency improvement and

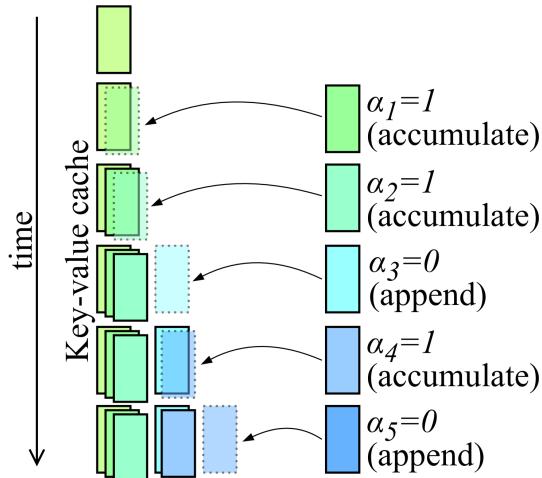


Figure 10.1: Schema of the Dynamic Memory Compression technique from (Nawrot et al., 2024). A merge decision is made at every step, which either extends the KV cache with a new slot or adds the current KV representations into the current slot. During training, the Gumbel reparameterization trick allows to mimic sampling the merge decision in a differentiable manner.

build heuristics that minimize the impact of removing  $K^h$  and  $V^h$  representations on performance, earning these methods the name of Key-Value (KV) cache compression.

Limitations of most KV cache compression schemes include that the heuristics are hand-crafted and may thus be suboptimal, but also that these methods are non-differentiable and cannot be used as efficient attention methods to train models from scratch.

[Nawrot et al. \(2024\)](#) overcome this differentiability limitation by using a Gumbel-sigmoid activation on top of the first coefficients of  $Q^h$  and  $K^h$  to predict a decision to merge into the current cache slot or to create a new cache slot. Thanks to this technique, the authors are able to continue the training of language models with an auxiliary objective that measures the compression ratio and optimize the KV cache size while retaining the language modeling performance.

The Gumbel reparameterization trick ([Jang et al., 2017](#)) allows to mimic multinomial sampling in a differentiable manner, which is useful in models that require to emulate such sampling during training. Nevertheless, these methods add stochastic noise to their input, which may alter the expressiveness of gradients. Moreover, [Nawrot et al. \(2024\)](#) only consider contiguous compression, where only representations corresponding to contiguous positions can be merged, which restricts the expressiveness of the compression scheme.

In this chapter, we provide ideas and experiments towards extending the dynamic KV cache compression method of [Nawrot et al. \(2024\)](#). Notably, we provide a mathematical framework for neural compressive methods in the causal setting without sampling via methods based on the Gumbel reparameterization trick. We test our framework by training causal models with compressive attention from scratch, and by continue-training models for KV cache compression.

## 10.1 MATHEMATICAL FRAMEWORK

We propose to lay the foundations for an efficient causal compressive attention module, based on basic matrix operations. For the sake of simplicity, we consider a basic single-head attention setup with input representations  $Q, K, V \in \mathbb{R}^{L \times d_m}$ , and without output projection  $W_O$ .

We recall from [Section 2.3.4](#) the expression of the attention map  $A$  and of the attention output  $v$ :

$$\begin{cases} A = \sigma\left(\frac{QK^T}{\sqrt{d_h}} + \mathcal{M}\right) \\ v = AV \end{cases} \quad (10.1)$$

where  $\sigma$  is the softmax function applied to the last dimension,  $\mathcal{M}$  is the causal mask and  $d_h = d_m$  as there is only one head.

We view compression as a matrix multiplication based on a *compression mapping*  $M \in \mathbb{R}^{L_C \times L}$ , with  $L_C \leq L$ . Ideally, we want to build compressed views of  $K$  and  $V$  in  $\mathbb{R}^{L_C \times d_m}$  by computing:

$$\begin{cases} K_C = MK \\ V_C = MV \end{cases}$$

and using  $K_C$  and  $V_C$  instead of  $K$  and  $V$  in the self-attention operations described in [Equation \(10.1\)](#).

This approach leads to an immediate complication with respect to causality, as the product  $QK_C^T$  allows for non-causal interactions, i.e. operations between features from  $Q_i$  and  $K_j$  where

$i > j$ . A similar problem occurs in the  $AV_C$  product. We propose to overcome this issue by first remarking that

$$QK_C^T = Q(MK)^T = (QK^T)M^T$$

and

$$AV_C = (AM)V$$

Hence, it is possible to cancel the influence of non-causal interactions before compression by nulling out the corresponding coordinates using a binary causal mask

$$\mathcal{M}_1 = \exp \mathcal{M}$$

which is basically a lower triangular matrix filled with 1. For instance, the query-key product can be rewritten as  $(QK^T \odot \mathcal{M}_1)M^T$  where  $\odot$  is the element-wise product.

Although this trick solves causality issues, it also makes the causal mask  $\mathcal{M}$  inaccurate with respect to the softmax function  $\sigma$ . Indeed, in [Equation \(10.1\)](#),  $\mathcal{M}$  removes attention map indices that do not respect causality from the computation of the softmax. In the compressed sequence scenario, and especially when  $M$  is sparse, these indices can be found in positions  $(i, j)$  where the compression mapping only pools from positions after  $i$  into position  $j$ , i.e. where  $(\mathcal{M}_1 M^T)_{ij} = 0$ .

We can thus design a causal mask for softmax  $\mathcal{M}_C$  that takes compression into account:

$$\mathcal{M}_C = \log \mathbf{1}_{\mathcal{M}_1 M^T > 0}$$

We can now formulate a compressed version of causal self-attention:

$$\begin{cases} A = \sigma \left( \left( \frac{QK^T}{\sqrt{d_h}} \odot \mathcal{M}_1 \right) M^T + \mathcal{M}_C \right) \\ v = (AM \odot \mathcal{M}_1)V \end{cases} \quad (10.2)$$

To extend this framework to multi-head attention, the operation described in [Equation \(10.2\)](#) can be computed for all heads, each head potentially applying a different compression mapping.

## 10.2 PREDICTING COMPRESSION MAPPINGS

The interest of [Equation \(10.2\)](#) lies in the fact that *any* compression mapping  $M \in \mathbb{R}^{L_C \times L}$  can be used. This includes static compression mappings, parameterized matrices, or even the results of parameterized operations on  $Q$ ,  $K$ , and/or  $V$  representations.

In this chapter, we particularly focus on compression maps that are well-suited for KV cache compression, which restrains design choices to compression maps that are built in a causal way themselves, as the cached  $K_C$  and  $V_C$  representations can only be updated using the currently available representations at generation step  $t$ , which includes  $K_t$ ,  $V_t$  and  $K_C$  and  $V_C$  themselves.

Consequently, a natural choice is to adapt our work on the MANTa module in [Chapter 9](#) to this framework. We call  $\Psi$  the function that maps frontier decisions  $p_{F_t}$  to the Gaussian kernel used to approximate the Poisson-binomial distribution in [Section 9.1.3](#). As the intermediate representations of Transformer-based models are already contextualized and contain positional

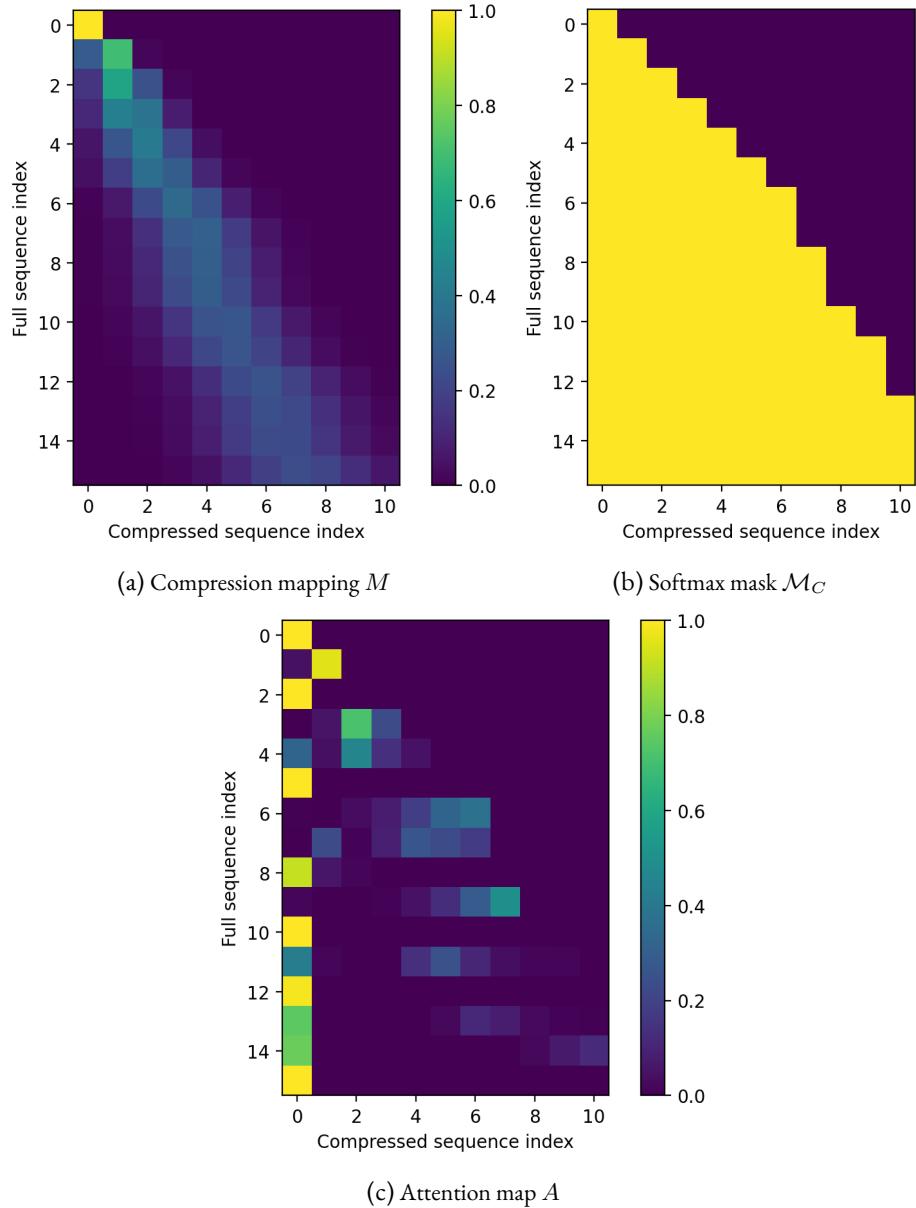


Figure 10.2: An example of compressed attention with random input representations and using the MANTa mapping to build  $M$ .

information, except for the first layer, we do not need a contextualizing module to obtain  $p_{F_t}$ , or a convolution to encode position when pooling representations.

Following Nawrot et al. (2024), we also implement a weighting mechanism that allows to discard representations altogether, and to weight contributions in the pooled representations.

Formally, in the single head scenario, we introduce two linear layers of weights  $W_F$  and  $W_\omega$  in  $\mathbb{R}^{3d_m \times 1}$  that are applied to  $\{QKV\} \in \mathbb{R}^{L \times 3d_m}$ , the concatenation of  $Q$ ,  $K$  and  $V$  in the feature dimension. The compression mapping is then computed as:

$$\begin{cases} p_F = \varsigma(\{QKV\}W_F + C_F) \\ \Omega = \varsigma(\{QKV\}W_\omega + C_\omega) \\ M = \Psi(p_F) \odot \Omega \end{cases} \quad (10.3)$$

where  $\varsigma$  is the sigmoid function, and  $C_F$  and  $C_\omega$  are constants that help initialize the model in a non-compressive way, i.e. where  $p_F \approx \mathbf{1}$  and  $\Omega \approx \mathbf{1}$  so that  $M \approx I_L$ .

As  $\{QKV\}$  is computed in a causal fashion,  $p_{F_t}$  and  $\Omega_t$  are not affected by future positions of the sequence, and will not change when generating further in the sequence. Hence, the first  $t$  columns of  $M$  are not affected by  $\{QKV\}_i$  when  $i > t$ . Moreover,  $\Psi$  can easily be computed in a memory-efficient way at inference, by storing the running average and variance for block position as  $t$  increases. This implies that the parallel computation of compressed self-attention in Transformer blocks can easily be adapted to an efficient KV caching system during inference.

Similarly to [Nawrot et al. \(2024\)](#), we train language models with an auxiliary loss that measures the average compression rate:

$$\mathcal{L}_C = \frac{1}{L} \sum_{t=1}^L p_{F_t}$$

The global loss of the model is then defined as:

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_C \mathcal{L}_C$$

## 10.3 PRELIMINARY EXPERIMENTS

In this section, we provide some of the first results we obtained with the method described in previous sections. This work still being in progress, we warn the reader that these results constitute proofs-of-concept rather than strong results that would allow conclusive statements about the method.

### 10.3.1 PRETRAINING

We train 70 million parameter models using the same architecture as its Pythia counterpart ([Biderman et al., 2023a](#)) on the CCNews ([Hamborg et al., 2017](#)) dataset. We use small batch sizes of 32 sequences of length 128 to accelerate training. We perform 100,000 steps, which represents 409M tokens. The weights are optimized using AdamW with no weight decay and a learning rate of  $10^{-3}$  for all parameters except  $W_F$  and  $W_\omega$  for which a learning rate of  $10^{-4}$  is used, accordingly with [Chapter 9](#). We also use a cosine scheduling with a 2000-step warm up phase. We set  $C_F = C_\omega = 5$  to make sure the model starts with a compression ratio close to 1. We observe that training throughput is not significantly affected by the compression operation, when using a pure PyTorch implementation of self-attention.

We explore values of  $\lambda_C$  in  $\{1 \cdot 10^{-2}, 2 \cdot 10^{-2}, 3 \cdot 10^{-2}\}$  leading to different final compression ratios, as pictured in Figure 10.3b.

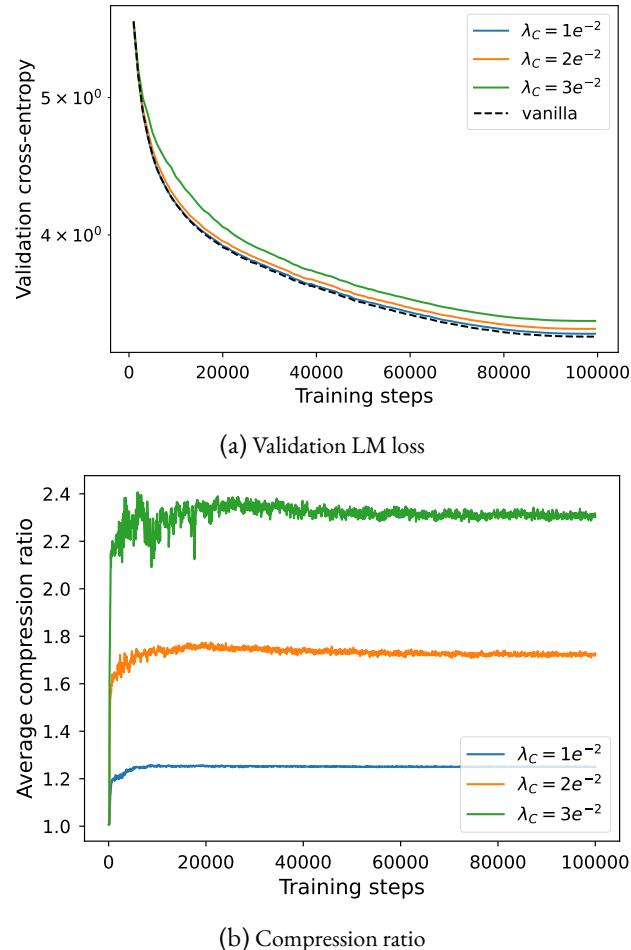


Figure 10.3: Training dynamics of our language models trained with compressed attention.

In Figure 10.4, we show examples of learned compression maps for the model trained with  $\lambda_C = 2 \cdot 10^{-2}$ .

Figure 10.4 depicts different compression schemes, with compression rates that increase as we move towards deeper layers. Overall, these experiments tend to demonstrate the potential of this approach, as it allows us to train models from scratch with compressed KV cache.

We leave our initial hypothesis on the impact of denser attention maps on the anisotropy of intermediate layers for future work, as proving it would require using larger models trained in regular hyperparameter setups, these small models being extremely anisotropic regardless of the compression behavior.

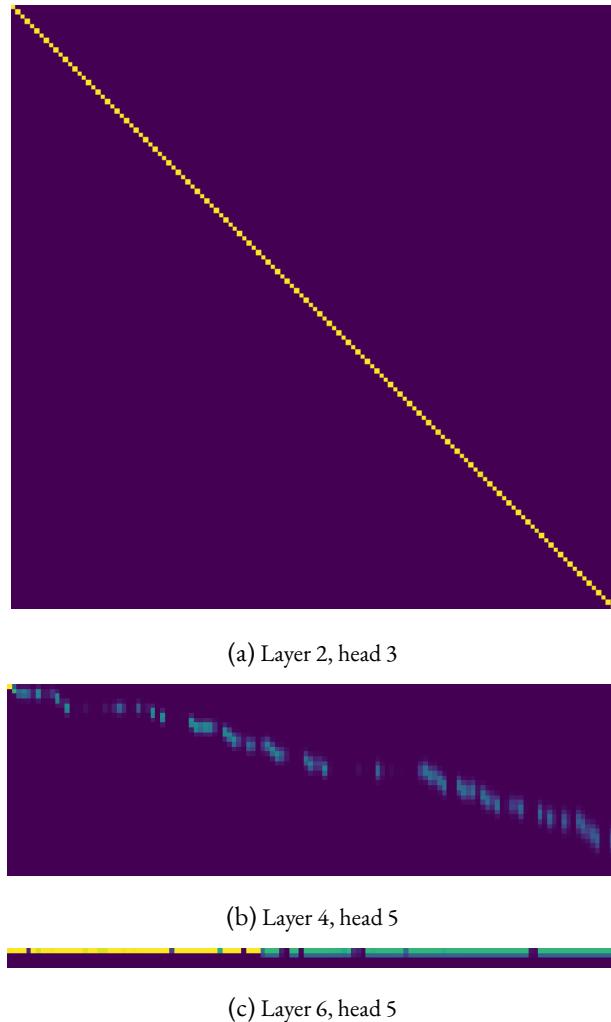


Figure 10.4: Compression maps  $M$  for the final checkpoint of the model trained with  $\lambda_C = 2 \cdot 10^{-2}$  (in transposed view)

### 10.3.2 APPLICATION TO KV CACHE COMPRESSION

We also apply our method to KV cache compression. To do so, we initialize our compressed attention architecture with the weights of a pretrained model of 160 million parameters from the Pythia suite, and choose sufficiently high values for  $C_F$  and  $C_\omega$  so that compression is minimal and the performance is retained. We perform an extensive hyperparameter search for this procedure, and report the results of this search in [Figure 10.5](#).

[Figure 10.5](#) illustrates the high sensitivity of this method to hyperparameter choices when applied to KV cache compression. Interestingly, we are able to reach a  $13\times$  compression ratio while retaining reasonable language modeling capabilities, and we also observe cases where the

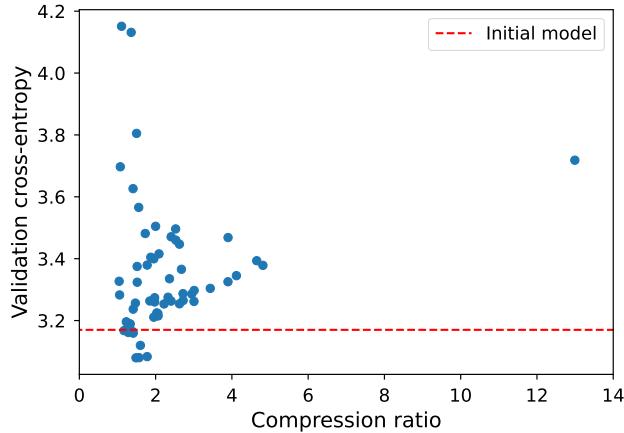


Figure 10.5: Results of our hyperparameter search on KV cache compression setups for a Pythia-160M. Each point represents a choice of hyperparameters.

continued training of the model on the CCNews data led to improved performance while still achieving non-negligible compression levels.

This exploration incentivizes future research on improving the stability of continued training when using MANTa-based compression mappings.

## CONCLUSION

In this chapter, we introduce a mathematical framework for fully differentiable compression of keys and values in the self-attention operation, which implicitly densifies the attention maps used by the models, and allows more memory-efficient modeling. We notably design a way to solve the causality issue that emerges when conceiving such models by using several masks that filter out the non-causal interactions between sequential features.

We proceed to explore a specific family of compression mappings based on the MANTa module presented in [Chapter 9](#). We apply our method to causal language model pretraining and demonstrate the potential of our method by achieving  $2\times$  compression while retaining similar performance levels. We also present initial results on the KV cache compression setup, opening up challenging questions for future work.

This preliminary study could also lead to other types of compression mappings, which could include mappings based on a basis of cosine-based vectors to model long-range non-contiguous pooling, negative weight contributions to the compressed sequence to mimic a discarding operation at a later generation step, or fixed-length compression with unconstrained weights.

This chapter paves the way for automatically learned dense attention maps through a novel mechanism for self-attention that allows to include a sequence-wise compression operation. Taking advantage of the sparsity of attention is a promising way towards more efficient language models.

As we discussed in [Chapter 7](#), the sparsity of self-attention can also induce distortions in the latent space of queries and keys. We hypothesize that a sufficiently expressive compression mapping function could mitigate such phenomenon and allows for easier selective interactions while minimizing the resulting distortions. We leave the exploration of this hypothesis to future work.



# 11 CONCLUSION

## 11.1 SUMMARY

Going back to our depiction of the current language modeling paradigm as a fundamentally behaviorist framework in [Chapter 1](#), we explored in this thesis one of the possible research direction that goes beyond pure behaviorism, namely the analysis of hidden state spaces from a qualitative viewpoint.

We first reviewed the state-of-the-art for language modeling techniques ([Chapter 2](#)), representation learning in NLP ([Chapter 3](#)), and representation analysis in NLP models ([Chapter 4](#)).

We then provided several qualitative representation evaluation scopes, that assess the expressivity and uniformity of representations in order to provide insights on the language models themselves.

In [Chapter 5](#), we evaluate biases of language models through the lens of geographical knowledge by extracting geospatial information from the representations of named entities corresponding to locations. We then remark that the biases of language models as measured by this metric increase as the model size grows. We show that these biases tend to amplify biases that are inherent to the training datasets, thus proving this phenomenon to be an instance of frequency-based issues in the representations of language models.

We proceed by studying the more general question of frequency-based distortions in language models in [Chapter 6](#), by extending previous works on the relation between token distributions and anisotropy. We explore a connection between the emergence of last-layer anisotropy, which is a form of representation degeneration, and an empirically observed performance saturation during training for some small language models. Moreover, we show that this representation degeneration is deeply connected with the softmax bottleneck, which can be summarized as a mismatch between the hidden dimension of the model and the intrinsic dimensionality of the target contextual log-probability space.

Nevertheless, this phenomenon is not sufficient to explain a similar anisotropy phenomenon measured in intermediate layers. In [Chapter 7](#), we analyze this degeneration in intermediate layers and show that it is not specific to subword-level language models. We discover that anisotropy is characteristic of the Transformer architecture, and facilitates the sparsity of self-attention maps.

Finally, we explore diverse techniques to mitigate representation degeneration in language models, by proposing changes to the training objective, the data granularity, and the self-attention operation.

[Chapter 8](#) explores an in-batch contrastive objective used to train masked or causal language models. This approach implicitly and empirically mitigates last-layer anisotropy, while also improving data and compute-efficiency and allowing for much larger token vocabularies. However, this objective is not well-suited for language generation and requires a fine-tuning step using the classical

## 11 Conclusion

cross-entropy loss. Hence, it does not entirely avoid the frequency-based distortions inherent to this objective.

Thus, we further extend the language modeling paradigm in [Chapter 9](#), by efficiently modeling text at byte-level, which should notably address the softmax bottleneck described in [Chapter 6](#). Although our approach leads to more robust models that perform better on noisy textual data, we observe that our method did not significantly affect the degenerated nature of the intermediate representations of language models.

We hypothesize that this degeneration phenomenon, linked to attention sparsity in [Chapter 7](#), can be mitigated by facilitating the densification of attention maps in Transformer-based models. In [Chapter 10](#), we lay a path towards gradient-based neural compression of keys and values representations, and perform initial experiments that prove the potential of our method.

Overall, we explore the use of model analysis tools as ways to identify limitations, and as a source of insights towards improving language models.

## 11.2 DISCUSSION

### 11.2.1 LIMITATIONS

Having discussed technical limitations individually for each chapter, we rather depict here more general limitations and potential objections to our overall approach.

First, we do not produce an extensive study of qualitative inner evaluation techniques for language models in this thesis. The research path that consists in making systematic observations about the inner workings of language models, and in then leveraging these observations to improve their performance is relatively underexplored, although it could be argued that it led to impactful works in recent years ([Hu et al., 2022](#); [Xiao et al., 2024](#)). This thesis rather focuses on a specific type of qualitative measurement, representation analysis, through the lens of inherent bias and latent uniformity. As such, our work only represents one example of research trajectory that explores alternatives to purely behaviorist strategies.

We are eager to see other qualitative evaluation approaches flourish in the future, for instance by leveraging interpretability methods, by identifying the inherent limitations of inductive biases, or by exploring the mechanical interactions between corresponding concepts in multimodal setups, in the hope that these works will lead to other extensions of the current AI paradigm.

Second, we want to emphasize that we do not consider representation degeneration to be inherently harmful to language models. As a matter of fact, as described in [Section 4.3.3](#), several works prove that more anisotropic model can perform better for some downstream applications. Apart from cases of extreme dimensional collapse, there is no reason why distorted representational spaces should *inevitably* lead to less performant models. We instead argue that the observed distortions are both counter-intuitive and should not be “natural”, in the sense that random gradient-based updates are unlikely to lead to such geometries. As we have seen along the chapters of this thesis, explaining these distortions can shed light on constraints applied by the training procedure or the model architecture. Hence, reinforcing these distortions may have a positive impact on the model’s behavior as it may help to strengthen inherent properties of the models (e.g. the sparsity of attention maps).

Finally, it could be argued that the methods developed in [Chapter 8](#), [Chapter 9](#) and [Chapter 6](#) still rely on classical evaluation methods, i.e. benchmarks evaluating models performance on examples corresponding to downstream tasks, as a final performance metric that decides on the relevance of each approach. We would like to clarify that we do not argue in this thesis *against* any form of behaviorism in the approaches during training and inference, but rather advocate that the current paradigm where approaches and models are selected and ranked based *solely* on such quantitative performance evaluations - which may include latency and memory-greediness measurements - could be completed with technical qualitative observations in order to design generally “better” approaches and models. In that process, and as these benchmarks mimic the final use of models, it is crucial to measure the impact of novel methods driven by these qualitative observations on the quantitatively measured downstream performance of the models, in order to assess whether these methods generally improve language models and whether these qualitative observations are actually relevant to the overall quality of the model.

### 11.2.2 BEHAVIORISM AND THE COMMUNITY

As we started presenting initial ideas about this work, a remark that was redundant among peers could be summarized with the following questions:

*Why should we care about anisotropy? Language models still work fine, right?*

While it is true that as far as we tested them, the most impressive recent large language models use anisotropic intermediate representations, these questions show that the behaviorist point of view is not only widespread in research works but is also deeply embedded in the way some members of the research community view the problems at hand in modern Natural Language Processing.

We argue that this focus on performance has been driven by industrial research through some of the most influential recent works, and corresponds with an engineering view of NLP. Interestingly, this view does not seem to reflect the opinion of the majority of NLP researchers, as pictured in a meta-survey conducted among the ACL community ([Michael et al., 2023](#)).

In this meta-survey, [Michael et al. \(2023\)](#) report that only 17% of NLP researchers believe that scale should solve any problem, and 51% that LMs “understand” language. As much as 88% opine that there is too much focus on benchmarks, 72% that there is too much focus on scale, and 82% that we should incorporate more interdisciplinary insights from sociolinguistics or cognitive science, for instance. Additionally, only 42% believe that interpretability is not a promising research direction, among which industrial researchers are over-represented.

Overall, we observe that a vast fraction of the research community, and more particularly academic scholars, could embrace other paradigms towards improving language modeling, through better interpretations of the models and by leveraging concepts from other fields.

The field of mechanistic interpretability, which aligns with this research direction, has recently drawn attention from the field. We hope that these techniques will provide insights that will allow to identify caveats in the structure of language models.

## 11.3 PERSPECTIVES

In this final section, we provide future perspectives for the work presented in this thesis, including ideas that are natural continuations of this work, and perspectives that this work offers with respect to the future of the field of NLP.

### 11.3.1 FUTURE WORK

**TOKENIZER-FREE HEADLESS LANGUAGE MODELING** A logical way forward from this work consists in combining the approaches described in [Chapter 8](#) and [Chapter 9](#) in order to alleviate models from the need to predict in byte-level output spaces at training time. As a matter of fact, training a decoder that handles pooled byte representations as in MANTa-LM using the contrastive loss described could mitigate the byte-level decoding throughput bottleneck described in [Chapter 9](#).

Two limitations arise from this idea. First, for causal language models, it is not trivial how to impose causality in such a framework, as the MANTa module does not impede byte information to be shared across blocks, and the prediction at index  $t + 1$  could leverage leaky information from the block at index  $t$ . Second, another issue that would affect both causal and non-causal language models is the difficulty to guarantee that the output space of the MANTa module (or any other character-pooling module) could be easily mapped to the output space of the whole model. For instance, if representation degeneration were to occur in the post-pooling section of the model itself for sparsity reasons, it would probably be harder to map input and output spaces especially if we use normalization layers in the character pooling module.

We have conducted experiments in that direction, by training masked language models based on the CharacterBERT pooling module ([El Boukkouri et al., 2020](#)) using the CWT loss, and observed training instability and poor convergence. We hypothesize that the second aforementioned reason is the main cause to the encountered difficulties, and leave more exploration of this direction for the future.

**CHARACTER-TO-SEMANTIC BOTTLENECK** Another hypothesis that could help explain the lower performance of tokenizer-free headless language models, and more generally of byte-based models, could be the inherent difficulty to map character-level representations to higher-granularity semantic features, and the reduced interpolation properties offered by such a map. In token-level language models, input token embeddings are the leaves of the back-propagation graph and can thus be freely optimized, i.e. they can adopt *any* values that optimize the final loss as they are features and not activations. In the case of byte or character-level models, the pooled representations used as the input of the larger predictive model are the results of the application of a module based on specific inductive biases on byte or character-level features. Hence, it can be questioned whether these pooled representations can efficiently and easily rival with the expressivity of classical token-level representations.

Exploring this question requires measuring the semantic expressivity of pooled representations in existing character-level models, and quantifying the inherent difficulty of the character-to-semantic mapping, either by training models to perform such a mapping, or by using a similar analysis to the softmax bottleneck identification in [Chapter 6](#).

**DOUBLY DYNAMIC KV CACHE COMPRESSION** Nawrot et al. (2024) and our work in Chapter 10 pave the way for gradient-based KV cache compression. However, both approaches are limited by their inability to properly *revisit* past decisions as generation advances. We can quickly remark that non-revisiting compression schemes are bound to linear memory complexity: if the probability of creating a new cache slot does not converge to 0, then the cache memory is lower bound by a linear function.

However, we currently foresee two solutions towards reaching such revisiting compression schemes: re-compressing compressed sequence during generation, or allow for a non-contiguous compression scheme, that would allow to avoid repeating information when adding new slots. The first method poses challenges during training, as revisiting the nature of  $K$  and  $V$  representations during generation is hard to translate to a parallel computation in the self-attention process without incurring substantial efficiency costs. The second method poses modeling challenges, as it would require another compression mapping mechanism, and would probably be harder to control with a compression loss based solely on the decision variables as in Chapter 10.

**THEORETICAL EXPLORATION OF THE SOFTMAX BOTTLENECK** Although Chapter 6 provides thorough empirical arguments in favor of our claim, we acknowledge that Theorem 6.3.3 is not sufficient to fully characterize the interaction between spectral properties of the context-to-logits mapping and the optimal performance of the model. While exploring this theoretical problem, we found that the mathematical tools at our disposal - and maybe within our grasp - needed to be developed to help in this specific scenario.

For instance, the more complete problem could be framed as a weighted low-rank approximation (WLRA) where the individual contextual cross-entropy measurements could be weighted by the corresponding context probability. Nevertheless, the WLRA problem is not as extensively studied in the literature as we expected (Razenshteyn et al., 2016), which did not provide relevant insights about our problem. Another foreseen limitation is the lack of clear relation between the rank function and the logarithm, especially when we need to estimate or bound the rank of a logit matrix resulting from a linear operation for which the rank can easily be measured.

We recognize that these limitations may also be overcome via a better expertise in the theoretical field that underly this question. We leave a deeper exploration of the theoretical characterization of our empirical observations for future work.

**ALTERNATIVES TO CROSS-ENTROPY** We argue that extending the observations made in Chapter 5 and Chapter 6 to other types of training objectives would yield interesting insights. First, we could study the impact of the CWT loss on both the representational biases of the model from an empirical viewpoint, and the softmax bottleneck effect from a theoretical viewpoint. We also hypothesize that knowledge distillation techniques such as DistilBERT (Sanh et al., 2019) may have interesting implications for these qualitative evaluation of the representations.

Along this thesis, we have explored alternative objectives for language models inspired from knowledge distillation, hoping to reduce the impact of frequency-based distortions on the inner workings of language models. For instance, we have framed language modeling as a multi-label classification task, where the model is asked to produce an acceptability probability for each token at each position, using labels generated using the contextual distributions of a classical language

## 11 Conclusion

model. This approach led to promising initial results, and we wish to explore this method further in future works.

**UNDERSTANDING QKV GEOMETRY** We have shown in [Chapter 7](#) that analyzing the geometry of attention-specific representations could shed light on phenomena that occur in language models. [Devoto et al. \(2024\)](#) take advantage of a geometrical analysis of QKV representations to propose an efficient KV cache compression technique. We suggest that this question is under-represented in the literature and could lead to substantial improvements, both regarding current models and when proposing novel architectures.

### 11.3.2 BROADER PERSPECTIVES

**MODEL EVALUATION** In this thesis, we argue in favor of an extension of what the community broadly deems as model evaluation. Practically, this includes refreshing probing techniques and similar counterparts to evaluate the internal representations of models. This may also be implemented by developing measurements of the informational efficiency of the inner mechanisms of models, for instance by measuring the amount of operations that lead to ignored information through sparse patterns, or the amount of computation that inherently extracts similar features from the data and is thus used inefficiently.

These examples could constitute benchmarks of their own, and be built into systematical evaluation libraries. We believe that creating uniform qualitative evaluation tools would dramatically ease this kind of analysis, as they could be seen as a model *debugging* toolboxes that would provide extensive insights for architecture research among others.

**SAFETY & ALIGNMENT** As stated in [Chapter 1](#), we believe that there is a direct, yet non-exhaustive, link between considering language-generating models as black-box systems and the increasing concerns in the general public and the scientific community with respect to AI safety. In our opinion, qualitative evaluation loops, where models are adapted to improve an observed suboptimal mechanism, should lead to more interpretable models as it will push the models towards more meaningful internal behaviors.

Thereby, we argue that non-behaviorist approaches actually bring us closer to safer AI systems, especially when they contribute to the design of such models. For instance, more interpretable models could be more easily edited and aligned, as their inner conceptual knowledge and its alignment to the users' expectations could be controlled more easily.

We finally hypothesize that behaviorist alignment methods do not guarantee that the model is inherently aligned with the target behavior. For instance, a model could learn to produce expected outputs superficially, while still being perfectly able to produce unsafe content. This is a major challenge in the alignment field, which may be solved by internal alignment through qualitative evaluation, and potentially model editing to enforce alignment at weight level directly.

**INTERPRETABILITY** In a nutshell, our work could be understood as a pledge for a paradigm shift in the interpretability field, going from current methods that help provide explanations and analyze the capabilities of the models to methods that provide insights about the limitations and possible improvements of models.

We believe that our contribution to interpretability itself could be further extended to more powerful tools that take linguistic information into account, and allow for a finer analysis of the inner workings of models. For instance, it would be interesting to see how the most advanced interpretability techniques relying on gradient-based estimations or neural circuits (see [Chapter 4](#)) could be applied to characterize the limitations of language models, and how improvements could be made, for instance by enhancing gradient-based feedback and causation patterns, or by facilitating the emergence of efficient and non-redundant neural circuits in language models.

Finally, we advocate that exploring these perspectives could fluidify the interactions between these three fields, which currently do not benefit from one another in an extensive way. We hope that building a framework that tends to unify these domains in the next years will bring substantial advancements to the field of NLP.





## *11 Conclusion*

## BIBLIOGRAPHY

- Abubakar Abid, Maheen Farooqi, and James Zou. 2021. [Persistent anti-muslim bias in large language models](#). In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES ’21, page 298–306, New York, NY, USA. Association for Computing Machinery.
- David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. 1985. [A learning algorithm for boltzmann machines](#). *Cognitive Science*, 9(1):147–169.
- Henry Adams, Manuchehr Aminian, Elin Farnell, Michael Kirby, Joshua Mirth, Rachel Neville, Chris Peterson, and Clayton Shonkwiler. 2020. A fractal dimension for measures via persistent homology. In *Topological Data Analysis*, pages 1–31, Cham. Springer International Publishing.
- Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. [Fine-grained analysis of sentence embeddings using auxiliary prediction tasks](#). In *International Conference on Learning Representations*.
- Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant Nair, Ilya Soloveychik, and Purushotham Kamath. 2024. [Keyformer: Kv cache reduction through key tokens selection for efficient generative inference](#). In *Proceedings of Machine Learning and Systems*, volume 6, pages 114–127.
- Arash Ahmadian, Saurabh Dash, Hongyu Chen, Bharat Venkitesh, Zhen Stephen Gou, Phil Blunsom, Ahmet Üstün, and Sara Hooker. 2023. [Intriguing properties of quantization at scale](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. [GQA: Training generalized multi-query transformer models from multi-head checkpoints](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.
- Mira Ait-Saada and Mohamed Nadif. 2023. [Is anisotropy truly harmful? a case study on text clustering](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1194–1203, Toronto, Canada. Association for Computational Linguistics.
- Robin Algayres, Tristan Ricoul, Julien Karadayi, Hugo Laurençon, Salah Zaiem, Abdelrahman Mohamed, Benoît Sagot, and Emmanuel Dupoux. 2022. [DP-parse: Finding word boundaries from raw speech with an instance lexicon](#). *Transactions of the Association for Computational Linguistics*, 10:1051–1065.

## Bibliography

- Carl Allen and Timothy Hospedales. 2019. [Analogies explained: Towards understanding word embeddings](#). In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 223–231. PMLR.
- Ebtessam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. [The falcon series of open language models](#).
- Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers, and Gregor Weber. 2020. [Common voice: A massively-multilingual speech corpus](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4218–4222, Marseille, France. European Language Resources Association.
- M. Aronoff. 1993. [Morphology by Itself: Stems and Inflectional Classes](#). Linguistic Inquiry Monographs. MIT Press.
- Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. [A simple but tough-to-beat baseline for sentence embeddings](#). In *International Conference on Learning Representations*.
- Jordan Ash, Surbhi Goel, Akshay Krishnamurthy, and Dipendra Misra. 2022. [Investigating the role of negatives in contrastive representation learning](#). In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 7187–7209. PMLR.
- Pranjal Awasthi, Nishanth Dikkala, and Pritish Kamath. 2022. [Do more negative samples necessarily hurt in contrastive learning?](#) In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1101–1116. PMLR.
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. [Layer normalization](#).
- Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020a. [wav2vec 2.0: A framework for self-supervised learning of speech representations](#).
- Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020b. [wav2vec 2.0: A framework for self-supervised learning of speech representations](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460. Curran Associates, Inc.
- Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. 2010. A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology*, 1(1):4.
- Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

- Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2022. [BEiT: BERT pre-training of image transformers](#). In *International Conference on Learning Representations*.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. [On the dangers of stochastic parrots: Can language models be too big?](#) In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT ’21, page 610–623, New York, NY, USA. Association for Computing Machinery.
- Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. [Representation learning: A review and new perspectives](#). *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.
- Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems*, 13.
- Yoshua Bengio and Jean-Sébastien Senecal. 2003. [Quick training of probabilistic neural nets by importance sampling](#). In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, volume R4 of *Proceedings of Machine Learning Research*, pages 17–24. PMLR. Reissued by PMLR on 01 April 2021.
- Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023a. [Pythia: A suite for analyzing large language models across training and scaling](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.
- 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. 2023b. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.
- Geetanjali Bihani and Julia Rayz. 2021. [Low anisotropy sense retrofitting \(LASeR\) : Towards isotropic and sense enriched representations](#). In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 81–95, Online. Association for Computational Linguistics.
- Richard Billingsley and James Curran. 2012. [Improvements to training an RNN parser](#). In *Proceedings of COLING 2012*, pages 279–294, Mumbai, India. The COLING 2012 Organizing Committee.
- Daniel Biś, Maksim Podkorytov, and Xiuwen Liu. 2021. [Too much in common: Shifting of embeddings in transformer language models and its implications](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5117–5130, Online. Association for Computational Linguistics.

## Bibliography

- William Biscarri, Sihai Dave Zhao, and Robert J. Brunner. 2018. [A simple and fast method for computing the poisson binomial distribution function](#). *Computational Statistics & Data Analysis*, 122:92–100.
- Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. [GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow](#).
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. [Enriching word vectors with subword information](#). *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Rishi Bommasani, Kelly Davis, and Claire Cardie. 2020. [Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4758–4781, Online. Association for Computational Linguistics.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. [Language models are few-shot learners](#).
- Tommaso Caselli, Irene Dini, and Felice Dell’Orletta. 2022. [How about time? probing a multilingual language model for temporal relations](#). In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3197–3209, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. [Universal sentence encoder for English](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.
- Haw-Shiuan Chang and Andrew McCallum. 2022. [Softmax bottleneck makes language models unable to represent multi-mode word distributions](#). In *Proceedings of the 60th Annual Meeting*

of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8048–8073, Dublin, Ireland. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML’20. JMLR.org.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014a. [Learning phrase representations using RNN encoder–decoder for statistical machine translation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. [Learning phrase representations using rnn encoder–decoder for statistical machine translation](#).

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2024. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1).

Gobinda G Chowdhury. 2010. *Introduction to modern information retrieval*. Facet publishing.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Rob Churchill and Lisa Singh. 2022. The evolution of topic modeling. *ACM Computing Surveys*, 54(10s):1–35.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019a. [BoolQ: Exploring the surprising difficulty of natural yes/no questions](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

## Bibliography

- Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022a. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.
- Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022b. [Canine: Pre-training an efficient tokenization-free encoder for language representation](#). *Transactions of the Association for Computational Linguistics*, 10:73–91.
- Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019b. [What does BERT look at? an analysis of BERT’s attention](#). In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [Electra: Pre-training text encoders as discriminators rather than generators](#). In *International Conference on Learning Representations*.
- Lionel Clément and Éric Villemonte de La Clergerie. 2005. [MAF: a Morphosyntactic Annotation Framework](#). In *2nd Language & Technology Conference (LTC’05)*, 2nd Language & Technology Conference (LTC’05), pages 90–94, Poznan, Poland.
- Together Computer. 2023. [Redpajama: an open dataset for training large language models](#).
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.
- Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. [Supervised learning of universal sentence representations from natural language inference data](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.
- Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018a. [What you can cram into a single \\$&!#\\* vector: Probing sentence embeddings for linguistic properties](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.
- Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018b. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2023. [Analyzing transformers in embedding space](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16124–16170, Toronto, Canada. Association for Computational Linguistics.

- Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. [A dataset of information-seeking questions and answers anchored in research papers](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.
- Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.
- Gregoire Deletang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. 2024. [Language modeling is compression](#). In *The Twelfth International Conference on Learning Representations*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. 2024. [A simple and effective  \$l\_2\$  norm-based strategy for kv cache compression](#).
- Paul-Ambroise Duquenne, Holger Schwenk, and Benoit Sagot. 2023. [SONAR: sentence-level multimodal and language-agnostic representations](#).
- Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun’ichi Tsujii. 2020. [CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2018. [Sigmoid-weighted linear units for neural network function approximation in reinforcement learning](#). *Neural Networks*, 107.
- Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. <Https://transformer-circuits.pub/2021/framework/index.html>.
- Kawin Ethayarajh. 2019. [How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

## Bibliography

- Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2019. [Towards understanding linear word analogies](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262, Florence, Italy. Association for Computational Linguistics.
- Fahim Faisal and Antonios Anastasopoulos. 2021. [Investigating post-pretraining representation alignment for cross-lingual question answering](#). In *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*, pages 133–148, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Fahim Faisal and Antonios Anastasopoulos. 2023. [Geographic and geopolitical biases of language models](#). In *Proceedings of the 3rd Workshop on Multi-lingual Representation Learning (MRL)*, pages 139–163, Singapore. Association for Computational Linguistics.
- Fahim Faisal, Yinkai Wang, and Antonios Anastasopoulos. 2022. [Dataset geography: Mapping language data to language users](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3381–3411, Dublin, Ireland. Association for Computational Linguistics.
- Angela Fan, Mike Lewis, and Yann Dauphin. 2018. [Hierarchical neural story generation](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.
- Manuel Faysse, Patrick Fernandes, Nuno M. Guerreiro, António Loison, Duarte M. Alves, Caio Corro, Nicolas Boizard, João Alves, Ricardo Rei, Pedro H. Martins, Antoni Bigata Casademunt, François Yvon, André F. T. Martins, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2024. [Croissantlm: A truly bilingual french-english language model](#).
- Christiane Fellbaum. 1998. [WordNet: An Electronic Lexical Database](#). The MIT Press.
- Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. [Language-agnostic BERT sentence embedding](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.
- Matthew Finlayson, John Hewitt, Alexander Koller, Swabha Swayamdipta, and Ashish Sabharwal. 2024. [Closing the curious case of neural text degeneration](#). In *The Twelfth International Conference on Learning Representations*.
- R. A. Fisher. 1922. [On the mathematical foundations of theoretical statistics](#). *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222:309–368.
- W. N. Francis and H. Kucera. 1979. [Brown corpus manual](#). Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US.
- Markus Freitag and Yaser Al-Onaizan. 2017. [Beam search strategies for neural machine translation](#). In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver. Association for Computational Linguistics.

- Hao Fu, Shaojun Zhou, Qihong Yang, Junjie Tang, Guiquan Liu, Kaikui Liu, and Xiaolong Li. 2021. [Lrc-bert: Latent-representation contrastive knowledge distillation for natural language understanding](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12830–12838.
- Kunihiko Fukushima. 1969. [Visual feature extraction by a multilayered network of analog threshold elements](#). *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333.
- Philip Gage. 1994. [A new algorithm for data compression](#). *The C Users Journal archive*, 12:23–38.
- Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019a. [Representation degeneration problem in training natural language generation models](#).
- Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. 2019b. [Representation degeneration problem in training natural language generation models](#). In *International Conference on Learning Representations*.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. [A framework for few-shot language model evaluation](#).
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. [SimCSE: Simple contrastive learning of sentence embeddings](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Corrado Gini. 1912. *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche.[Fasc. I.]*. Tipogr. di P. Cuppini.
- Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.
- Andreas Grivas, Nikolay Bogoychev, and Adam Lopez. 2022. [Low-rank softmax can have unargmaxable classes in theory but rarely in practice](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6738–6758, Dublin, Ireland. Association for Computational Linguistics.
- Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2018. [Cognate-aware morphological segmentation for multilingual neural translation](#). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 386–393, Belgium, Brussels. Association for Computational Linguistics.

## Bibliography

- E.J. Gumbel. 1935. [Les valeurs extrêmes des distributions statistiques](#). *Annales de l'institut Henri Poincaré*, 5(2):115–158.
- Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. 2023. [Visual attention network](#). *Computational Visual Media*, 9(4):733–752.
- Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. [Distributional vectors encode referential attributes](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 12–21, Lisbon, Portugal. Association for Computational Linguistics.
- Wes Gurnee and Max Tegmark. 2024. [Language models represent space and time](#). In *The Twelfth International Conference on Learning Representations*.
- Michael Gutmann and Aapo Hyvärinen. 2010. [Noise-contrastive estimation: A new estimation principle for unnormalized statistical models](#). In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy. PMLR.
- Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. 2017. [news-please: A generic news crawler and extractor](#). In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223.
- Katharina Hämmерl, Alina Fastowski, Jindřich Libovický, and Alexander Fraser. 2023a. [Exploring anisotropy and outliers in multilingual language models for cross-lingual semantic sentence similarity](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7023–7037, Toronto, Canada. Association for Computational Linguistics.
- Katharina Hämmерl, Alina Fastowski, Jindřich Libovický, and Alexander Fraser. 2023b. [Exploring anisotropy and outliers in multilingual language models for cross-lingual semantic sentence similarity](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7023–7037, Toronto, Canada. Association for Computational Linguistics.
- R. W. Hamming. 1950. [Error detecting and error correcting codes](#). *The Bell System Technical Journal*, 29(2):147–160.
- ZS Harris. 1954. Distributional structure.
- Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. [Deep residual learning for image recognition](#). In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. [Deep residual learning for image recognition](#). In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

- Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. [Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing](#).
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. [{DEBERTA}: {DECODING}-{enhanced} {bert} {with} {disentangled} {attention}](#). In *International Conference on Learning Representations*.
- Dan Hendrycks and Kevin Gimpel. 2023. [Gaussian error linear units \(gelus\)](#).
- John Hewitt. 2019. [Designing and interpreting probes](#).
- Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. [Learning distributed representations of sentences from unlabelled data](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, San Diego, California. Association for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- A. E. Hoerl and R. W. Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67.
- Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. [Robust disambiguation of named entities in text](#). In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh, Scotland, UK. Association for Computational Linguistics.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. [Training compute-optimal large language models](#).
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020a. [The curious case of neural text degeneration](#). In *International Conference on Learning Representations*.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020b. [The curious case of neural text degeneration](#). In *International Conference on Learning Representations*.
- Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1990. [Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks](#). *Neural Networks*, 3(5):551–560.
- Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. [Hubert: Self-supervised speech representation learning by masked prediction of hidden units](#). *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 29:3451–3460.

## Bibliography

- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- T. Hua, W. Wang, Z. Xue, S. Ren, Y. Wang, and H. Zhao. 2021. [On feature decorrelation in self-supervised learning](#). In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9578–9588, Los Alamitos, CA, USA. IEEE Computer Society.
- Yuzhen Huang, Jinghan Zhang, Zifei Shan, and Junxian He. 2024. [Compression represents intelligence linearly](#). In *First Conference on Language Modeling*.
- Sergey Ioffe and Christian Szegedy. 2015. [Batch normalization: Accelerating deep network training by reducing internal covariate shift](#). In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France. PMLR.
- Shadi Iskander, Kira Radinsky, and Yonatan Belinkov. 2023. [Shielded representations: Protecting sensitive attributes through iterative gradient-based projection](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5961–5977, Toronto, Canada. Association for Computational Linguistics.
- Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. 2021. [Perceiver IO: A general architecture for structured inputs & outputs](#). *CoRR*, abs/2107.14795.
- Nihal Jain, Dejiao Zhang, Wasi Uddin Ahmad, Zijian Wang, Feng Nan, Xiaopeng Li, Ming Tan, Ramesh Nallapati, Baishakhi Ray, Parminder Bhatia, Xiaofei Ma, and Bing Xiang. 2023. [ContraCLM: Contrastive learning for causal language model](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6436–6459, Toronto, Canada. Association for Computational Linguistics.
- Eric Jang, Shixiang Gu, and Ben Poole. 2017. [Categorical reparameterization with gumbel-softmax](#). In *International Conference on Learning Representations*.
- Matthew A. Jaro. 1989. [Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida](#). *Journal of the American Statistical Association*, 84(406):414–420.
- Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. [What does BERT learn about the structure of language?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.
- Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. [On using very large target vocabulary for neural machine translation](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. [Mistral 7b](#).
- Yibo Jiang, Goutham Rajendran, Pradeep Kumar Ravikumar, Bryon Aragam, and Victor Veitch. 2024. [On the origins of linear representations in large language models](#). In *Forty-first International Conference on Machine Learning*.
- Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. [PubMedQA: A dataset for biomedical research question answering](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.
- Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. 2022. [Understanding dimensional collapse in contrastive self-supervised learning](#). In *International Conference on Learning Representations*.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. [SpanBERT: Improving pre-training by representing and predicting spans](#). *Transactions of the Association for Computational Linguistics*, 8:64–77.
- Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam M. Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *ArXiv*, abs/1602.02410.
- Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st edition. Prentice Hall PTR, USA.
- Sekitoshi Kanai, Yasuhiro Fujiwara, Yuki Yamanaka, and Shuichi Adachi. 2018. [Sigsoftmax: Reanalysis of the softmax bottleneck](#). In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. 2020. [Scaling laws for neural language models](#). *ArXiv*, abs/2001.08361.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI conference on artificial intelligence*.
- Tassilo Klein and Moin Nabi. 2023. [miCSE: Mutual information contrastive learning for low-shot sentence embeddings](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6177, Toronto, Canada. Association for Computational Linguistics.
- Arne Köhn. 2015. [What's in an embedding? analyzing word embeddings through multilingual evaluation](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073, Lisbon, Portugal. Association for Computational Linguistics.

## Bibliography

- Hadas Kotek, Rikker Dockum, and David Sun. 2023. Gender bias and stereotypes in large language models. In *Proceedings of the ACM collective intelligence conference*, pages 12–24.
- Anders Krogh and John A. Hertz. 1991. A simple weight decay can improve generalization. In *Proceedings of the 4th International Conference on Neural Information Processing Systems, NIPS'91*, page 950–957, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Taku Kudo. 2018. [Subword regularization: Improving neural network translation models with multiple subword candidates](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.
- N. Kishore Kumar and J. Schneider. 2017. [Literature survey on low rank approximation of matrices](#). *Linear and Multilinear Algebra*, 65(11):2212–2244.
- Sachin Kumar and Yulia Tsvetkov. 2019. [Von mises-fisher loss for training sequence to sequence models with continuous outputs](#). In *Proc. of ICLR*.
- Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.
- Yann LeCun. 2022. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62.
- Benjamin Lefauveux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, and Daniel Haziza. 2022. xformers: A modular and hackable transformer modelling library. <https://github.com/facebookresearch/xformers>.
- VI Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10, page 707.
- Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. [On the sentence embeddings from pre-trained language models](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online. Association for Computational Linguistics.
- Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. [Towards general text embeddings with multi-stage contrastive learning](#).
- Davis Liang, Hila Gonen, Yuning Mao, Rui Hou, Naman Goyal, Marjan Ghazvininejad, Luke Zettlemoyer, and Madian Khabsa. 2023. [Xlm-v: Overcoming the vocabulary bottleneck in multilingual masked language models](#).
- Davis Liang, Hila Gonen, Yuning Mao, Rui Hou, Naman Goyal, Marjan Ghazvininejad, Luke Zettlemoyer, and Madian Khabsa. 2023. [XLM-V: Overcoming the Vocabulary Bottleneck in Multilingual Masked Language Models](#). *arXiv e-prints*, page arXiv:2301.10472.

- Anne-Laure Ligozat, Julien Lefevre, Aurélie Bugeau, and Jacques Combaz. 2022. [Unraveling the hidden environmental impacts of ai solutions for environment life cycle assessment of ai solutions](#). *Sustainability*, 14(9).
- Ying-Chen Lin. 2021. [Breaking the softmax bottleneck for sequential recommender systems with dropout and decoupling](#).
- Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. [Linguistic knowledge and transferability of contextual representations](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. [Roberta: A robustly optimized bert pretraining approach](#).
- Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986.
- Robert E. Longacre. 1970. [Hierarchy in Language](#), pages 173–196. De Gruyter Mouton, Berlin, Boston.
- Max M. Louwerse and Nick Benesh. 2012. [Representing spatial structure through maps and language: Lord of the rings encodes the spatial structure of middle earth](#). *Cognitive Science*, 36(8):1556–1569.
- Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. [Char-BERT: Character-aware pre-trained language model](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 39–50, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Zhuang Ma and Michael Collins. 2018. [Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3698–3707, Brussels, Belgium. Association for Computational Linguistics.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. [Learning word vectors for sentiment analysis](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.
- Anemily Machina and Robert Mercer. 2024. [Anisotropy is not inherent to transformers](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4892–4907, Mexico City, Mexico. Association for Computational Linguistics.

## Bibliography

- Andreas Madsen, Siva Reddy, and Sarath Chandar. 2022. [Post-hoc interpretability for neural nlp: A survey](#). *ACM Comput. Surv.*, 55(8).
- Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamel Seddah, and Benoît Sagot. 2020. [CamemBERT: a tasty French language model](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.
- Clara Meister, Wojciech Stokowiec, Tiago Pimentel, Lei Yu, Laura Rimell, and Adhiguna Kuncoro. 2023. [A natural bias for language generation models](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 243–255, Toronto, Canada. Association for Computational Linguistics.
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. [Pointer sentinel mixture models](#).
- Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. [Pointer sentinel mixture models](#). In *International Conference on Learning Representations*.
- William Merrill, Ashish Sabharwal, and Noah A. Smith. 2022. [Saturated transformers are constant-depth threshold circuits](#). *Transactions of the Association for Computational Linguistics*, 10:843–856.
- Julian Michael, Ari Holtzman, Alicia Parrish, Aaron Mueller, Alex Wang, Angelica Chen, Divyam Madaan, Nikita Nangia, Richard Yuanzhe Pang, Jason Phang, and Samuel R. Bowman. 2023. [What do NLP researchers believe? results of the NLP community metasurvey](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16334–16368, Toronto, Canada. Association for Computational Linguistics.
- Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. [Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp](#).
- Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013a. [Efficient estimation of word representations in vector space](#).
- Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. [Linguistic regularities in continuous space word representations](#). In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.
- Tomáš Mikolov, Martin Karafiat, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. [Recurrent neural network based language model](#). In *Proc. Interspeech 2010*, pages 1045–1048.
- Yasumasa Miyamoto and Kyunghyun Cho. 2016. [Gated word-character recurrent language model](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1992–1997, Austin, Texas. Association for Computational Linguistics.

- Andriy Mnih and Yee Whye Teh. 2012. [A fast and simple algorithm for training neural probabilistic language models.](#)
- Md Mofijul Islam, Gustavo Aguilar, Pragaash Ponnusamy, Clint Solomon Mathialagan, Chengyuan Ma, and Chenlei Guo. 2022. A vocabulary-free multilingual neural tokenizer for end-to-end task learning. *arXiv e-prints*, pages arXiv–2204.
- Daniel Morales-Brottons, Thijs Vogels, and Hadrien Hendrikx. 2024. [Exponential moving average of weights in deep learning: Dynamics and benefits.](#) *Transactions on Machine Learning Research*.
- Frederic Morin and Yoshua Bengio. 2005. [Hierarchical probabilistic neural network language model.](#) In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pages 246–252. PMLR. Reissued by PMLR on 30 March 2021.
- Jiaqi Mu and Pramod Viswanath. 2018. [All-but-the-top: Simple and effective postprocessing for word representations.](#) In *International Conference on Learning Representations*.
- Putra Fissabil Muhammad, Retno Kusumaningrum, and Adi Wibowo. 2021. [Sentiment analysis using word2vec and long short-term memory \(lstm\) for indonesian hotel reviews.](#) *Procedia Computer Science*, 179:728–735. 5th International Conference on Computer Science and Computational Intelligence 2020.
- Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. [StereoSet: Measuring stereotypical bias in pretrained language models.](#) 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 5356–5371, Online. Association for Computational Linguistics.
- Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. [CrowS-pairs: A challenge dataset for measuring social biases in masked language models.](#) In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.
- Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. 2023. [Efficient transformers with dynamic token pooling.](#) In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6403–6417, Toronto, Canada. Association for Computational Linguistics.
- Piotr Nawrot, Adrian Łaćucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. 2024. [Dynamic memory compression: Retrofitting LLMs for accelerated inference.](#) In *Forty-first International Conference on Machine Learning*.
- Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. [On structuring probabilistic dependences in stochastic language modelling.](#) *Computer Speech & Language*, 8(1):1–38.
- Jerry Ngo and Yoon Kim. 2024. [What do language models hear? probing for auditory representations in language models.](#)

## Bibliography

- Nostalgebraist. 2020. [interpreting gpt: the logit lens](#).
- Aniruddha Nrusimha, Mayank Mishra, Naigang Wang, Dan Alistarh, Rameswar Panda, and Yoon Kim. 2024. [Mitigating the impact of outlier channels for language model quantization with activation regularization](#).
- Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. 2024. [Transformers are multi-state rnns](#).
- Laurel Orr, Megan Leszczynski, Simran Arora, Sen Wu, Neel Guha, Xiao Ling, and Christopher Re. 2020. [Bootleg: Chasing the tail with self-supervised named entity disambiguation](#).
- Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. [Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures](#). In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*, Cardiff, United Kingdom. Leibniz-Institut für Deutsche Sprache.
- John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. 2008. [Gpu computing](#). *Proceedings of the IEEE*, 96(5):879–899.
- Sinno Jialin Pan and Qiang Yang. 2010. [A survey on transfer learning](#). *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.
- Rrubaa Panchendarajan and Aravindh Amaresan. 2018. [Bidirectional LSTM-CRF for named entity recognition](#). In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong. Association for Computational Linguistics.
- Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. [The linear representation hypothesis and the geometry of large language models](#). In *Forty-first International Conference on Machine Learning*.
- Guilherme Penedo, Hynek Kydlicek, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. [The fineweb datasets: Decanting the web for the finest text data at scale](#).
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [GloVe: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. [Deep contextualized word representations](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. [Knowledge enhanced contextual word representations](#). In *Proceedings*

of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. [Dissecting contextual word embeddings: Architecture and representation](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. [Language model tokenizers introduce unfairness between languages](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 36963–36990. Curran Associates, Inc.

Jackson Petty, Sjoerd van Steenkiste, Ishita Dasgupta, Fei Sha, Dan Garrette, and Tal Linzen. 2023. [The impact of depth and width on transformer language model generalization](#).

Adriana Placani. 2024. [Anthropomorphism in ai: hype and fallacy](#). *AI and Ethics*, 4(3):691–698.

Nirmalendu Prakash and Roy Ka-Wei Lee. 2023. [Layered bias: Interpreting bias in pretrained large language models](#). In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 284–295, Singapore. Association for Computational Linguistics.

Ofir Press, Noah Smith, and Mike Lewis. 2022. [Train short, test long: Attention with linear biases enables input length extrapolation](#). In *International Conference on Learning Representations*.

Ofir Press and Lior Wolf. 2017. [Using the output embedding to improve language models](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. [BPE-dropout: Simple and effective subword regularization](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Giovanni Puccetti, Anna Rogers, Aleksandr Drozd, and Felice Dell’Orletta. 2022. [Outlier dimensions that disrupt transformers are driven by frequency](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1286–1304, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. [Robust speech recognition via large-scale weak supervision](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR.

Alec Radford and Karthik Narasimhan. 2018. [Improving language understanding by generative pre-training](#).

## Bibliography

- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020a. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *Journal of Machine Learning Research*, 21(140):1–67.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020b. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *Journal of Machine Learning Research*, 21(140):1–67.
- Sara Rajaei and Mohammad Taher Pilehvar. 2021. [A cluster-based approach for improving isotropy in contextual embedding space](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 575–584, Online. Association for Computational Linguistics.
- Sara Rajaei and Mohammad Taher Pilehvar. 2022. [An isotropy analysis in the multilingual BERT embedding space](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1309–1316, Dublin, Ireland. Association for Computational Linguistics.
- Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.
- Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. [Null it out: Guarding protected attributes by iterative nullspace projection](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online. Association for Computational Linguistics.
- Ilya Razenshteyn, Zhao Song, and David P. Woodruff. 2016. [Weighted low rank approximations with provable guarantees](#). In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC ’16, page 250–263, New York, NY, USA. Association for Computing Machinery.
- Nils Reimers and Iryna Gurevych. 2019. [Sentence-BERT: Sentence embeddings using Siamese BERT-networks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. ["why should i trust you?": Explaining the predictions of any classifier](#). In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

- Matthias C. Rillig, Marlene Ågerstrand, Mohan Bi, Kenneth A. Gould, and Uli Sauerland. 2023a. [Risks and benefits of large language models for the environment](#). *Environmental Science & Technology*, 57(9):3464–3466.
- Matthias C. Rillig, Marlene Ågerstrand, Mohan Bi, Kenneth A. Gould, and Uli Sauerland. 2023b. [Risks and benefits of large language models for the environment](#). *Environmental Science & Technology*, 57(9):3464–3466. PMID: 36821477.
- Herbert Robbins and Sutton Monro. 1951. [A Stochastic Approximation Method](#). *The Annals of Mathematical Statistics*, 22(3):400 – 407.
- Alexey Romanov and Chaitanya Shivade. 2018. [Lessons from natural language inference in the clinical domain](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596, Brussels, Belgium. Association for Computational Linguistics.
- William Rudman and Carsten Eickhoff. 2024a. [Stable anisotropic regularization](#). In *The Twelfth International Conference on Learning Representations*.
- William Rudman and Carsten Eickhoff. 2024b. [Stable anisotropic regularization](#). In *The Twelfth International Conference on Learning Representations*.
- William Rudman, Nate Gillman, Taylor Rayne, and Carsten Eickhoff. 2022. [IsoScore: Measuring the uniformity of embedding space utilization](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3325–3339, Dublin, Ireland. Association for Computational Linguistics.
- David E. Rumelhart and James L. McClelland. 1987. *Learning Internal Representations by Error Propagation*, pages 318–362.
- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. [Imagenet large scale visual recognition challenge](#). *International Journal of Computer Vision*, 115(3):211–252.
- Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. [How good is your tokenizer? on the monolingual performance of multilingual language models](#). 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 3118–3135, Online. Association for Computational Linguistics.
- Jonne Saleva and Constantine Lignos. 2021. [The effectiveness of morphology-aware segmentation in low-resource neural machine translation](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 164–174, Online. Association for Computational Linguistics.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*.

## Bibliography

- Nikhil Sardana and Jonathan Frankle. 2023. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws.
- Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. 2019. wav2vec: Unsupervised Pre-Training for Speech Recognition. In *Proc. Interspeech 2019*, pages 3465–3469.
- Holger Schwenk and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 157–167, Vancouver, Canada. Association for Computational Linguistics.
- Thibault Sellam, Steve Yadlowsky, Ian Tenney, Jason Wei, Naomi Saphra, Alexander D’Amour, Tal Linzen, Jasmijn Bastings, Iulia Raluca Turc, Jacob Eisenstein, Dipanjan Das, and Ellie Pavlick. 2022. The multiBERTs: BERT reproductions for robustness analysis. In *International Conference on Learning Representations*.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. 2018. Time-contrastive networks: Self-supervised learning from video.
- Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.
- Utkarsh Sharma and Jared Kaplan. 2022. Scaling laws from the data manifold dimension. *Journal of Machine Learning Research*, 23(9):1–34.
- Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need.
- A. Sheth, K. Roy, and M. Gaur. 2023. Neurosymbolic artificial intelligence (why, what, and how). *IEEE Intelligent Systems*, 38(03):56–62.
- Luohe Shi, Hongyi Zhang, Yao Yao, Zuchao Li, and Hai Zhao. 2024. Keep the cost down: A review on methods to optimize llm’s kv-cache consumption.
- Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.
- Oleh Shliazhko, Alena Fenogenova, Maria Tikhonova, Anastasia Kozlova, Vladislav Mikhailov, and Tatiana Shavrina. 2024a. mGPT: Few-Shot Learners Go Multilingual. *Transactions of the Association for Computational Linguistics*, 12:58–79.
- Oleh Shliazhko, Alena Fenogenova, Maria Tikhonova, Anastasia Kozlova, Vladislav Mikhailov, and Tatiana Shavrina. 2024b. mgpt: Few-shot learners go multilingual. *Transactions of the Association for Computational Linguistics*, 12:58–79.

- Karen Sparck Jones. 1988. *A statistical interpretation of term specificity and its application in retrieval*, page 132–142. Taylor Graham Publishing, GBR.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. [Roformer: Enhanced transformer with rotary position embedding](#). *Neurocomputing*, 568:127063.
- Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. [Whitening sentence representations for better semantics and faster retrieval](#).
- Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. 2022a. [A contrastive framework for neural text generation](#).
- Yixuan Su, Fangyu Liu, Zaiqiao Meng, Tian Lan, Lei Shu, Ehsan Shareghi, and Nigel Collier. 2022b. [TaCL: Improving BERT pre-training with token-aware contrastive learning](#). In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2497–2507, Seattle, United States. Association for Computational Linguistics.
- Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. 2020. [Contrastive distillation on intermediate representations for language model compression](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 498–508, Online. Association for Computational Linguistics.
- Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.
- Rich Sutton. 2019. [The bitter lesson](#).
- Mingxing Tan and Quoc Le. 2019. [EfficientNet: Rethinking model scaling for convolutional neural networks](#). In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.
- Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. 2022. [Scale efficiently: Insights from pretraining and finetuning transformers](#). In *International Conference on Learning Representations*.
- Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations*.
- Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussonot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney,

## Bibliography

- Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Pier Giuseppe Sessa, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. [Gemma: Open models based on gemini research and technology](#).
- Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. [BERT rediscovers the classical NLP pipeline](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.
- Shiro Terashima, Kazuya Takeda, and Fumitada Itakura. 2003. [A linear space representation of language probability through svd of n-gram matrix](#). *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 86(8):61–70.
- Shivin Thukral, Kunal Kukreja, and Christian Kavouras. 2021. [Probing language models for understanding of temporal expressions](#). In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 396–406, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Yuandong Tian, Xinlei Chen, and Surya Ganguli. 2021. [Understanding self-supervised learning dynamics without contrastive pairs](#). In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10268–10278. PMLR.
- Evgeniia Tokarchuk and Vlad Niculae. 2022. [On target representation in continuous-output neural machine translation](#). In *Proceedings of the 7th Workshop on Representation Learning for NLP*, pages 227–235, Dublin, Ireland. Association for Computational Linguistics.
- Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jegou. 2021. [Training data-efficient image transformers and distillation through attention](#). In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [Llama: Open and efficient foundation language models](#).

- Eduard Tulchinskii, Kristian Kuznetsov, Kushnareva Laida, Daniil Cherniavskii, Sergey Nikolenko, Evgeny Burnaev, Serguei Barannikov, and Irina Piontkovskaya. 2023. [Intrinsic dimension estimation for robust detection of AI-generated texts](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*.
- A. M. Turing. 1950. [Computing machinery and intelligence](#). *Mind*, 59(236):433–460.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019a. [Representation learning with contrastive predictive coding](#).
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019b. [Representation learning with contrastive predictive coding](#).
- Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS’17, page 6309–6318, Red Hook, NY, USA. Curran Associates Inc.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Jesse Vig and Yonatan Belinkov. 2019. [Analyzing the structure of attention in a transformer language model](#). In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.
- Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. [Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. [GLUE: A multi-task benchmark and analysis platform for natural language understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.
- Lingxiao Wang, Jing Huang, Kevin Huang, Ziniu Hu, Guangtao Wang, and Quanquan Gu. 2020a. [Improving neural language generation with spectrum control](#). In *International Conference on Learning Representations*.
- Pei-Hsin Wang, Sheng-Iou Hsieh, Shieh-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juan. 2020b. [Contextual temperature for language modeling](#).
- Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. 2020c. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

## Bibliography

- Tongzhou Wang and Phillip Isola. 2020. [Understanding contrastive representation learning through alignment and uniformity on the hypersphere](#). In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR.
- Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.
- Warren Weaver. 1952. [Translation](#). In *Proceedings of the Conference on Mechanical Translation*, Massachusetts Institute of Technology.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. [Chain of thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*.
- Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. [Neural text generation with unlikelihood training](#). In *International Conference on Learning Representations*.
- Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. 2020. [Visual transformers: Token-based image representation and processing for computer vision](#).
- Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. 2021. [Cvt: Introducing convolutions to vision transformers](#). In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22–31.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th international conference on world wide web*, pages 1391–1399.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. [Efficient streaming language models with attention sinks](#). In *The Twelfth International Conference on Learning Representations*.
- Lizhong Xiao, Guangzhong Wang, and Yang Zuo. 2018. [Research on patent text classification based on word2vec and lstm](#). In *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, volume 01, pages 71–74.
- Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. 2021. [Segformer: Simple and efficient design for semantic segmentation with transformers](#). In *Advances in Neural Information Processing Systems*.

- Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. [Nyströmformer: A nyström-based algorithm for approximating self-attention](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14138–14148.
- Lingling Xu, Haoran Xie, Zongxi Li, Fu Lee Wang, Weiming Wang, and Qing Li. 2023. [Contrastive learning models for sentence representations](#). *ACM Trans. Intell. Syst. Technol.*, 14(4).
- Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022a. [ByT5: Towards a token-free future with pre-trained byte-to-byte models](#). *Transactions of the Association for Computational Linguistics*, 10:291–306.
- Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022b. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.
- Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. [mT5: A massively multilingual pre-trained text-to-text transformer](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.
- Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. [ConSERT: A contrastive framework for self-supervised sentence representation transfer](#). 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 5065–5075, Online. Association for Computational Linguistics.
- Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. [Breaking the softmax bottleneck: A high-rank RNN language model](#). In *International Conference on Learning Representations*.
- Paul Youssef, Osman Koraş, Meijie Li, Jörg Schlötterer, and Christin Seifert. 2023. [Give me the facts! a survey on factual knowledge probing in pre-trained language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15588–15605, Singapore. Association for Computational Linguistics.
- LILI YU, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. [MEGABYTE: Predicting million-byte sequences with multiscale transformers](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Sangwon Yu, Jongyoon Song, Heeseung Kim, Seongmin Lee, Woo-Jong Ryu, and Sungroh Yoon. 2022. [Rare tokens degenerate all tokens: Improving neural text generation via adaptive gradient gating for rare token embeddings](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29–45, Dublin, Ireland. Association for Computational Linguistics.
- Jerrold H Zar. 2005. Spearman rank correlation. *Encyclopedia of Biostatistics*, 7.

## Bibliography

- Man Zhang, Yili Hong, and Narayanaswamy Balakrishnan. 2017. [An algorithm for computing the distribution function of the generalized poisson-binomial distribution](#).
- Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. [Tinyllama: An open-source small language model](#).
- Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuhui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. [Opt: Open pre-trained transformer language models](#).
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. [Bertscore: Evaluating text generation with bert](#). In *International Conference on Learning Representations*.
- Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. 2023. [H2o: Heavy-hitter oracle for efficient generative inference of large language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 34661–34710. Curran Associates, Inc.
- Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018a. [Gender bias in coreference resolution: Evaluation and debiasing methods](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana. Association for Computational Linguistics.
- Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018b. [Learning gender-neutral word embeddings](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4847–4853, Brussels, Belgium. Association for Computational Linguistics.
- Kaitlyn Zhou, Kawin Ethayarajh, and Dan Jurafsky. 2021a. [Frequency-based distortions in contextualized word embeddings](#).
- Kaitlyn Zhou, Kawin Ethayarajh, and Dan Jurafsky. 2021b. [Frequency-based distortions in contextualized word embeddings](#).
- Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.
- George Kingsley Zipf. 1935. *The psycho-biology of language : an introduction to dynamic philology*. The psycho-biology of language: an introduction to dynamic philology. Houghton Mifflin, Oxford, England.

## APPENDIX



## A ADDITIONAL RESULTS FOR CHAPTER 7

### A.1 OTHER PROJECTIONS FOR $Q_s^h$ AND $K_s^h$

As mentioned in the Discussion (Section 7.4), we reproduce visualizations from Section 7.3 using different projection choices. Namely, we compute the SVD on  $K_s^h$  only in Figure 1 and Figure 3, and on  $Q_s^h$  only in Figure 2 and Figure 4.

The plots show that not only does the distribution used for the SVD drifts away from the origin along training, but also that the other distribution drifts away from the origin in an opposite direction. In other words, the singular components of each distribution are also relevant to describe the drift of the other distribution. Hence, Figure 1 and Figure 2 support our conclusion that the drift directions of keys and queries are aligned during training.

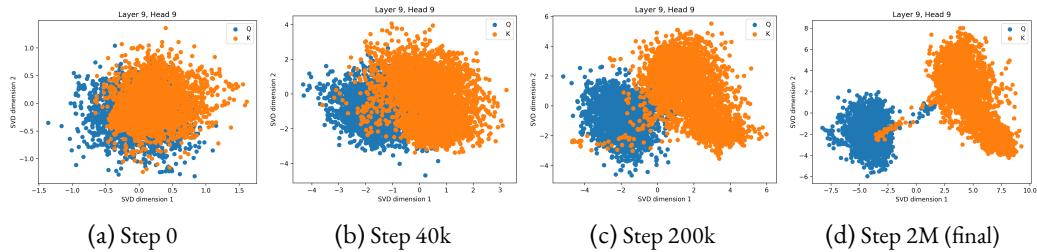


Figure 1: Evolution of  $Q_s^h$  and  $K_s^h$  distributions along training. Vectors are projected using the SVD computed on  $K_s^h$ .

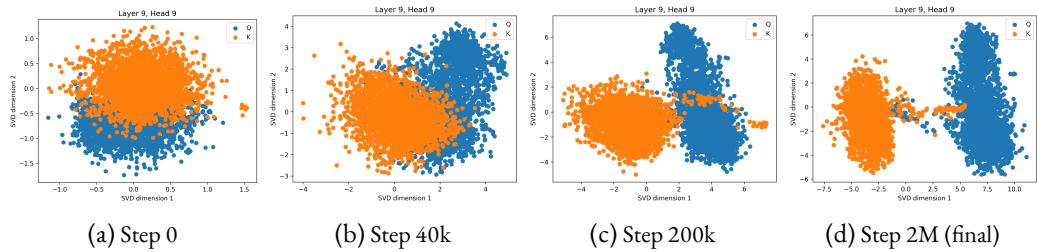


Figure 2: Evolution of  $Q_s^h$  and  $K_s^h$  distributions along training. Vectors are projected using the SVD computed on  $Q_s^h$ .

### A.2 STABILITY ACROSS MULTIBERT SEEDS

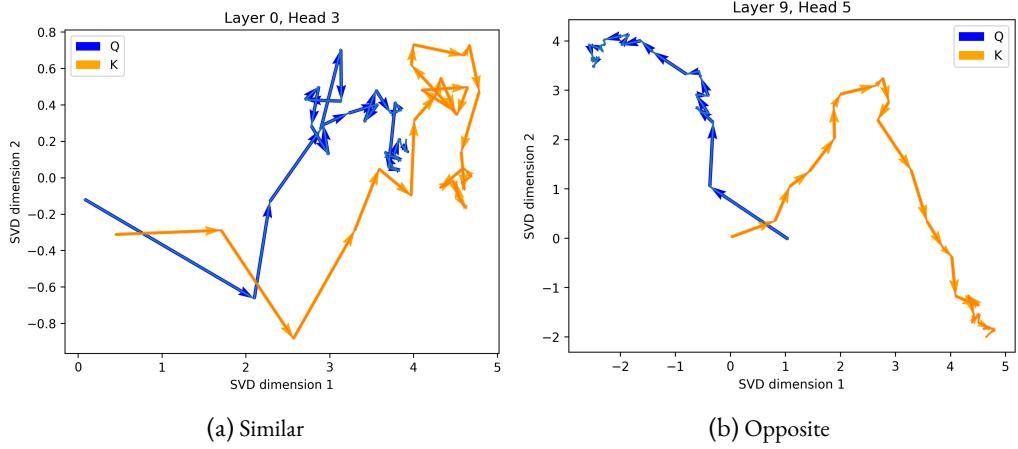


Figure 3: Evolution of  $\bar{Q}_s^h$  and  $\bar{K}_s^h$  along training for two different heads in the network, projected via the SVD of  $K_s^h$ .

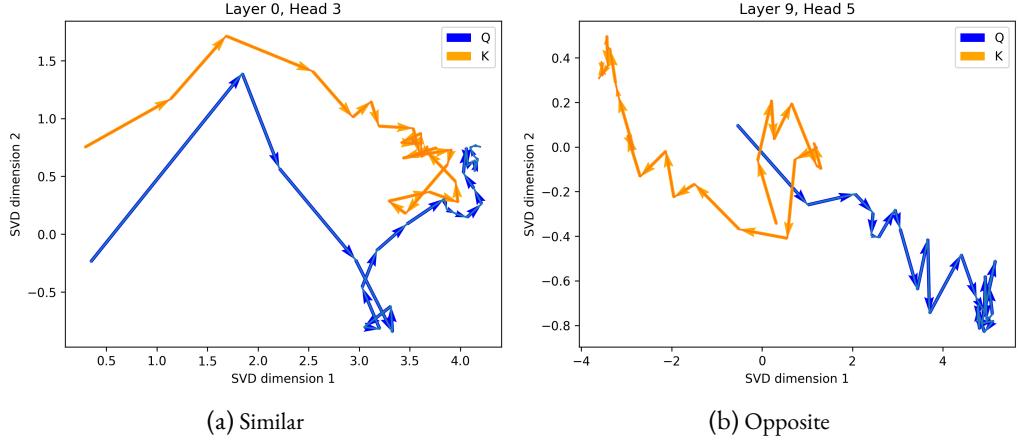


Figure 4: Evolution of  $\bar{Q}_s^h$  and  $\bar{K}_s^h$  along training for two different heads in the network, projected via the SVD of  $Q_s^h$ .

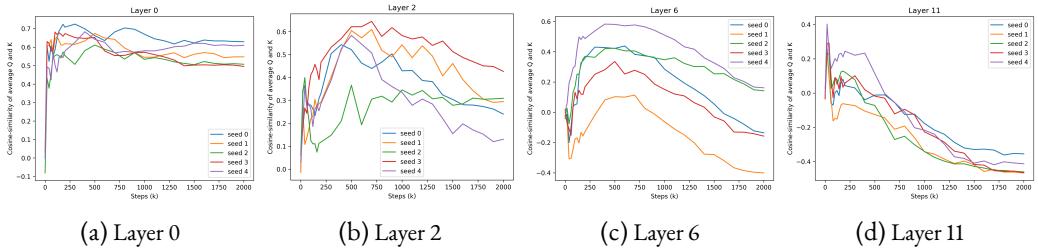


Figure 5: Evolution of cosine-similarity between  $\bar{Q}_s^h$  and  $\bar{K}_s^h$  along training for various initialization seeds. Representations are concatenated across heads, and each color represents one seed of the MultiBERT models. We observe similar trends across seeds.

## B PRETRAINING HYPERPARAMETERS FOR CHAPTER 8

### B.1 MONOLINGUAL ENCODERS

Dataset	OpenWebText2
Architecture	bert-base-uncased
Tokenizer	pythia-70m-deduped
Optimizer	AdamW
Learning rate	1e-4
Precision	16
Weight decay	0.01
Gradient clipping	1
Device batch size	32 / 64
Batch size	256 / 512
Sequence length	128
LR schedule	Triangular
Warmup steps	10000
Nb. steps	1000000

Table 1: Pre-training hyperparameters used for the monolingual encoders. When they differ between vanilla and headless models, we provide separate values formatted as (vanilla / headless). Model names written as `model-name` refer to their HuggingFace release.

### B.2 MONOLINGUAL DECODERS

Dataset	OpenWebText2
Architecture	pythia-70m-deduped
Tokenizer	pythia-70m-deduped
Optimizer	AdamW
Adam $\epsilon$	1e-8
Adam $(\beta_1, \beta_2)$	(0.9, 0.95)
Learning rate	1e-3
Precision	16
Weight decay	0.1
Gradient clipping	1
Device batch size	8 / 8
Batch size	1024 / 1024
Sequence length	2048
LR schedule	Cosine
Warmup steps	1430
Nb. steps	143000

Table 2: Pre-training hyperparameters used for the monolingual encoders. When they differ between vanilla and headless models, we provide separate values formatted as (vanilla / headless).

### B.3 MULTILINGUAL ENCODERS

Dataset	Wikipedia (multilingual)
Architecture	<code>distilbert-base-multilingual-cased</code>
Tokenizer	<code>distilbert-base-multilingual-cased</code>
Optimizer	AdamW
Learning rate	$2e-4$
Precision	16
Weight decay	0.01
Gradient clipping	1
Device batch size	64
Batch size	64
Sequence length	128
LR schedule	Triangular
Warmup steps	10000
Nb. steps	400000

Table 3: Pre-training hyperparameters used for the multilingual encoders.

### B.4 SMALL MONOLINGUAL ENCODERS

Dataset	CC-News
Architecture	<code>google/bert_uncased_L-4_H-512_A-8</code>
Tokenizer	<code>google/bert_uncased_L-4_H-512_A-8</code>
Optimizer	AdamW
Learning rate	$2e-4$
Precision	16
Weight decay	0.01
Gradient clipping	1
Device batch size	64
Batch size	64
Sequence length	128
LR schedule	Triangular
Warmup steps	10000
Nb. steps	400000

Table 4: Pre-training hyperparameters used for the small monolingual encoders used in [Figure 8.7](#).

## B.5 SMALL MONOLINGUAL DECODERS

Dataset	CC-News
Architecture	gpt2
Hidden size	192
Number heads	3
Number layers	3
Tokenizer	gpt2
Optimizer	AdamW
Learning rate	2.5e-4
Precision	16
Weight decay	0.01
Gradient clipping	1
Sequence length	128
LR schedule	Cosine
Warmup steps	2000
Nb. steps	1000000

Table 5: Pre-training hyperparameters used for the small monolingual decoders used in [Figure 8.8](#). These models rely on the GPT-2 architecture with a few changes. These changes scale down the model size to 11M parameters.

## C FINETUNING HYPERPARAMETERS FOR CHAPTER 8

### C.1 BALANCED CROSS-ENTROPY

Optimizer	AdamW
Learning rate	5e-6
Weight decay	0.01
Batch size	32
LR schedule	Constant
Linear warm-up	10%
Epochs	10

Table 6: Fine-tuning hyperparameters for monolingual encoder models trained with regular cross-entropy on the GLUE benchmark.

## C.2 MONOLINGUAL ENCODERS

Optimizer	AdamW
Learning rate	1e-5
Cross-entropy	Balanced
Weight decay	0
Batch size	32
LR schedule	Constant
Linear warm-up	10%
Epochs	10

Table 7: Fine-tuning hyperparameters for monolingual encoder models trained with balanced cross-entropy on the GLUE benchmark.

## C.3 MONOLINGUAL DECODERS

Dataset	OpenWebText2
Optimizer	AdamW
Learning rate	1e-5
Cross-entropy	Regular
Weight decay	0
Batch size	256
LR schedule	Constant
Linear warm-up	2000
Nb. steps	10000

Table 8: Fine-tuning hyperparameters for the headless monolingual decoder model using the causal language modeling objective.

## C.4 MULTILINGUAL ENCODERS

Optimizer	AdamW
Learning rate	2e-5
Cross-entropy	Regular
Weight decay	0
Batch size	128
LR schedule	Constant
Linear warm-up	10%

Table 9: Fine-tuning hyperparameters for the multilingual encoder models in Translate-Train and Translate-Test scenarios.

## D HYPERPARAMETERS FOR CHAPTER 9

Hyperparameter	MANTa <sub>Small</sub>	MANTa <sub>Base</sub>
Input Embeddings size	64	128
Num. layers	1	2
Num. heads	8	8
Attention window	16	16
Convolution kernel size	3	3

Table 10: Hyperparameters for MANTa

Hyperparameter	ByT5 <sub>Small</sub>	T5 <sub>Small</sub>	T5 <sub>Base</sub>
Hidden size	1472	512	768
Num. layers (encoder)	12	6	12
Num. layers (decoder)	4	6	12
Num. heads	6	8	12
Feed-forward dim.	3584	2048	3072
Dropout rate	0.1	0.1	0.1

Table 11: Hyperparameters for encoder-decoders