

static/media/sorbonne.pdf

static/media/inria.pdf

# UNIVERSITÉ SORBONNE UNIVERSITÉ

ECOLE DOCTORALE INFORMATIQUE, TÉLÉCOMMUNICATIONS ET ELECTRONIQUE - ED130

INRIA DE PARIS / ÉQUIPE ALMANACH

## THÈSE DE DOCTORAT

Discipline : Informatique

Présentée par

**Nathan GODEY**

Dirigée par

**Éric VILLEMONTÉ DE LA CLERGERIE et Benoît SAGOT**

Pour obtenir le grade universitaire de

DOCTEUR de l'UNIVERSITÉ SORBONNE UNIVERSITÉ

---

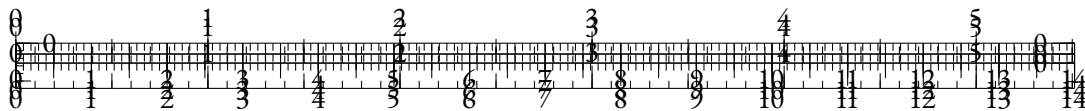
## Improving Representations for Language Modeling

---

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

Alonzo CHURCH	Princeton University	Examineur
Christopher COLUMBUS	Kingdom of Castile	Invited member
Margaret HAMILTON	University of Michigan	Rapporteur
Emmy NOETHER	Georg-August-Universität Göttingen	Examineur
Jürgen SCHMIDHUBER	IDSIA	Directeur
Jean LECUN	Facebook AI	Co-directeur
Claude SHANNON	MIT	Examineur
Alan TURING	Princeton University	Rapporteur





## ABSTRACT

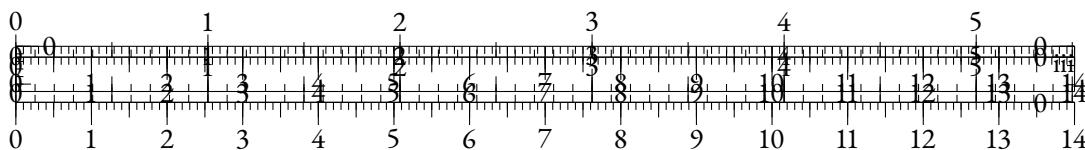
The field of Natural Language Processing has recently known a major paradigm shift that has led to significant improvements over the perceived capabilities of resulting systems. This shift, namely the advent of generative systems in the stead of predictive ones, has induced a profound change in the implicit 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.

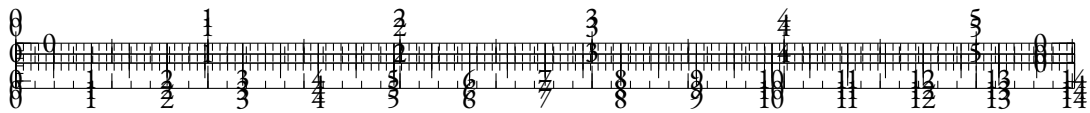
We find that the intermediate representations of self-supervised language models are affected by several forms of biases. First, they suffer from data-inherent biases that can be traced back to socio-cultural considerations, as we demonstrate by probing geographical knowledge in these features. Moreover, 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 [EDINBURGH PROJECT].

Overall, our work proves the relevance of representation analysis in the context of improving language systems.

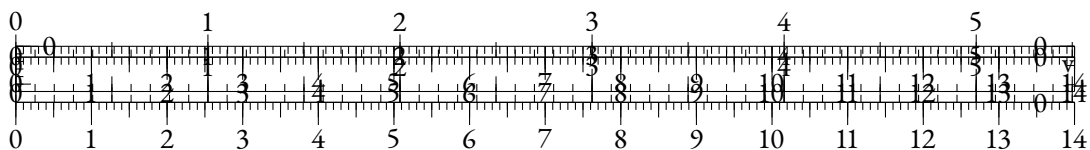


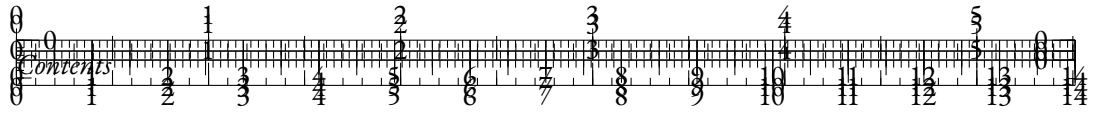




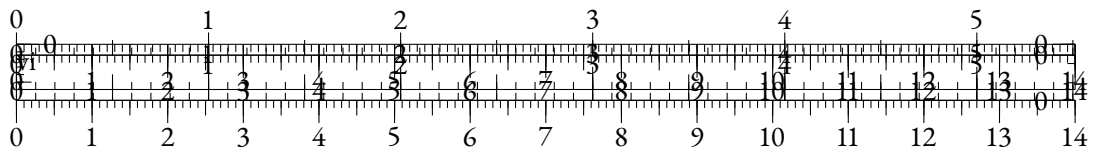
# CONTENTS

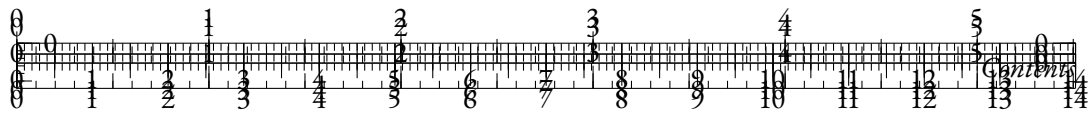
I	PRELIMINARY	1
1	INTRODUCTION	3
1.1	Background . . . . .	3
1.2	Motivation . . . . .	3
1.3	Scope . . . . .	4
1.4	Contributions . . . . .	5
2	RELATED WORKS	7
2.1	Representation Learning . . . . .	7
2.1.1	Introduction . . . . .	7
2.1.2	Statistical approaches . . . . .	7
2.1.3	Auto-encoders . . . . .	7
2.1.4	Contrastive approaches . . . . .	8
2.2	Language Modeling . . . . .	8
2.2.1	Introduction . . . . .	8
2.2.2	Methods . . . . .	9
2.2.3	Architectures . . . . .	10
2.2.4	Limitations . . . . .	11
2.3	Representation Analysis for NLP . . . . .	12
2.3.1	Representations and Linguistic Properties . . . . .	12
2.3.2	Analyzing Self-Attention . . . . .	13
2.3.3	Similarity and Geometry . . . . .	14
2.3.4	Representation Degeneration . . . . .	16
2.4	Beyond Classical Language Modeling . . . . .	17
2.4.1	Tokenizer-Free Language Modeling . . . . .	17
2.4.2	Efficient Attention . . . . .	18
2.4.3	Alternative Training Tasks & Objectives . . . . .	18
II	ANALYSIS OF THE REPRESENTATIONS OF LANGUAGE MODELS	21
2.5	Geographical . . . . .	23
2.5.1	Introduction & Related work . . . . .	23
2.5.2	Scaling Laws of Geographical Probing . . . . .	23
2.5.3	Geographical Bias and Scale . . . . .	25
2.5.4	Discussion . . . . .	29



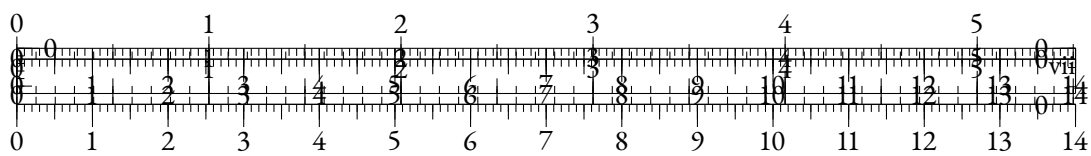


2.6	Softmax Bottleneck . . . . .	29
2.6.1	Introduction . . . . .	29
2.6.2	Related Works . . . . .	30
2.6.3	Language Model Saturation . . . . .	31
2.6.4	Performance Saturation is Rank Saturation . . . . .	32
2.6.5	The Softmax Bottleneck & Language Dimensionality . . . . .	34
2.6.6	Discussion . . . . .	39
2.6.1	Proofs . . . . .	41
2.6.2	Hyperparameters . . . . .	42
2.7	Anisotropy . . . . .	42
2.7.1	Introduction . . . . .	42
2.7.2	Related Work . . . . .	43
2.7.3	Anisotropy in pre-trained Transformers . . . . .	45
2.7.4	Exploring the representation drift . . . . .	49
2.7.5	Queries and keys: training dynamics . . . . .	50
2.7.6	Discussion . . . . .	52
2.7.7	Pearson correlation of the drift norm and anisotropy . . . . .	53
2.7.8	Cosine-similarity and anisotropy . . . . .	53
2.7.9	Other projections for $Q_s$ and $K_s$ . . . . .	54
2.7.10	Stability across MultiBERT seeds . . . . .	54
III	EXTENSIONS OF THE LANGUAGE MODELING PARADIGM . . . . .	65
2.8	Headless . . . . .	67
2.8.1	Introduction . . . . .	67
2.8.2	Related Work . . . . .	68
2.8.3	Method . . . . .	69
2.8.4	Experiments . . . . .	71
2.8.5	Multilingual Encoder . . . . .	74
2.8.6	Discussion . . . . .	75
2.8.1	Modeling considerations . . . . .	78
2.8.2	Limitations . . . . .	79
2.8.3	Ethics Statement . . . . .	79
2.8.4	Pretraining hyperparameters . . . . .	80
2.8.5	Finetuning hyperparameters . . . . .	82
2.8.6	Representing synonyms . . . . .	84
2.8.7	Implementation . . . . .	84
2.9	MANTa . . . . .	84
2.9.1	Introduction . . . . .	84
2.9.2	Related Work . . . . .	88
2.9.3	MANTa . . . . .	89
2.9.4	Experiments and Results . . . . .	93
2.9.5	Training Speedups . . . . .	95
2.9.6	Discussion . . . . .	95



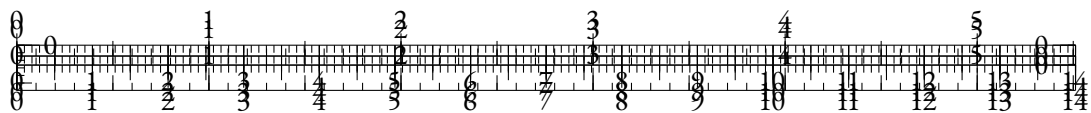


2.9.7	Conclusion . . . . .	97
2.9.1	Improving Pooling Speed . . . . .	98
2.9.2	Hyperparameters . . . . .	99
2.9.3	Additional results . . . . .	102
2.9.4	MANTa Module . . . . .	102





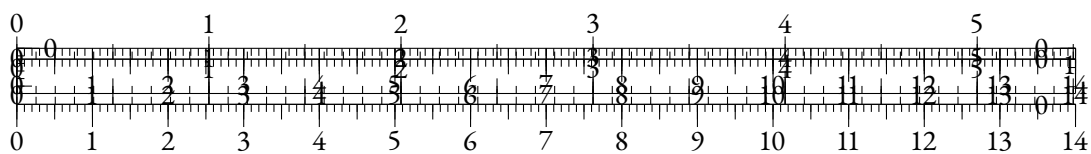




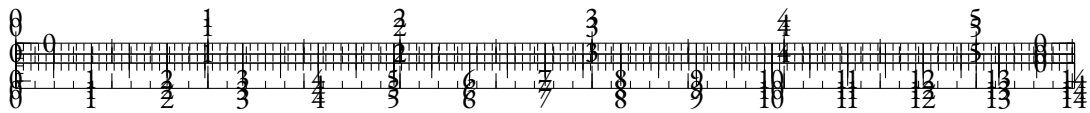
## PART I

### A GOOD PART

You can also use parts in order to partition your great work into larger ‘chunks’. This involves some manual adjustments in terms of the layout, though.







# 1 INTRODUCTION

In which the reasons for creating this package are laid bare for the whole world to see and we encounter some usage guidelines.

This package contains a minimal, modern template for writing your thesis. While originally meant to be used for a Ph. D. thesis, you can equally well use it for your honour thesis, bachelor thesis, and so on—some adjustments may be necessary, though.

## 1.1 BACKGROUND

I was not satisfied with the available templates for  $\text{\LaTeX}$  and wanted to heed the style advice given by people such as Robert Bringhurst (?) or Edward R. Tufte (??). While there *are* some packages out there that attempt to emulate these styles, I found them to be either too bloated, too playful, or too constraining. This template attempts to produce a beautiful look without having to resort to any sort of hacks. I hope you like it.

## 1.2 MOTIVATION

The package tries to be easy to use. If you are satisfied with the default settings, just add

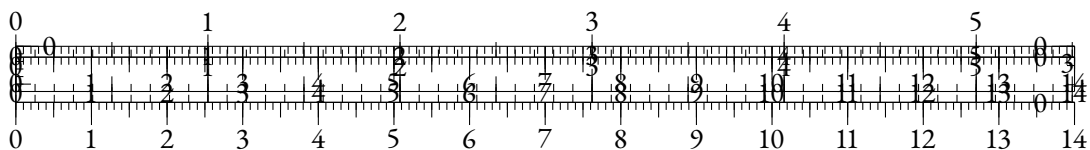
```
\documentclass{mimosis}
```

at the beginning of your document. This is sufficient to use the class. It is possible to build your document using either  $\text{\LaTeX}$ ,  $\text{X}\text{\LaTeX}$ , or  $\text{Lua}\text{\LaTeX}$ . I personally prefer one of the latter two because they make it easier to select proper fonts.

Prior to using this template, the first thing you want to do is probably a little bit of customisation. You can achieve quick changes in look and feel by picking your own fonts. With the `fontspec` package loaded and  $\text{X}\text{\LaTeX}$  or  $\text{Lua}\text{\LaTeX}$  as your compiler, this is pretty simple:

```
\setmainfont{Your main font}
\setsansfont{Your sans-serif font}
\setmonofont{Your monospaced font}
```

Make sure to select nice combinations of that are pleasing to *your* eyes—this is your document and it should reflect your own style. Make sure to specify font names as they are provided by your system. For instance, you might want to use the following combination:



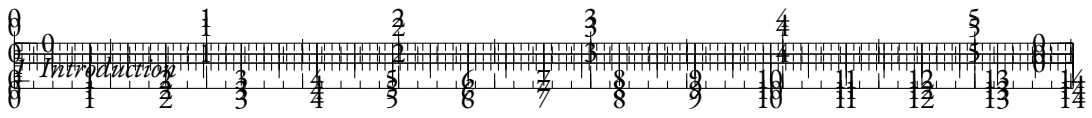


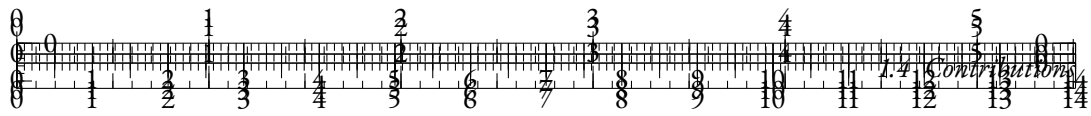
Table 1.1: A list of the most relevant packages required (and automatically imported) by this template.

You can also remove the `scale` directive, but I find that most fonts pair better if they are adjusted in size a little bit. Experiment with it until you find a combination that you enjoy.

Along with the standard environments, this template offers `paralist` for lists within paragraphs. Here's a quick example: The American constitution speaks, among others, of (i) life (ii) liberty (iii) the pursuit of happiness. These should be added in equal measure to your own conduct. To typeset units correctly, use the `siunitx` package. For example, you might want to restrict your daily intake of liberty to 750 mg.

### 1.3 SCOPE

What we wanted to do.

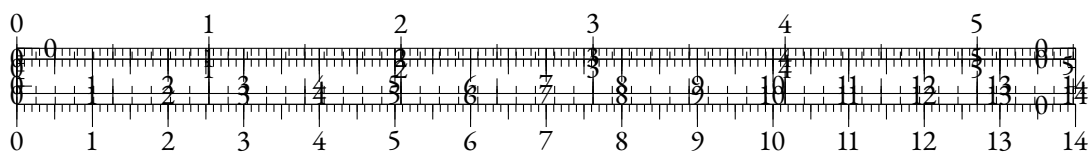


## 1.4 CONTRIBUTIONS

Since this class heavily relies on the `scrbook` class, you can use *their* styling commands in order to change the look of things. For example, if you want to change the text in sections to **bold** you can just use

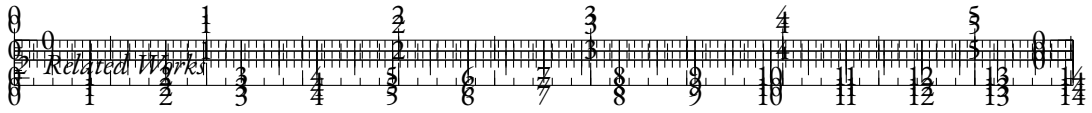
```
\setkomafont{sectioning}{\normalfont\bfseries}
```

at the end of the document preamble—you don't have to modify the class file for this. Please consult the source code for more information.









**Basic Auto-Encoders:** The simplest form of auto-encoders involves a single hidden layer that compresses the input into a lower-dimensional latent space. The model is trained to minimize the reconstruction error between the input and the output.

**Variational Auto-Encoders (VAEs):** VAEs extend basic auto-encoders by imposing a probabilistic structure on the latent space. They use a probabilistic encoder to map inputs to a distribution in the latent space, allowing for the generation of new samples by sampling from this distribution. VAEs are useful in tasks requiring generative capabilities, such as text generation.

**Denoising Auto-Encoders (DAEs):** DAEs are trained to reconstruct the original data from corrupted versions. This process encourages the model to learn robust features that are invariant to noise, improving the quality of the learned representations.

#### 2.1.4 CONTRASTIVE APPROACHES

Contrastive approaches in representation learning aim to learn effective embeddings by contrasting positive and negative examples. The core idea is to bring similar items closer together in the embedding space while pushing dissimilar items apart. These methods are essential for capturing nuanced relationships in the data and enhancing the quality of learned representations.

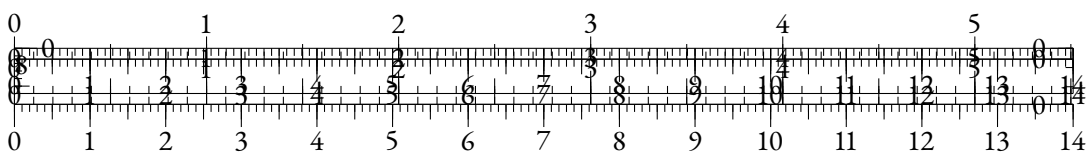
- **Contrastive Loss:** The fundamental concept in contrastive learning is the contrastive loss function, which drives the learning process by encouraging the model to distinguish between positive pairs (similar items) and negative pairs (dissimilar items).
- **Triplet Loss:** Triplet loss is a popular contrastive learning technique that uses triplets of samples: an anchor, a positive (similar to the anchor), and a negative (dissimilar to the anchor). The objective is to minimize the distance between the anchor and the positive while maximizing the distance between the anchor and the negative. This approach is widely used in tasks such as face recognition and text similarity.
- **Noise Contrastive Estimation (NCE):** NCE is another contrastive learning method that reformulates the problem of estimating a probability distribution into a binary classification problem. The model learns to distinguish between observed data and artificially generated noise samples. NCE is particularly useful in large-scale language models where direct computation of probabilities is computationally expensive.

## 2.2 LANGUAGE MODELING

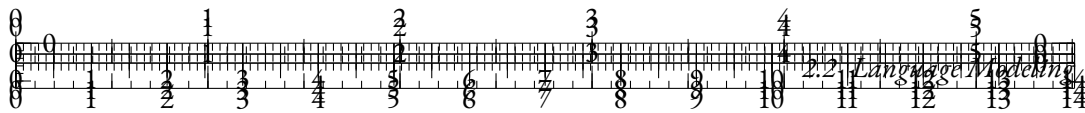
### 2.2.1 INTRODUCTION

Language modeling is a fundamental task in natural language processing (NLP) that involves predicting the next word in a sequence given the preceding context. This task is crucial for various applications, including speech recognition, machine translation, text generation, and more. Language models capture the probability distribution of word sequences, enabling them to generate coherent and contextually appropriate text.

Historically, language models relied on n-gram approaches, which predict a word based on the previous n-1 words. However, these models faced limitations in handling long-range dependencies







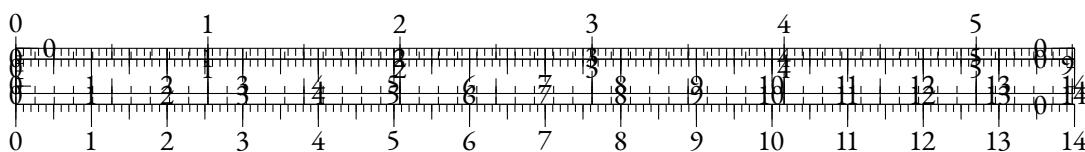
and sparsity issues. With the advent of deep learning, neural language models have significantly advanced the field, providing more powerful and flexible approaches to capturing language patterns.

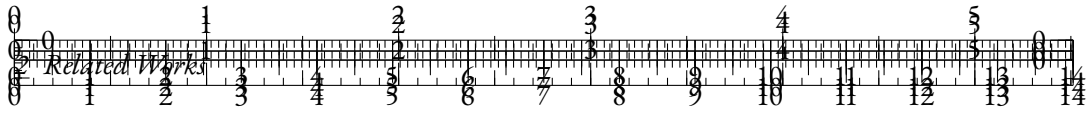
## 2.2.2 METHODS

The process of training a language model involves several key steps, from preparing the text data to optimizing the model using specific objectives. Here is a simplified pipeline:

- **Tokenization:** The first step in building a language model is tokenization, which involves breaking down the text into smaller units called tokens. Tokens can be words, subwords, or characters. This step converts raw text into a format that the model can process, typically resulting in a sequence of token indices.
- **Embedding:** Once the text is tokenized, each token is mapped to a continuous vector representation, known as an embedding. These embeddings capture semantic information about the tokens and are learned during training.
- **Contextualization:** The model processes the sequence of token embeddings to capture the contextual relationships between tokens. This involves passing the embeddings through layers of neural networks that refine the representations based on the surrounding context.
- **Prediction:** For each position in the sequence, the model predicts the probability distribution of the next token. This step involves transforming the contextualized representations into logits, which are unnormalized scores for each token in the vocabulary.
- **Objective Function:** The model is trained to minimize a specific objective function that measures the difference between the predicted probabilities and the actual next token. The most common objective function for language modeling is cross-entropy loss, which quantifies the accuracy of the model's predictions.
- **Contrastive Methods:** In addition to traditional cross-entropy loss, contrastive methods can be used to improve the quality of the learned representations. These methods involve creating pairs of similar and dissimilar examples and training the model to distinguish between them, enhancing the model's ability to capture nuanced relationships in the data.
- **Regularization:** To prevent overfitting and improve generalization, regularization techniques are applied during training. Common regularization methods include dropout (randomly dropping units during training to prevent co-adaptation), weight decay (penalizing large weights), and data augmentation (creating variations of the training data to improve robustness).

In summary, the objective pipeline for training a language model involves tokenizing the text, embedding the tokens, capturing contextual relationships, making predictions, and optimizing the model using objective functions like cross-entropy and contrastive loss, along with regularization techniques to ensure robust and effective learning.

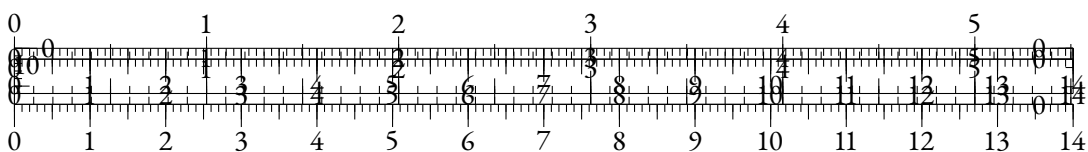


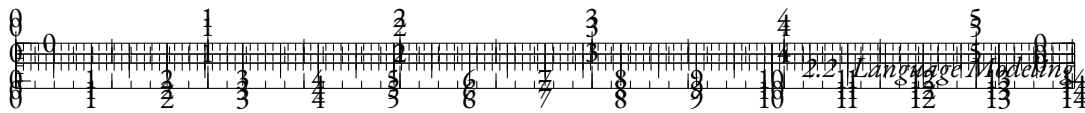


### 2.2.3 ARCHITECTURES

The architecture of a language model significantly influences its performance and capabilities. Key architectures include Recurrent Neural Networks (RNNs) and Transformers, each with unique mechanisms for processing sequences.

- **Recurrent Neural Networks (RNNs):** RNNs process sequences one element at a time, maintaining a hidden state that captures information from previous steps. Variants like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs) include gating mechanisms to mitigate issues like vanishing and exploding gradients. These gates control the flow of information, allowing the network to capture long-range dependencies more effectively.
- **Transformers:** Transformers use self-attention mechanisms to weigh the importance of different words in a sequence relative to each other. They are composed of an encoder and a decoder:
  - **Encoder:** The encoder consists of multiple layers, each containing two main components: a multi-head self-attention mechanism and a position-wise feedforward neural network. The self-attention mechanism allows the model to consider all positions in the input sequence simultaneously, capturing dependencies regardless of their distance.
  - **Decoder:** The decoder also consists of multiple layers, with three main components: a masked multi-head self-attention mechanism, an encoder-decoder attention mechanism, and a position-wise feedforward neural network. The masked self-attention ensures that each position can only attend to earlier positions, maintaining the autoregressive property during generation.
- **Attention Mechanisms:** The attention mechanisms in transformers come in two forms:
  - **Masked Attention:** Used in the decoder, masked attention ensures that the model cannot attend to future tokens, preserving the causality needed for autoregressive tasks like text generation.
  - **Causal Attention:** Similar to masked attention, causal attention restricts the attention to past and present tokens only, which is essential for maintaining the correct sequence of predictions.
  - **Self-Attention:** Used in both the encoder and decoder, self-attention allows each token to attend to all other tokens in the sequence, capturing global dependencies and contextual information.
- **Bidirectional vs. Unidirectional Models:**
  - **Bidirectional Models:** Examples include BERT (Bidirectional Encoder Representations from Transformers), which attends to both past and future contexts in the input sequence. This is useful for tasks requiring comprehensive context understanding, such as question answering and sentiment analysis.





- **Unidirectional Models:** Examples include GPT (Generative Pre-trained Transformer), which attends only to past tokens. This autoregressive approach is particularly effective for text generation tasks.
- **Encoder-Decoder Models:** Models like T5 (Text-to-Text Transfer Transformer) utilize both encoder and decoder structures. The encoder processes the input sequence into a context-rich representation, which the decoder then uses to generate the output sequence. This architecture is versatile, handling a wide range of text-to-text tasks under a unified framework.

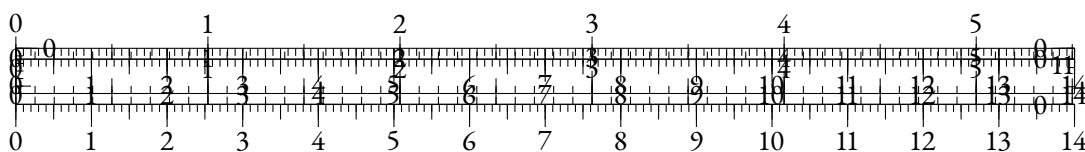
In summary, the architecture of language models ranges from RNNs that process sequences sequentially to transformers that leverage self-attention for capturing dependencies across entire sequences. These architectures, with their various attention mechanisms and structural differences, enable powerful and flexible modeling of natural language.

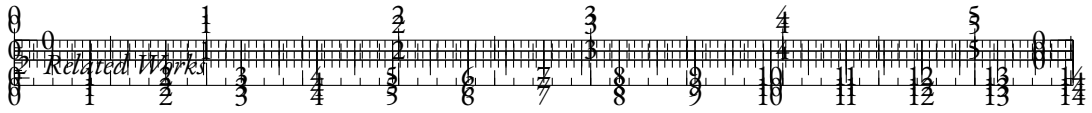
#### 2.2.4 LIMITATIONS

Despite their advancements, language models face several limitations:

- **Data and Computation Requirements:** Training state-of-the-art language models requires vast amounts of data and computational resources. This limitation restricts access to such models to organizations with substantial resources and makes the training process energy-intensive.
- **Bias and Fairness:** Language models can learn and perpetuate biases present in the training data, leading to biased and potentially harmful outputs. Addressing these biases is a critical area of ongoing research, as it impacts the fairness and ethical use of NLP systems.
- **Context Length:** While transformers have improved the handling of long-range dependencies, they are still limited by the maximum input length they can process. Techniques like segment-level recurrence or hierarchical models are being explored to address this limitation.
- **Interpretability:** Deep neural language models are often seen as black boxes, making it challenging to understand and interpret their predictions. Enhancing the interpretability of these models is essential for building trust and ensuring their safe application.
- **Generalization:** Language models sometimes struggle to generalize to out-of-distribution examples or novel contexts not seen during training. Ensuring robust generalization remains an important challenge, particularly for applications in dynamic or unpredictable environments.

In summary, while language models have made significant strides in NLP, they face notable challenges related to data and computation requirements, bias and fairness, context length, interpretability, and generalization. Addressing these limitations is crucial for the continued advancement and ethical deployment of NLP technologies.



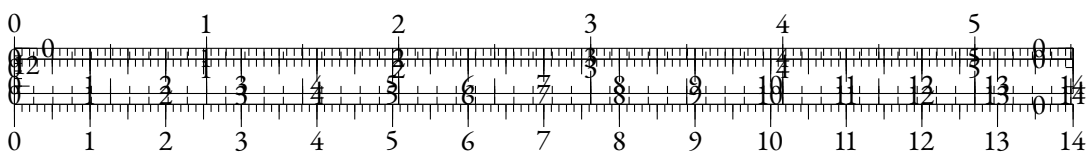


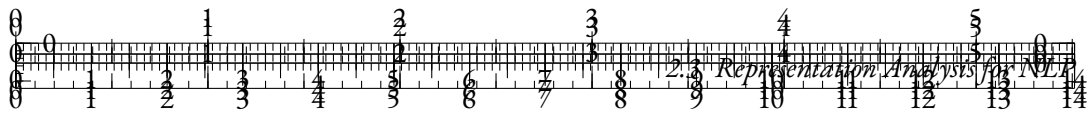
## 2.3 REPRESENTATION ANALYSIS FOR NLP

### 2.3.1 REPRESENTATIONS AND LINGUISTIC PROPERTIES

Representations learned by NLP models capture various linguistic properties that are essential for understanding language. These properties can be broadly categorized into syntactic and semantic information.

- **Syntactic Properties:** Syntactic properties refer to the structural aspects of language, including grammar and sentence structure. Effective representations encode the following syntactic information:
  - **Part-of-Speech Tags:** Representations can capture the grammatical categories of words, such as nouns, verbs, adjectives, etc. This helps in understanding the roles that words play in sentences.
  - **Dependency Relations:** Words in a sentence often have grammatical dependencies with each other (e.g., subject-verb, adjective-noun). Representations that capture these dependencies can help in tasks like parsing and syntax-based translation.
  - **Constituent Structure:** Representations may also encode higher-level syntactic structures, such as phrases and clauses, which are important for understanding the hierarchical organization of sentences.
  - **Word Order:** The sequence in which words appear in a sentence is crucial for meaning. Representations that preserve word order information can help in tasks like machine translation and text generation.
- **Semantic Properties:** Semantic properties involve the meanings of words and their relationships. Effective representations capture the following semantic information:
  - **Word Meanings:** Representations encode the meanings of individual words. This can be analyzed through tasks like word similarity, where similar words (e.g., "cat" and "feline") have similar embeddings.
  - **Contextual Meaning:** The meaning of a word can change based on its context. Contextual embeddings, such as those from models like BERT, capture these nuances by considering surrounding words. For example, the word "bank" has different meanings in "river bank" and "financial bank".
  - **Synonymy and Antonymy:** Effective representations capture semantic relationships such as synonyms (words with similar meanings) and antonyms (words with opposite meanings). This is crucial for tasks like paraphrase detection and sentiment analysis.
  - **Polysemy:** Words with multiple meanings (polysemous words) should have representations that reflect their different senses depending on context. For instance, "bark" should have different embeddings when referring to a tree's outer layer versus a dog's sound.
  - **Compositionality:** The meaning of phrases and sentences is often compositional, meaning it is derived from the meanings of individual words and their arrangement.





Representations that capture compositionality help in understanding complex expressions and idiomatic phrases.

To analyze these properties, various probing techniques are used:

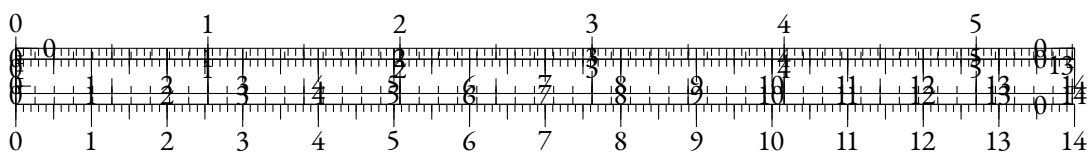
- **Probing Classifiers:** Small supervised classifiers are trained on top of fixed embeddings to predict linguistic properties like part-of-speech tags, syntactic roles, or semantic roles. High accuracy indicates that the embeddings capture relevant linguistic information.
- **Visualization:** Techniques such as t-SNE or PCA can visualize the high-dimensional embeddings in a lower-dimensional space, helping to inspect clusters and relationships between words.
- **Correlation Analysis:** Correlating embedding distances with human-judged linguistic distances (e.g., similarity or relatedness scores) can provide insights into how well the representations capture semantic relationships.
- **Linguistic Tasks:** Evaluating representations on downstream linguistic tasks, such as named entity recognition, sentiment analysis, or syntactic parsing, provides practical evidence of the embeddings' effectiveness in capturing linguistic properties.

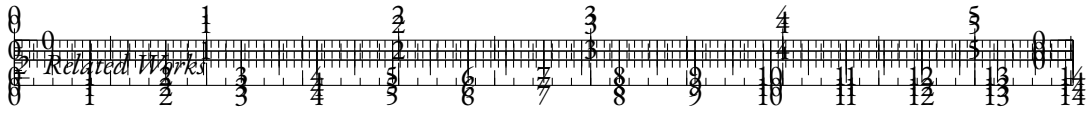
In summary, representations learned by NLP models encapsulate both syntactic and semantic properties of language. Analyzing these representations through various probing techniques helps in understanding their effectiveness and guiding further improvements in model design and training.

### 2.3.2 ANALYZING SELF-ATTENTION

Self-attention mechanisms in transformer models allow for the examination of how tokens attend to each other, providing insights into the model's internal workings. Analyzing self-attention helps to understand what information the model considers important and how it processes different parts of the input sequence.

- **Attention Patterns:** By visualizing attention weights, we can analyze how the model distributes attention across different tokens. Attention patterns reveal which tokens are considered relevant for predicting the next token in a sequence. Typical visualization techniques include attention heatmaps and attention heads visualizations. These patterns can show whether the model focuses on nearby words, distant words, or specific syntactic structures.
- **Head Specialization:** Transformers use multi-head self-attention mechanisms, where multiple attention heads operate in parallel. Each head can learn to focus on different types of relationships or aspects of the input. Analyzing head specialization involves examining the distinct roles of each attention head. For example, some heads might specialize in capturing syntactic dependencies (like subject-verb relationships), while others might focus on semantic roles (like identifying entities and their attributes).





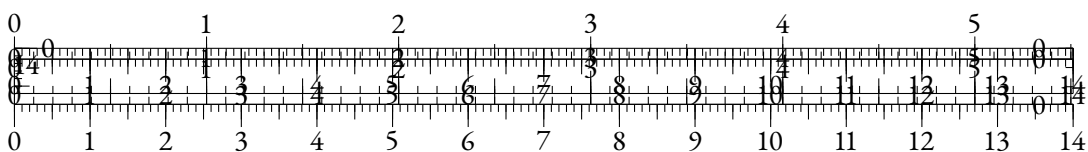
- **Layer-wise Analysis:** Self-attention can be analyzed at different layers of the transformer. Lower layers often capture more local and syntactic information, while higher layers tend to capture more global and semantic information. Layer-wise analysis helps in understanding the hierarchical nature of learned representations and how information is progressively abstracted.
- **Global vs. Local Attention:** Analyzing whether the model's attention is more global (considering distant tokens) or local (focusing on nearby tokens) helps in understanding its contextual understanding. For instance, attention to distant tokens can indicate the model's ability to capture long-range dependencies.
- **Attention as Explanation:** Attention weights are sometimes used as explanations for model predictions. However, it is important to note that while attention provides some interpretability, it is not a definitive explanation of model behavior. Additional analysis and methods are often needed to fully understand the model's decision-making process.

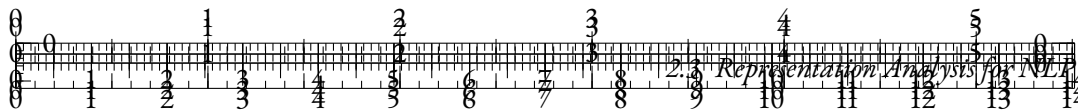
In summary, analyzing self-attention mechanisms in transformer models provides valuable insights into how these models process and prioritize different parts of the input sequence. Visualization and interpretation of attention patterns, head specialization, and layer-wise behavior help in understanding the internal workings of the model and improving its performance.

### 2.3.3 SIMILARITY AND GEOMETRY

The geometric properties of the learned representations provide valuable insights into the structure and effectiveness of the embedding space. Understanding similarity and geometry is crucial for evaluating how well the model captures relationships between words and phrases.

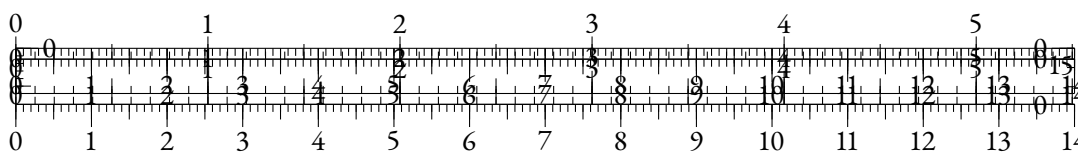
- **Similarity Metrics:** Analyzing similarity metrics helps in understanding how close or distant different word embeddings are within the vector space.
  - **Cosine Similarity:** This metric measures the cosine of the angle between two vectors, indicating how similar they are in terms of direction. High cosine similarity between embeddings suggests that the words are semantically similar.
  - **Euclidean Distance:** This metric measures the straight-line distance between two points in the embedding space. Smaller distances indicate greater similarity. While less commonly used than cosine similarity, it can provide additional insights into the embedding space's structure.
- **Clustering:** Grouping similar word embeddings together can reveal natural clusters within the embedding space.
  - **K-Means Clustering:** This algorithm partitions the embedding space into  $k$  clusters, where each word belongs to the cluster with the nearest mean. This can reveal semantic groupings, such as synonyms or related concepts.
  - **Hierarchical Clustering:** This method builds a hierarchy of clusters, which can be visualized as a dendrogram. It provides a more detailed view of the relationships between embeddings at different levels of granularity.

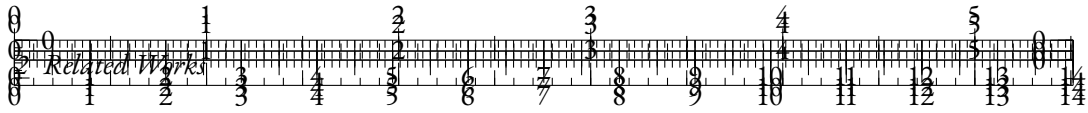




- **Dimensionality Reduction:** Visualizing high-dimensional embeddings in a lower-dimensional space can help in understanding their geometric properties.
  - **t-SNE (t-Distributed Stochastic Neighbor Embedding):** This technique reduces the dimensionality of embeddings while preserving local structures, making it useful for visualizing clusters and relationships.
  - **PCA (Principal Component Analysis):** PCA reduces the dimensionality by projecting the embeddings onto the directions of maximum variance. This helps in identifying the principal components that capture most of the variance in the data.
- **Embedding Space Geometry:** Studying the geometric properties of the embedding space provides insights into how well the model organizes linguistic information.
  - **Density and Distribution:** Analyzing the density and distribution of embeddings can reveal whether the space is uniformly populated or contains sparse regions. A well-distributed space indicates good coverage of the language.
  - **Subspace Structures:** Identifying subspaces within the embedding space that correspond to specific linguistic features (e.g., tense, number, or gender) can provide insights into how these features are encoded. For example, certain directions in the embedding space may correspond to semantic shifts like singular to plural forms.
- **Analogies and Linear Relationships:** Embeddings often capture analogical relationships through linear transformations. For instance, the relationship between "king" and "queen" can be similar to the relationship between "man" and "woman."
  - **Word Analogies:** By performing vector arithmetic (e.g., "king" - "man" + "woman"), one can retrieve vectors close to the expected answer (e.g., "queen"). This demonstrates the model's ability to capture meaningful relationships.
  - **Linear Projections:** Identifying and interpreting linear projections that correspond to specific semantic or syntactic properties can help in understanding the embedding space's structure. For example, projecting embeddings onto the gender subspace can reveal gender biases.
- **Intrinsic Evaluation Tasks:** These tasks evaluate the quality of word embeddings based on their geometric properties.
  - **Word Similarity Tasks:** These tasks measure how well the similarity between word embeddings aligns with human-judged similarities. Common datasets include WordSim-353 and SimLex-999.
  - **Word Analogies Tasks:** These tasks evaluate the model's ability to solve analogy problems, such as "man is to king as woman is to ?". The accuracy in these tasks reflects the model's capability to capture linear relationships.

In summary, analyzing the similarity and geometry of learned representations involves examining similarity metrics, clustering, dimensionality reduction, and intrinsic evaluation tasks. These analyses provide insights into the structure of the embedding space and the quality of the captured linguistic relationships.

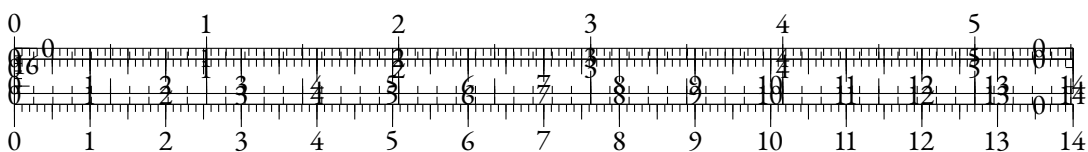




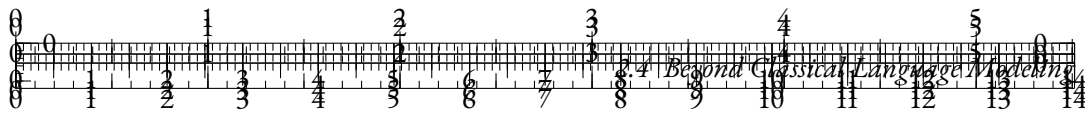
#### 2.3.4 REPRESENTATION DEGENERATION

Representation degeneration refers to geometric issues in learned embeddings where the embeddings lose their discriminative power and become less effective at capturing meaningful distinctions. This can manifest in several geometric phenomena:

- **Anisotropy:** Anisotropy in the embedding space occurs when the space is stretched or distorted in specific directions, causing the representations to be unevenly distributed. In the context of language models, anisotropic spaces can result in embeddings that are more spread out in certain dimensions while being compressed in others. This can affect the model's ability to capture and differentiate between various semantic and syntactic features.
- **Outlier Dimensions:** Outlier dimensions are directions in the embedding space that do not capture meaningful information and may represent noise or irrelevant features. These dimensions can distort the embeddings, leading to poor performance on tasks that rely on accurate semantic and syntactic understanding. Identifying and addressing outlier dimensions is essential for improving the quality of embeddings.
- **Representation Collapse:** Representation collapse refers to the phenomenon where embeddings of different tokens become indistinguishable and collapse into a narrow subspace. This often occurs when embeddings lose their diversity and become too similar to each other. Representation collapse reduces the model's ability to differentiate between tokens, adversely affecting downstream task performance. It can be detected by analyzing the clustering of embeddings or by examining their distribution.
- **Biases in Latent Spaces:** Embedding spaces can encode biases present in the training data, leading to undesirable biases in the latent space. For instance, gender, race, or cultural biases can manifest in specific dimensions, causing the model to make biased predictions or generate unfair outputs. Analyzing the latent space for biased subspaces or skewed distributions is crucial for addressing fairness issues in NLP models. Techniques such as adversarial debiasing or fair representation learning can help mitigate these biases.
- **Dimensional Collapse:** Dimensional collapse occurs when the embeddings occupy only a small subset of the available dimensions, effectively reducing the dimensionality of the learned representations. This can happen due to overfitting or excessive regularization, leading to embeddings that do not utilize the full capacity of the vector space. Analyzing the effective dimensionality and ensuring that embeddings utilize the available space can help in mitigating this issue.
- **Loss of Geometric Structure:** Effective embeddings should maintain meaningful geometric relationships, such as clustering of similar words and separation of dissimilar ones. Degeneration can lead to a loss of these geometric structures, where embeddings fail to reflect semantic or syntactic relationships accurately. Techniques like manifold learning or embedding space visualization can be used to analyze and restore geometric properties.
- **Regularization and Hyperparameter Tuning:** Overly aggressive regularization can cause embeddings to collapse into subspaces, while insufficient regularization might lead to







overfitting. Proper tuning of regularization parameters and hyperparameters is crucial for maintaining the balance between effective representation learning and avoiding degeneration.

**Mitigating Geometric Degeneration** involves several strategies:

- **Enhanced Training Techniques:** Using more diverse and extensive datasets can help the model learn richer representations and prevent degeneration. Techniques such as data augmentation and noise injection can also improve the robustness of embeddings.
- **Dimensionality Analysis:** Analyzing and managing the dimensionality of the embedding space helps in ensuring that the model makes full use of the available dimensions. Methods like Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) can help in identifying and addressing dimensional collapse.
- **Bias Mitigation Techniques:** Applying techniques like adversarial training, counterfactual data augmentation, and fairness constraints can help in reducing biases in the latent space and ensuring more equitable representations.
- **Regularization Techniques:** Applying appropriate regularization techniques to prevent overfitting and ensure embeddings retain their discriminative power. Techniques like weight decay, dropout, and layer normalization should be carefully tuned.
- **Model Architecture Adjustments:** Modifying the model architecture to increase capacity or adjust attention mechanisms can help in learning better representations and addressing geometric issues. For example, increasing the number of attention heads or layers can enhance the model's ability to capture complex relationships.

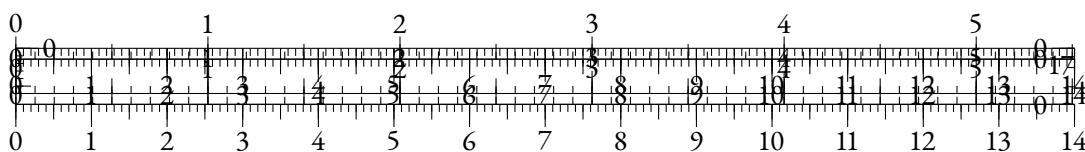
In summary, representation degeneration in the geometric sense involves anisotropy, outlier dimensions, representation collapse, and biases in latent spaces. Addressing these issues is crucial for maintaining the effectiveness of learned embeddings and ensuring that they provide accurate and fair representations in NLP models.

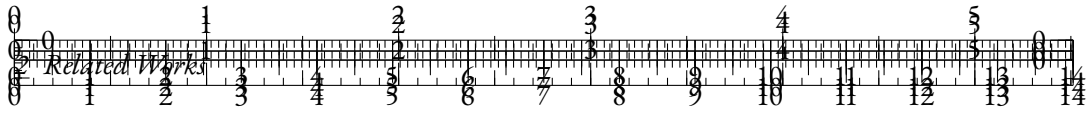
## 2.4 BEYOND CLASSICAL LANGUAGE MODELING

This section explores advancements and innovations that extend beyond traditional language modeling approaches. These innovations aim to improve various aspects of model performance, efficiency, and flexibility.

### 2.4.1 TOKENIZER-FREE LANGUAGE MODELING

Traditional language models rely heavily on tokenization to convert text into discrete units that the model processes. However, recent approaches are moving towards tokenizer-free language modeling, which seeks to bypass or minimize the need for tokenization.





- **Direct Subword Encoding:** Some models directly operate on raw text by learning representations at the character or subword level, eliminating the need for predefined tokenization schemes. This can improve handling of rare or out-of-vocabulary words and reduce preprocessing complexity.
- **Byte-Level Models:** Byte-level language models work directly with raw byte sequences, which allows them to handle any character set and avoid the limitations of fixed vocabulary sizes. These models learn to process text without the need for explicit tokenization, enabling more flexible and universal text representation.
- **End-to-End Models:** End-to-end models process text directly from input to output, bypassing intermediate tokenization steps. These models can be more adaptable and efficient in certain applications, such as real-time text generation or language understanding tasks where tokenization might introduce latency.

#### 2.4.2 EFFICIENT ATTENTION

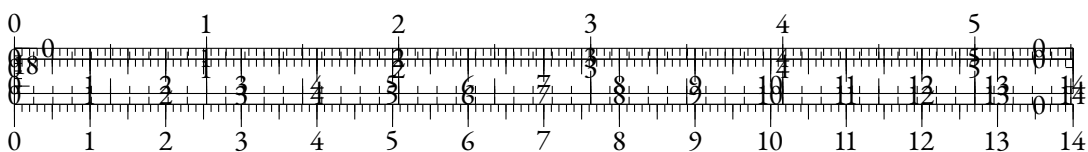
Efficient attention mechanisms aim to address the computational and memory inefficiencies associated with traditional self-attention mechanisms, especially in large-scale models.

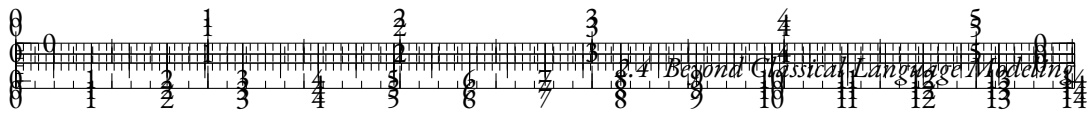
- **Sparse Attention:** Sparse attention mechanisms focus on attending to a subset of tokens rather than the entire sequence. Techniques like local attention, where only nearby tokens are attended to, and global attention, where certain tokens receive full attention, reduce the complexity and improve efficiency.
- **Low-Rank Approximations:** These methods approximate the attention matrix with low-rank representations, reducing computational and memory requirements. Approximations like LinFormer and Performer utilize mathematical techniques to simplify the attention computation.
- **Memory-Augmented Attention:** Memory-augmented attention mechanisms introduce external memory structures to store and retrieve information, reducing the need to compute attention over the entire sequence. This can be particularly useful for long sequences or tasks requiring long-term dependencies.
- **Kernel-Based Attention:** Kernel-based attention techniques approximate the attention mechanism using kernel functions, which can significantly speed up computations. Examples include the Reformer and LinFormer models, which utilize kernel-based methods to achieve linear time complexity.

#### 2.4.3 ALTERNATIVE TRAINING TASKS & OBJECTIVES

Alternative training tasks and objectives explore new ways to train language models beyond traditional language modeling objectives, such as predicting the next token.

- **Contrastive Learning:** Contrastive learning techniques train models by contrasting positive and negative examples, improving the quality of learned representations. Approaches

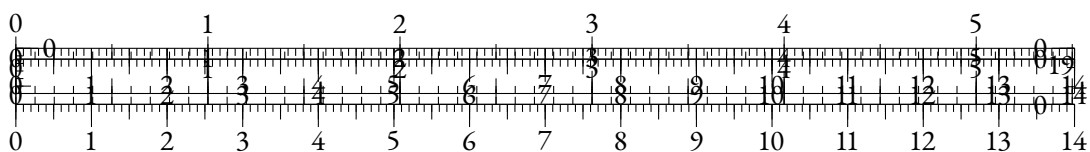




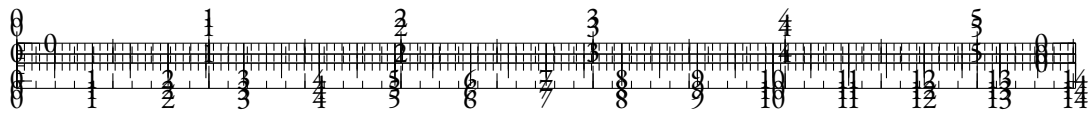
like SimCLR or MoCo adapt this framework to NLP by learning embeddings that are close for similar examples and far apart for dissimilar ones.

- **Multi-Task Learning:** Multi-task learning involves training a model on multiple tasks simultaneously, enabling it to generalize better across different domains. By sharing representations across tasks, models can leverage complementary information and improve performance on each individual task.
- **Masked Language Modeling (MLM):** MLM involves masking portions of the input text and training the model to predict the masked tokens. This approach, used in models like BERT, helps the model learn bidirectional context and capture more nuanced language understanding.
- **Denoising Autoencoders:** Denoising autoencoders are trained to reconstruct corrupted input text. This training objective helps the model learn robust representations by focusing on recovering clean text from noisy or incomplete inputs, which improves generalization to various tasks.
- **Self-Supervised Objectives:** Self-supervised learning tasks generate supervisory signals from the data itself, reducing the need for labeled examples. Tasks like next-sentence prediction, sentence permutation, and sentence similarity help models learn useful features from raw text data.

In summary, the advancements beyond classical language modeling include tokenizer-free approaches that simplify text processing, efficient attention mechanisms that address computational challenges, and alternative training tasks and objectives that enhance model learning and performance. These innovations contribute to more effective and adaptable language models, pushing the boundaries of what is achievable with NLP technologies.



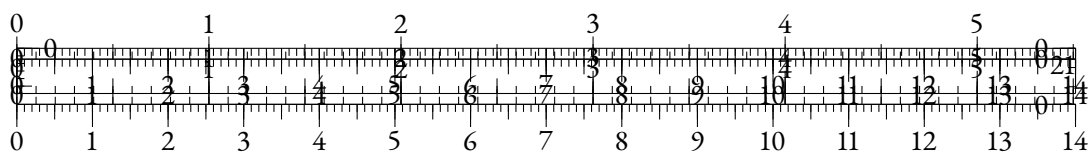




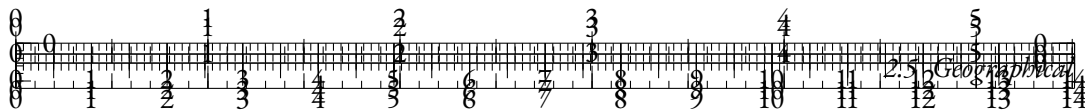
## PART II

### A GOOD PART

You can also use parts in order to partition your great work into larger ‘chunks’. This involves some manual adjustments in terms of the layout, though.







## 2.5 GEOGRAPHICAL

### 2.5.1 INTRODUCTION & RELATED WORK

In recent years, numerous studies analyzing the hidden representations of self-supervised language models have provided insights into how these models incorporate linguistic knowledge from their training data (Gupta et al., 2015; Köhn, 2015; Shi et al., 2016; Hupkes and Zuidema, 2018; Conneau et al., 2018a; Jawahar et al., 2019).

This line of work has been called probing, as most approaches are generally based on the training of classifiers—or *probes*—upon frozen hidden representations.

Analyzing the representations of language models can point out sociocultural biases that were inherently learned by the models during training (Zhao et al., 2018), and training probes can help with mitigating these biases (Ravfogel et al., 2020; Iskander et al., 2023).

Among probing tasks, several works have focused on geographical representations that are implicitly embedded in language models. 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). Imbalanced frequency distributions of geographical references in pretraining data causes distortions in the representational space (Zhou et al., 2021a). 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.

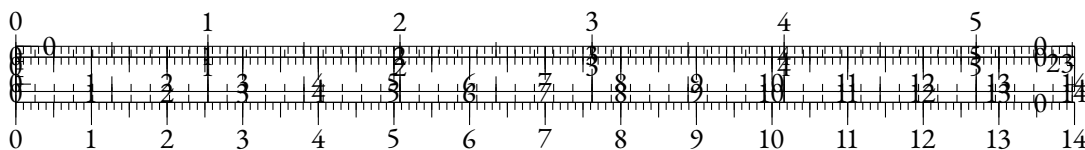
In this work, we propose to extend the analysis by Gurnee and Tegmark (2024) to smaller language models, in order to observe how scale affects the ability of models to implicitly embed geographical information from raw training data. We show that such ability consistently improves with model size, and 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.

### 2.5.2 SCALING LAWS OF GEOGRAPHICAL PROBING

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



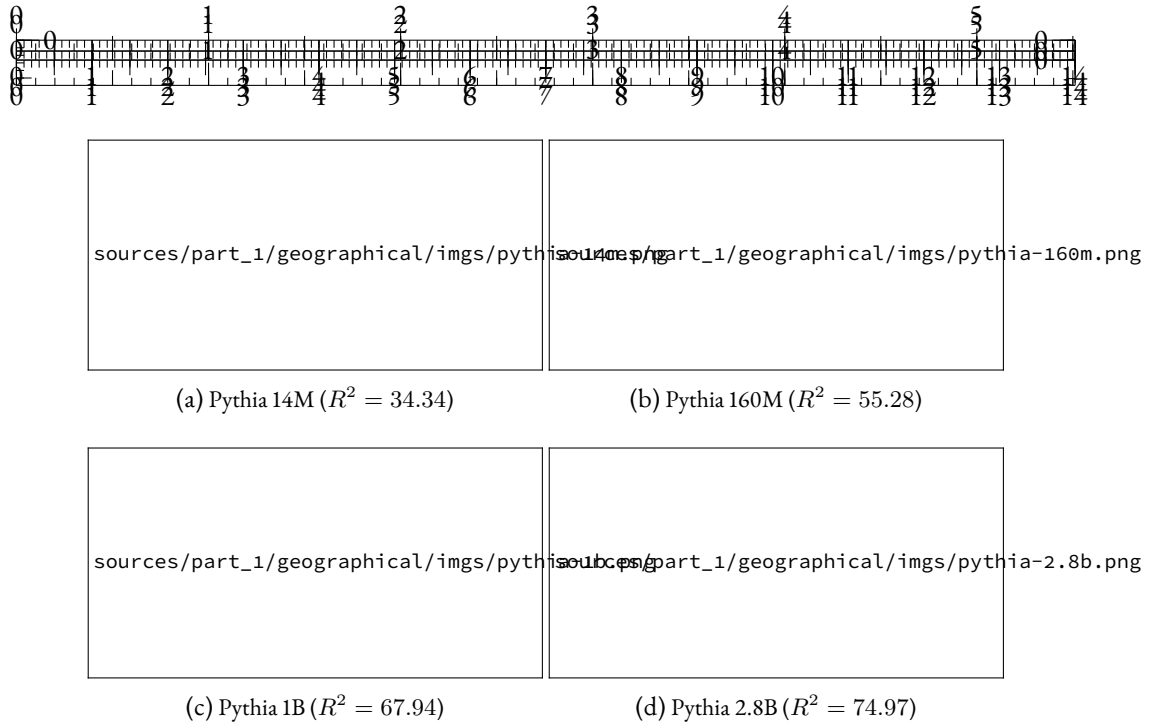


Figure 2.1: Predicted coordinates of test set instances for different model sizes. Each color represents a different continent.

## 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.

## RESULTS

In Figure 2.1, 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.

We show in Figure 2.2 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 (Shliazhko et al., 2024), and Llama-2 (Touvron et al., 2023). We also display results for several encoder models: BERT (Devlin et al., 2019; Turc et al.,



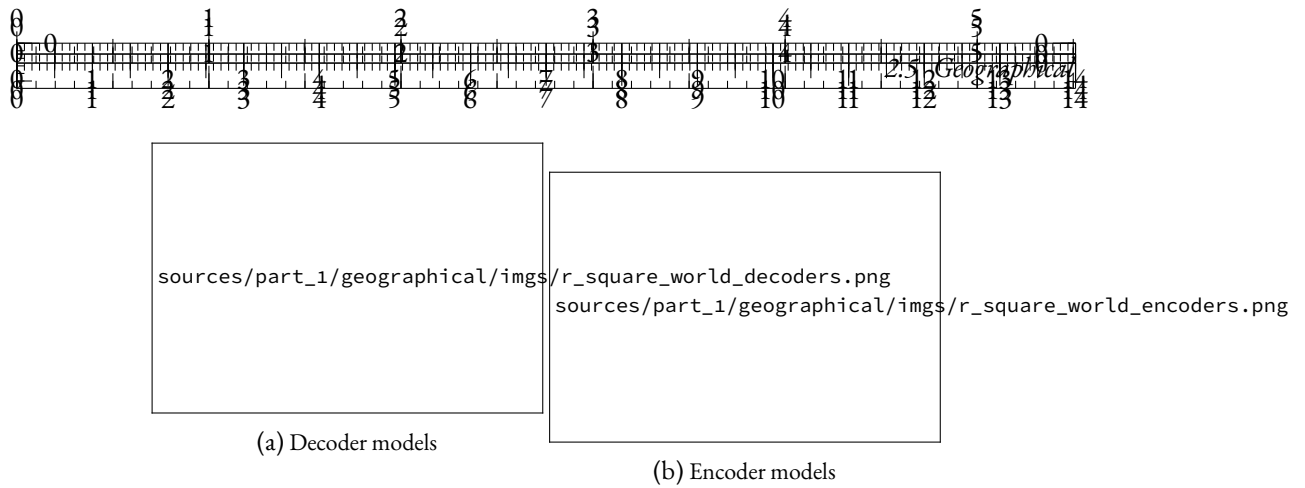


Figure 2.2: Evolution of the  $R^2$  coefficient on the test set for various model suites.

2020), RoBERTa (Liu et al., 2019), ELECTRA (Clark et al., 2020b), 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., 2020) 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.

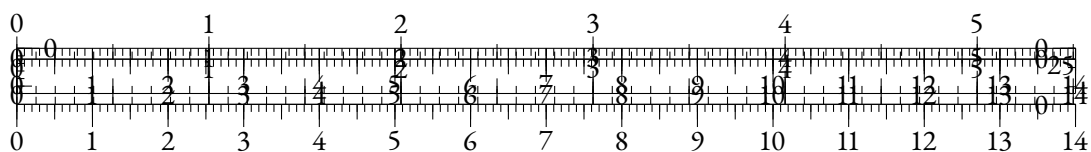
We verified that the better performance of larger models was not solely related with the ability of the probes to extract better patterns from their higher-dimensionality hidden representations. We achieved this by concatenating representations with themselves to increase dimensionality without introducing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

### 2.5.3 GEOGRAPHICAL BIAS AND SCALE

In Figure 2.1, 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.

#### MEASURING BIAS

We group probe performance as measured by mean-squared error (MSE) on predicted coordinates, and average measures by continent in Figure 2.3. 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.



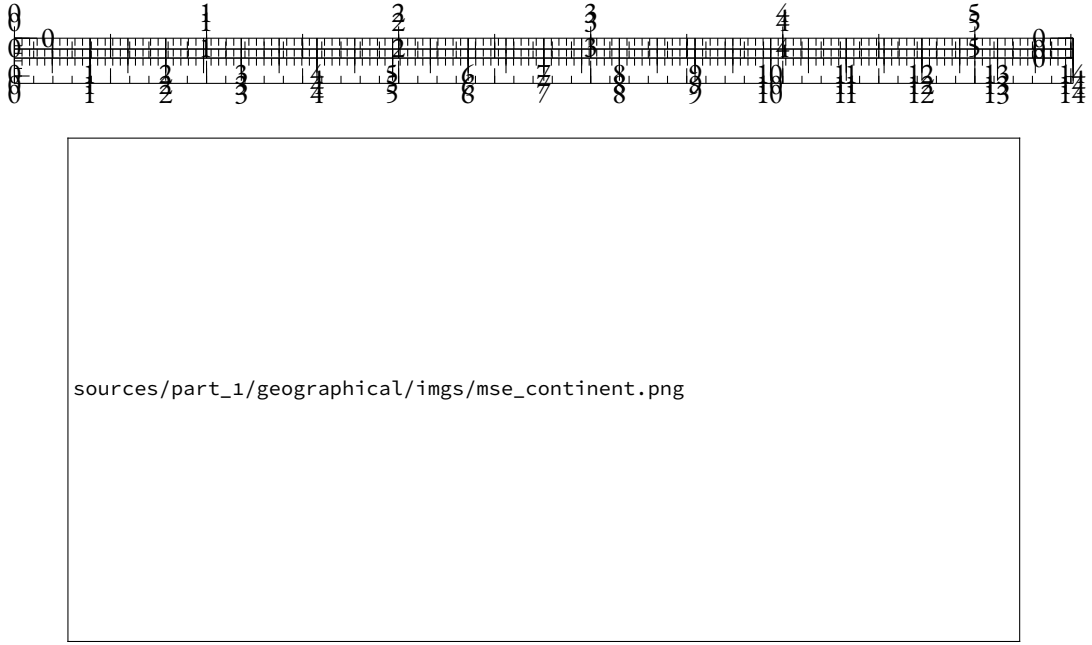


Figure 2.3: Average MSE by continent for different sizes in the Pythia suite.

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.

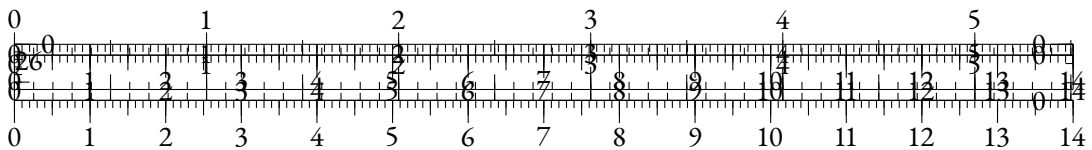
Figure 2.4 shows that the larger the model is, the more heterogeneous the probe performance is across countries and continents. This contradicts the impression given by Figure 2.1, and shows that scale does not solve the geographical discrepancy caused by bias inherent to the training data.

In Figure 2.5, 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.

#### 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 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



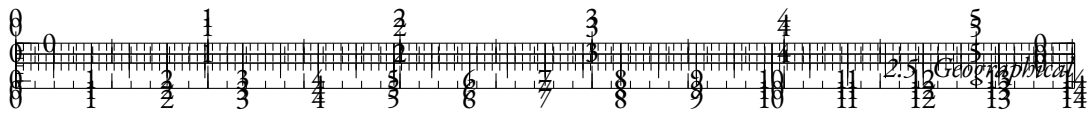
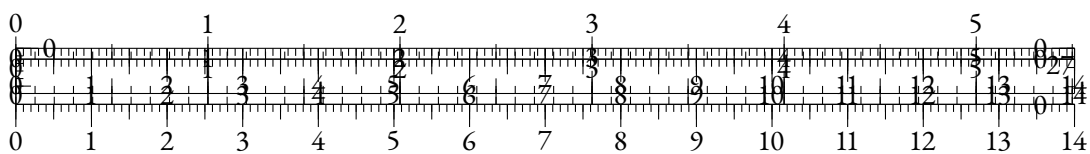


Figure 2.4: Gini coefficients of MSE on the test set averaged by country or by continent, as model size increases.



Figure 2.5: Test log-MSE for Pythia-1B as plotted on a World map.



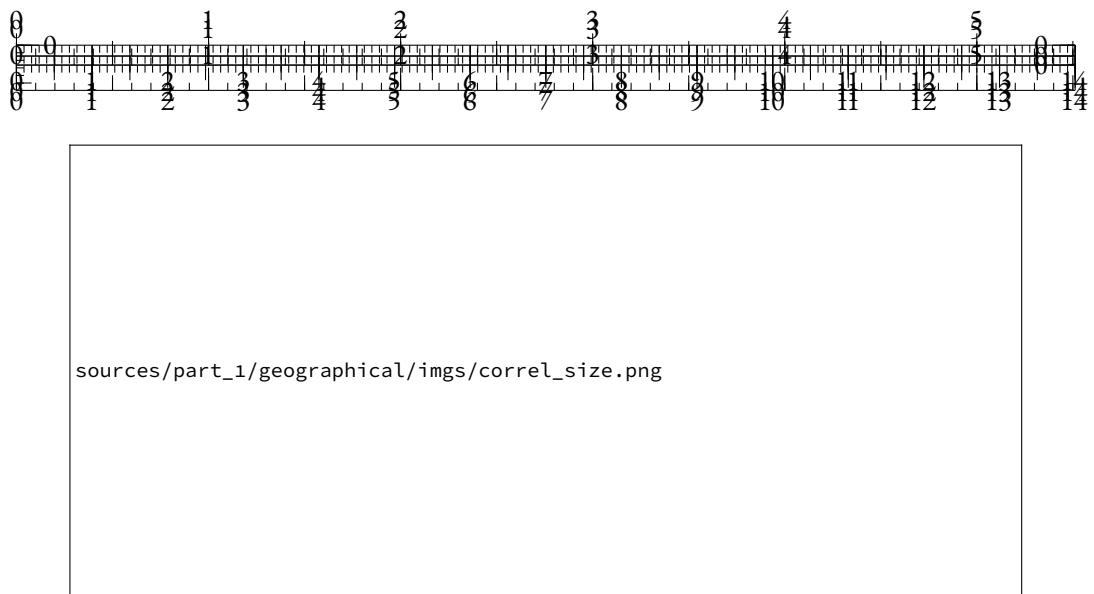


Figure 2.6: Pearson correlation coefficients of various factors with location-wise MSE, for several Pythia model sizes. \*: Tests that yielded p-values above 0.05.

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.

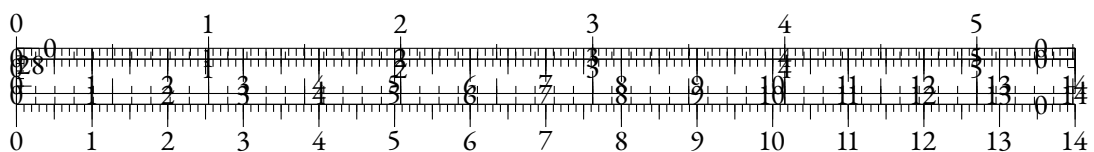
We do not count occurrences of location names directly, as matching locations on the basis of their names does not account for named entity ambiguity. An example of ambiguous location name is *Fully*, which is a town in Switzerland. An exact match strategy overestimates by large margins the occurrence 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.

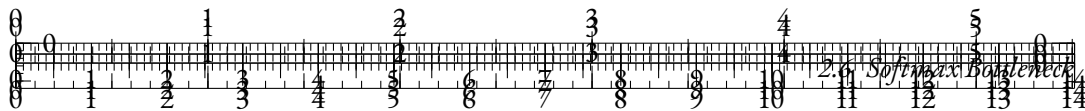
We display Pearson correlations between each of the aforementioned factors and the entity-level MSE for each model size in Figure 2.6. As in Figure 2.1a, we observe that the error on coordinates prediction is negatively correlated with the latitude, i.e. southern locations are less accurately identified. 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 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.

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





### 2.5.4 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 study, 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).

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.

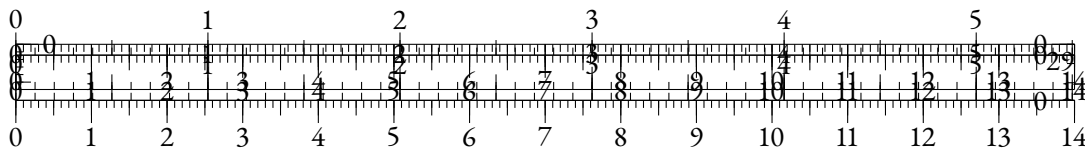
While it may initially seem that this performance increase mitigates heterogeneity between Southern and Northern countries, we actually show that larger models tend to be more biased according to the Gini coefficient taken on prediction error. This tends to show that scaling language models can amplify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

## 2.6 SOFTMAX BOTTLENECK

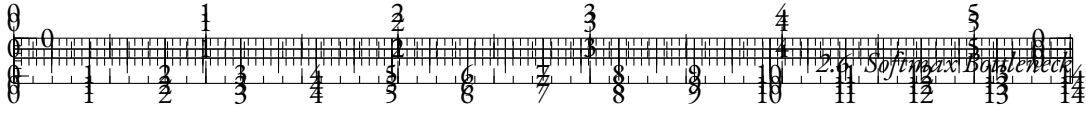
### 2.6.1 INTRODUCTION

The representation degeneration problem is a common phenomenon that affects self-supervised learning methods used for textual data (Gao et al., 2019b; Lai et al., 2023), among other modalities (Jing et al., 2022; Godey et al., 2024). This phenomenon consists in the emergence of degenerated structures in the intermediate latent spaces of language models throughout training. In particular, many observations on the intermediate representations of Language Models (LMs) have shed light on their low angular variability (or *anisotropy*) by showing that cosine-similarity between pairs of intermediate embeddings tend to be unexpectedly high (Zhou et al., 2021b; Rajaei and Pilehvar, 2022). Other works have identified outlier dimensions that emerged during training (Puccetti et al., 2022). However, these observations were mostly made on relatively small-scale models of dimensions comparable to BERT (Devlin et al., 2019) or models from the GPT-2 family (?).

These models are usually composed of a neural network  $f_\theta$  that takes sequences of tokens  $(y_{<i}) \in [1, V]^{i-1}$  as inputs and produces a relatively low-dimensional contextual representation in  $\mathbb{R}^d$ , where  $d$  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







**SOFTMAX BOTTLENECK** The concept of *softmax bottleneck* was introduced in Yang et al. (2018), where the authors show that a model using a hidden dimension inferior to the rank of the contextual probability matrix cannot predict correctly in every context. They then hypothesize that this rank is relatively high in natural language and propose an alternative method for the predictive layer of language models. Subsequent works have explored negative effects of the softmax linear layer on language modeling performance (Chang and McCallum, 2022) and possible alternatives (Lin, 2021; Kanai et al., 2018). We extend this line of work by quantifying the critical dimensionalities involved in the softmax bottleneck.

**REPRESENTATION DEGENERATION** is a phenomenon in which pretrained models tend to adopt low-entropy singular value distributions (Jing et al., 2022). In language modeling, representation degeneration takes the form of anisotropy (Ethayarajh, 2019; Rajaei and Pilehvar, 2021) and was proven to be related with the Zipfian shape of token distribution (Gao et al., 2019b; Biš et al., 2021). We study this phenomenon along training and its relation with saturation.

**DATA DIMENSIONALITY AND PERFORMANCE** Sharma and Kaplan (2022) link the scaling laws observed across pretrained models to data dimensionality, through the lens of Intrinsic Dimension (Camastra and Staiano, 2016). While they show that Singular Value Decomposition (SVD) is not suited for studying the dimensionality of the data manifold in the universal approximation paradigm, we argue that it is well-suited, to a certain extent, when studying the performance of a linear classifier limited by the dimensionality of input representations.

### 2.6.3 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).

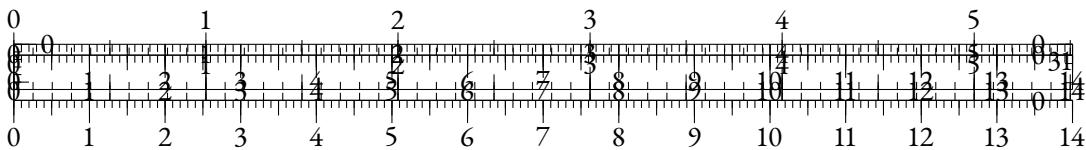
In Figure 2.7a, we clearly see that models up to 410M parameters suffer from the saturation phenomenon, characterized as an increase of the in-domain loss in advanced training stages.

In Figure 2.7b, 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%.

A similar performance saturation is also observed on datasets used for evaluation in the LM Evaluation Harness (Gao et al., 2023), as shown in Table 2.1.



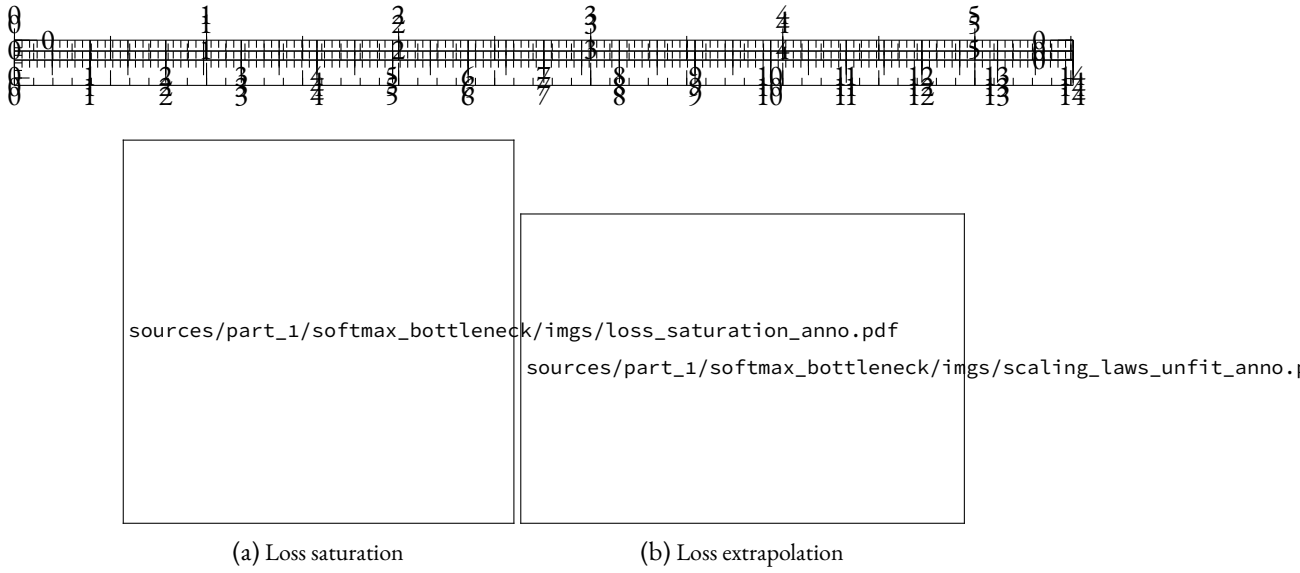


Figure 2.7: Performance of Pythia models on the Pile. On the left, we compare training dynamics of models from 14M (top) to 410M (bottom) parameters, displaying darker shades as we approach the minimal value. On the right, we fit a power law on larger models and find that final checkpoints of smaller models underperform compared to predictions.

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

#### 2.6.4 PERFORMANCE SATURATION IS RANK SATURATION

##### ANISOTROPY AT SCALE

Anisotropy is a common form of representation degeneration that has been observed among various small language models. It consists in reduced angular variability of the representation distribution at a given layer. Previous works (Ethayarajh, 2019; Godey et al., 2024) notice that almost all layers of small Transformers language models are anisotropic. A common measure for anisotropy in a set  $H$  of vector representations is the average cosine-similarity:

$$\mathcal{A}(H) = \frac{1}{|H|^2 - |H|} \sum_{h_i, h_j \in H, i \neq j} \frac{h_i^T h_j}{\|h_i\|_2 \cdot \|h_j\|_2}$$

However, 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 (?), 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.





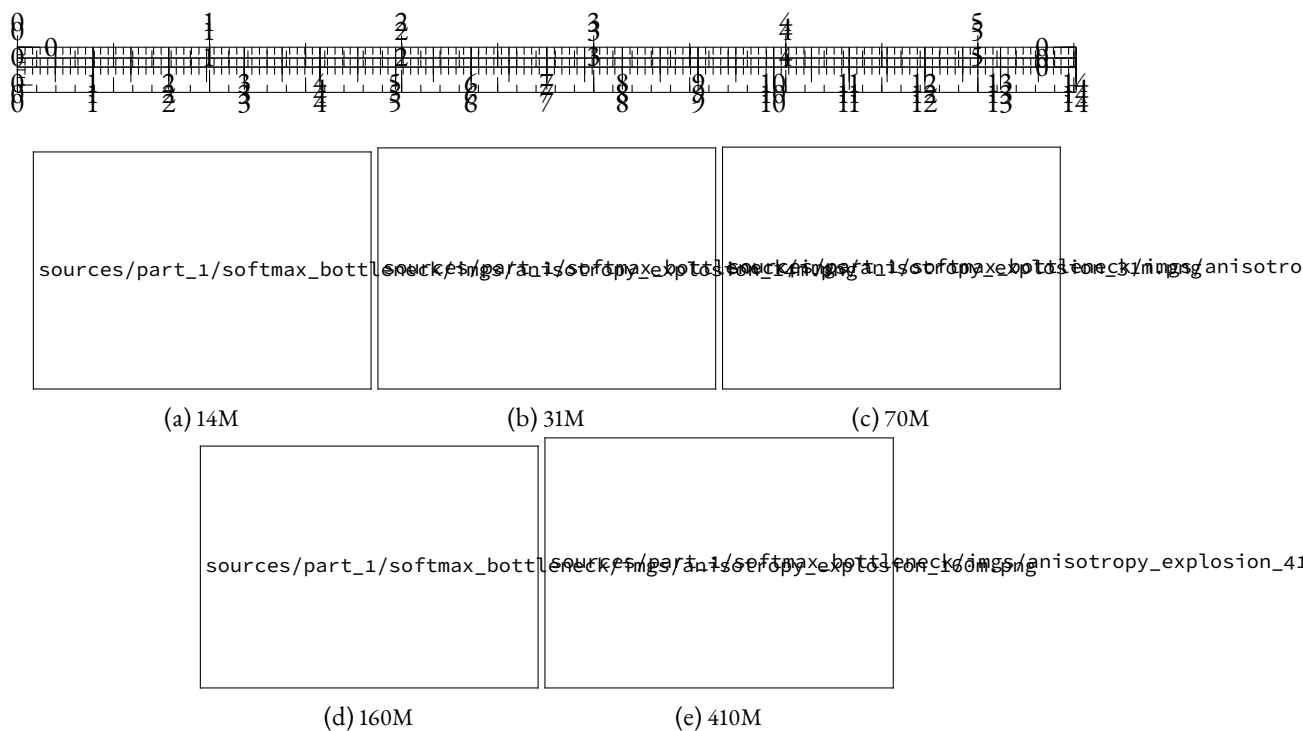


Figure 2.9: Evolution of the language modeling performance on the Wikipedia test set from the LM Evaluation Harness (Gao et al., 2023) and last-layer anisotropy of Pythia models along training (color).

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 2.11 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.

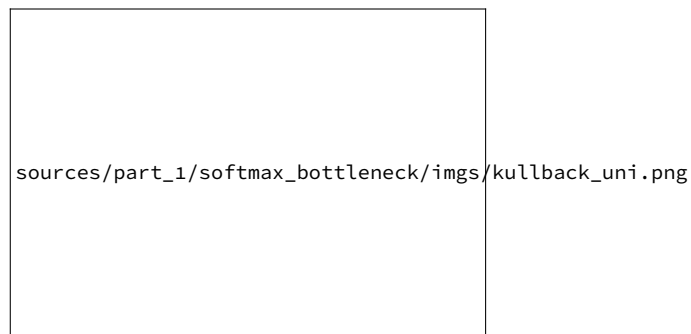
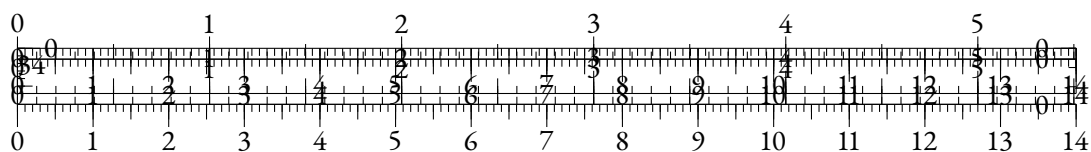


Figure 2.11: Training dynamics of the singular entropy, for different Pythia models.

## 2.6.5 THE SOFTMAX BOTTLENECK & LANGUAGE DIMENSIONALITY

### INHERENT DIMENSIONALITY OF NATURAL LANGUAGE

Intuitively, the saturation of the singular values distribution observed only for smaller models in Section 2.6.4 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



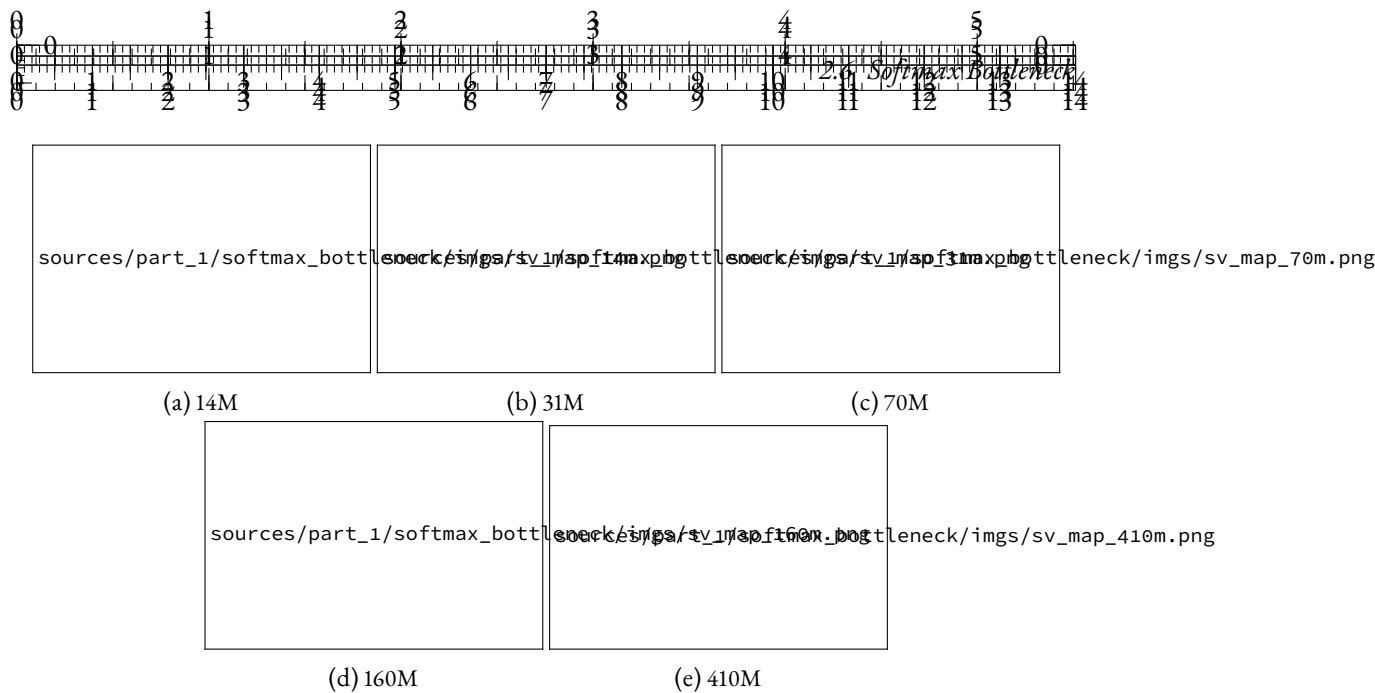


Figure 2.10: Evolution of the singular value distributions of the LM heads of Pythia models during training, normalized by the maximum singular value.

to estimate the dimensionality of the contextual probability distribution the head’s outputs are supposed to match.

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  $W = AB \in \mathbb{R}^{V \times d}$ , where the coefficients of  $A \in \mathbb{R}^{V \times r}$  and  $B \in \mathbb{R}^{r \times d}$  are drawn from  $\mathcal{N}(0, 1)$  ( $d$  being the hidden dimension of the model). The rank of such  $W$  matrices is limited by the parameter  $r \in [1, d]$ , 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 details in Section 2.6.2).

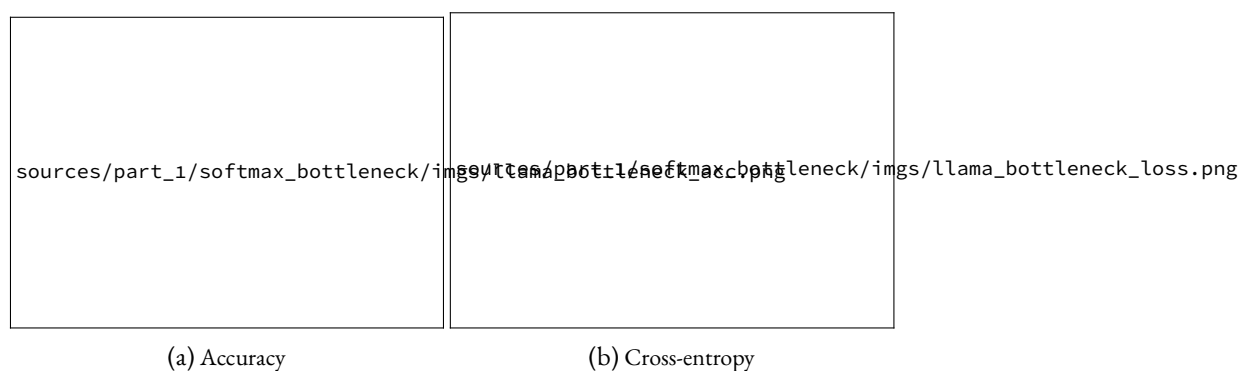
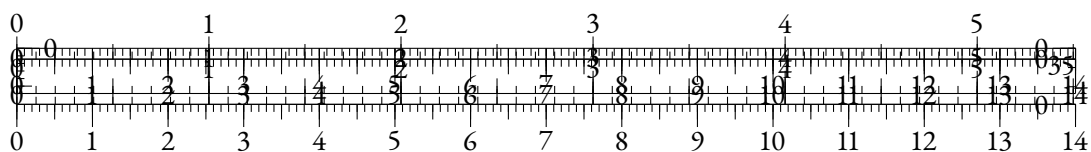
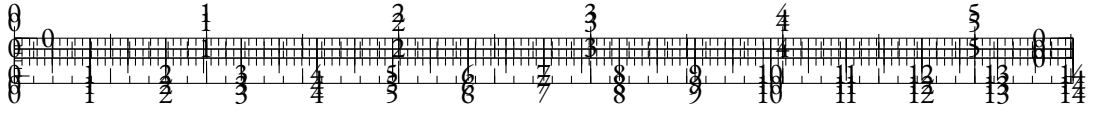


Figure 2.12: Performance of several models as the bottleneck dimension of the head increases.





In Figure 2.12, we observe that perplexity starts to noticeably decrease when the rank of the language modeling head  $W$  is 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 their singular value distributions, as in Terashima et al. (2003). In Figure 2.13, 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 2.6.2), normalized by the Frobenius norm of  $W$ :

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

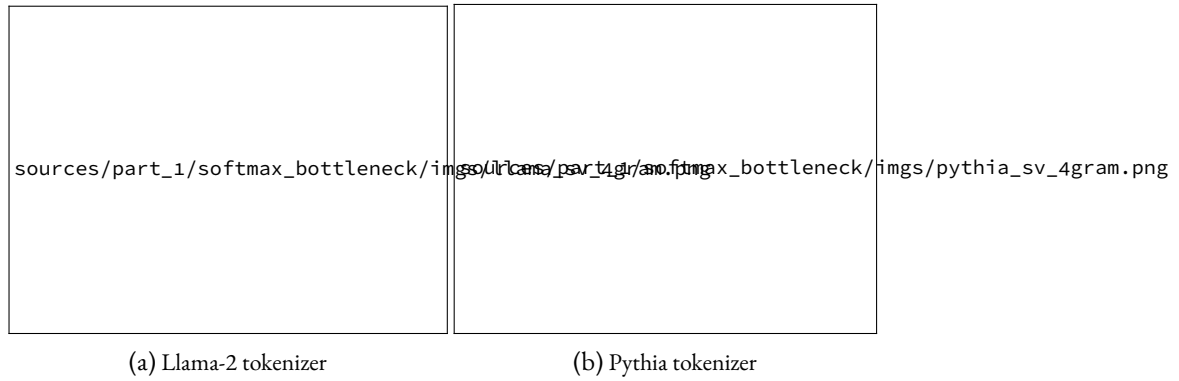
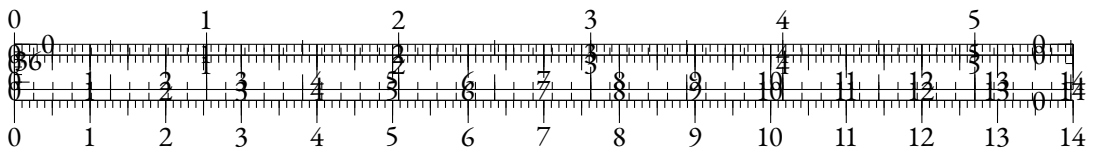


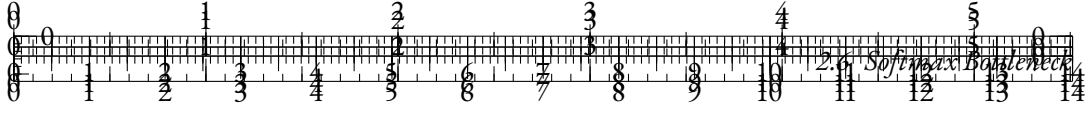
Figure 2.13:  $W$ -error as  $d$  increases, for different tokenizers and datasets. We observe that while  $W$ -error can be halved using 1000 or 2000 dimensions, it only becomes negligible after 10,000-15,000 dimensions.

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.

## 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 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  $(y_i)_{i \in [1, |y|]}$  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  $y_{<i}$  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  $x_i^* = \phi^*(y_{<i})$ .

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

$$W^* = \arg \min_{W \in \mathbb{R}^{V \times \infty}} \sum_{y \in \mathcal{T}} \sum_i \mathcal{L}(W, x_i^*, y_i) \quad (2.1)$$

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

$$\mathcal{L}(W, x, y) = -\log(\sigma(Wx)_y)$$

In practice, a neural language model  $\phi_\theta$  produces contextual representations  $x_i = \phi_\theta(y_{<i})$  of dimension  $d \in \mathbb{N}^*$ . The linear language modeling head  $W_\theta \in \mathbb{R}^{V \times d}$  is trained concurrently with  $\phi_\theta$  with the same objective as in Equation 2.1.

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

$$W_d^* = \arg \min_{W \in \mathbb{R}^{V \times \infty}} \sum_{y \in \mathcal{T}} \sum_i \mathcal{L}(W, x_i^*, y_i) \text{ s.t. } \text{rank}(W) \leq d \quad (2.2)$$

Lemma 2.6.1 shows that by approaching  $W^*$  directly, we can asymptotically expect to close the performance gap.

**Lemma 2.6.1.** (proof in Section 2.6.1) 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  $\varepsilon \rightarrow 0$ :

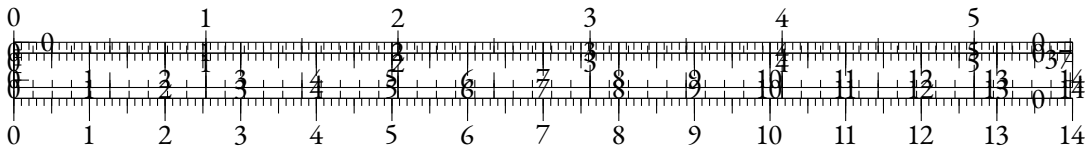
$$|\mathcal{L}(W, x_i^*, y_i) - \mathcal{L}(W^*, x_i^*, y_i)| = O(\varepsilon)$$

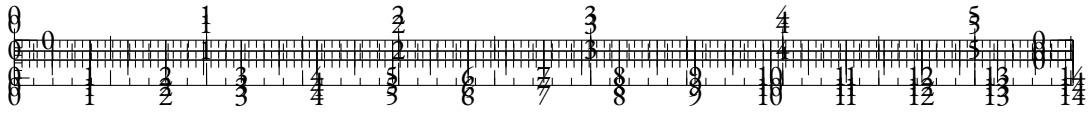
Hence, our problem is linked to a low-rank matrix approximation (Kumar and Schneider, 2017), which has direct connections with spectral theory. In our case, we can use the Eckart–Young–Mirsky theorem.

**Lemma 2.6.2.** (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 2.6.3.



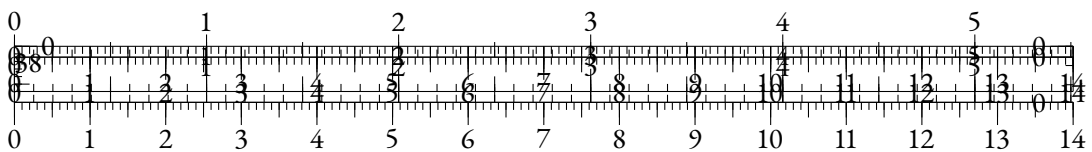


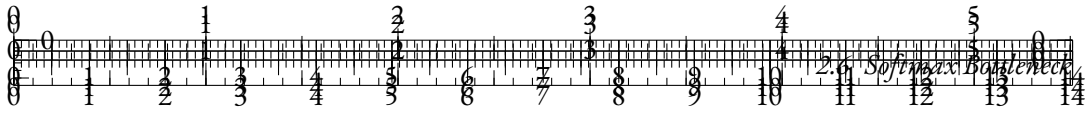
**Theorem 2.6.3.** (proof in Section 2.6.1) Let’s consider  $(\sigma_i)$  the singular values of  $W^*$  in decreasing order. Then, when  $d \rightarrow V$ , the loss gap induced by a  $d$ -dimensional bottleneck on the linear LM head follows:

$$\sum_{y \in \mathcal{T}} \sum_i \mathcal{L}(W_d^*, x_i^*, y_i) - \mathcal{L}(W^*, x_i^*, y_i) = O\left(\sqrt{\sum_{i=d+1}^V \sigma_i^2}\right)$$

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 2.6.3 is not particularly strong, as discussed in Section 2.6.1.

In Figure 2.14, we compare the results of the head bottleneck experiment of the Pythia-1B model in Section 2.6.5 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.





## 2.6.6 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 2.6.4 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; Biś et al., 2021) and show the importance of token frequency in the LM head mechanism (Meister et al., 2023).

Beyond the scope of this article, 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.

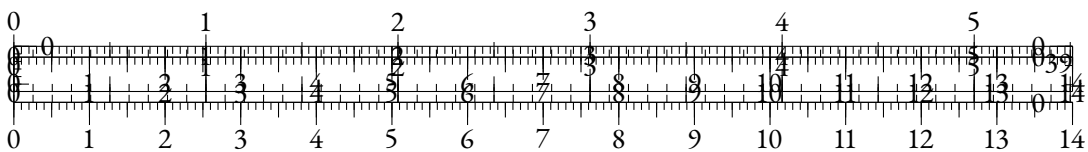
## CONCLUSION

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

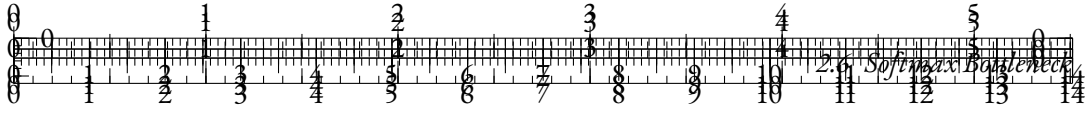


Figure 2.14: Final loss with trained rank-constrained heads (mimicking  $W_d^*$ ), as a function of the theoretical  $W$ -error for rank  $d$  on the head of the Pythia-1B model.









## 2.6.1 PROOFS

### LEMMA 2.6.1

The proof is mainly based on calculations and limited development:

$$\begin{aligned}
& |\mathcal{L}(W, x_i^*, y_i) - \mathcal{L}(W^*, x_i^*, y_i)| \\
&= \left| -\log \frac{\exp((W x_i^*)_{y_i})}{\sum_{j \in V} \exp((W x_i^*)_j)} + \log \frac{\exp((W^* x_i^*)_{y_i})}{\sum_{j \in V} \exp((W^* x_i^*)_j)} \right| \\
&= \left| -(\varepsilon M x_i^*)_{y_i} + \log \frac{\sum_{j \in V} \exp((W^* x_i^*)_j) \exp((\varepsilon M x_i^*)_j)}{\sum_{j \in V} \exp((W^* x_i^*)_j)} \right| \\
&= \left| -\varepsilon (M x_i^*)_{y_i} + \log \left( 1 + \frac{\sum_{j \in V} \varepsilon \exp((M x_i^*)_j)}{\sum_{j \in V} \exp((W^* x_i^*)_j)} + o(\varepsilon) \right) \right| \\
&= \left| -\varepsilon (M x_i^*)_{y_i} + \varepsilon \frac{\sum_{j \in V} \exp((M x_i^*)_j)}{\sum_{j \in V} \exp((W^* x_i^*)_j)} \right| + o(\varepsilon) \\
&= \varepsilon \left| -(M x_i^*)_{y_i} + \frac{\sum_{j \in V} \exp((M x_i^*)_j)}{\sum_{j \in V} \exp((W^* x_i^*)_j)} \right| + o(\varepsilon)
\end{aligned}$$

The continuous function  $M \rightarrow \left| -(M x_i^*)_{y_i} + \frac{\sum_{j \in V} \exp((M x_i^*)_j)}{\sum_{j \in V} \exp((W^* x_i^*)_j)} \right|$  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.

### THEOREM 2.6.3

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

$$\left| \sum_{y \in \mathcal{T}} \sum_i \mathcal{L}(W_d^*, x_i^*, y_i) - \mathcal{L}(W^*, x_i^*, y_i) \right| \leq \sum_{y \in \mathcal{T}} \sum_i |\mathcal{L}(W_d, x_i^*, y_i) - \mathcal{L}(W^*, x_i^*, y_i)| \quad (2.3)$$

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

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

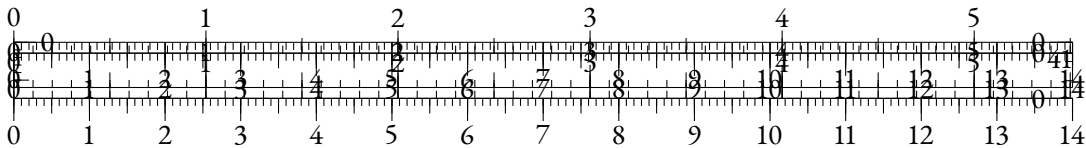






Figure 2.15: Chosen peak learning rates used for the rank-constrained head experiments for each model.

representation degeneration problem. Specifically, this degeneration is characterized by anisotropy, a property of hidden representations that makes them all close to each other in terms of angular distance (cosine-similarity).

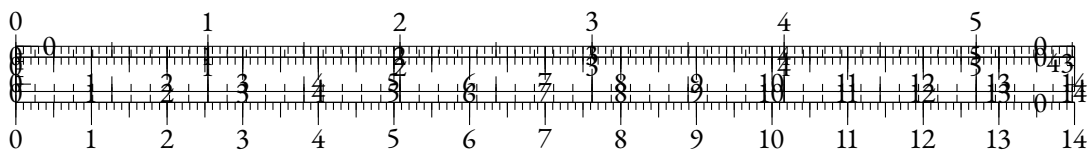
Anisotropy has been widely observed among self-supervised models based on Transformers, and literature currently suggests that it may be a consequence of optimizing the cross-entropy loss on long-tailed distributions of tokens (Gao et al., 2019b; Biš et al., 2021). However, it remains uncertain whether anisotropy is a fundamental property of Transformers-based models or a consequence of the pre-training process.

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.

## 2.7.2 RELATED WORK

The general phenomenon of anisotropy in token-based Transformers for language models has been shown in Ethayarajh (2019). Figure 2.16 extends one of their experiment to more architectures. Gao et al. (2019b) shows that the degeneration of representations comes from the distributions of



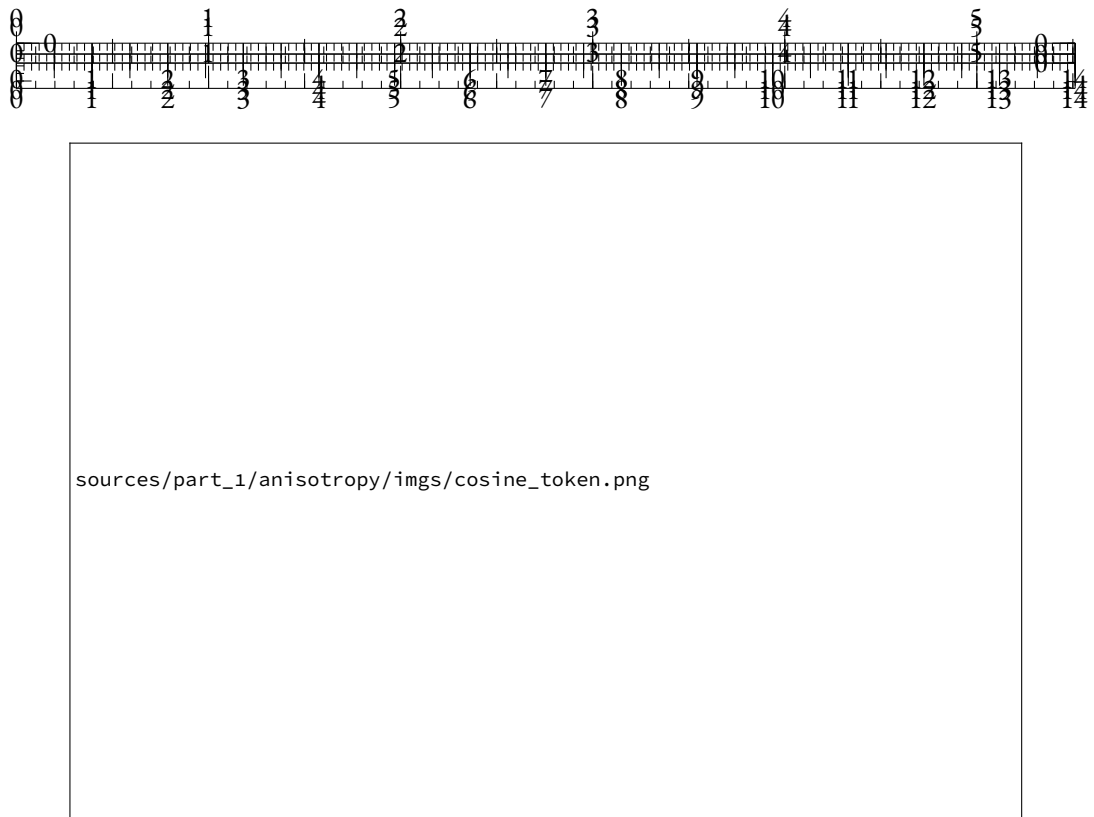
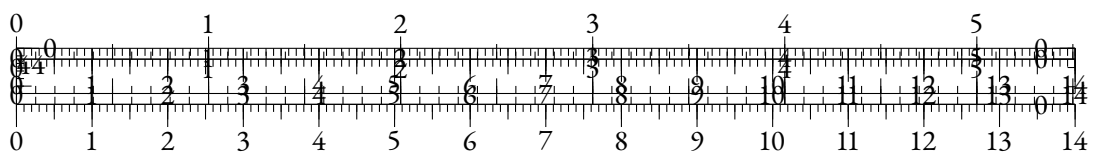


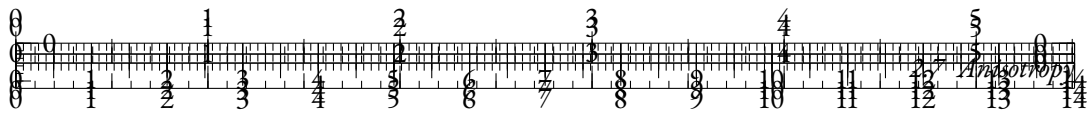
Figure 2.16: Average cosine-similarity between hidden representations across layers for token-level NLP models. For T5-base, we concatenate encoder and decoder results.

subwords in natural language, namely the existence of unused and rare tokens that tend to push all representations away from the origin towards a specific direction.

Other works have established a connection between word frequency and distortions of the latent spaces (Yu et al., 2022; Puccetti et al., 2022; Rajaei 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.

Several methods have been proposed to reduce anisotropy in Transformers-based LMs at token-level (Rajaei and Pilehvar, 2021; Wang et al., 2020), or at sentence-level (Gao et al., 2021; Yan et al., 2021; Su et al., 2021). They usually consist in post-processing the representations, 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.





## 2.7.3 ANISOTROPY IN PRE-TRAINED TRANSFORMERS

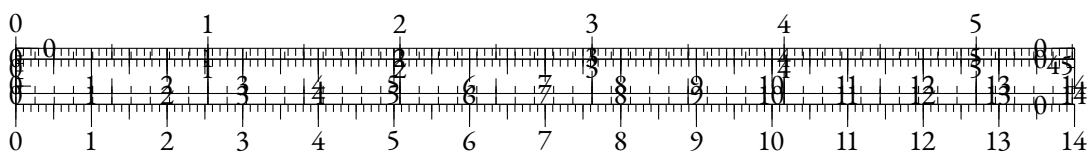
### CHARACTER-BASED NLP

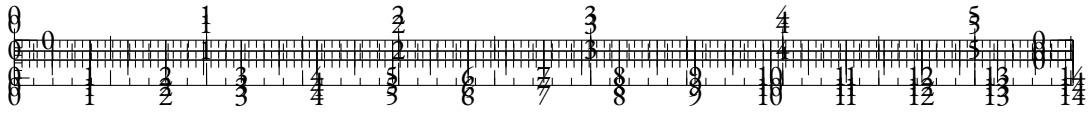


Figure 2.17: Average cosine-similarity between hidden representations across layers for character-level models.

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:

- 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., 2022a) 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 (Godey et al., 2022) 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 it would imply an embedding drift at 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 2.17, 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.

#### OTHER MODALITIES

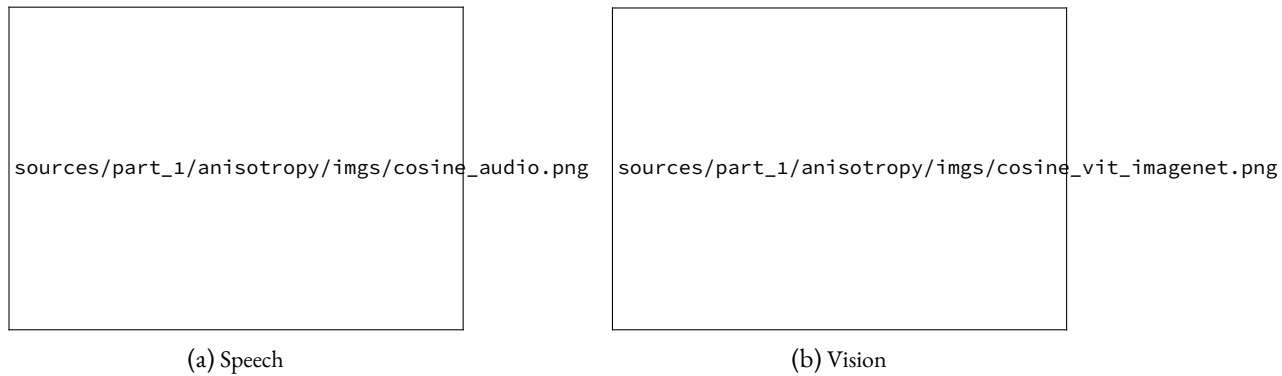
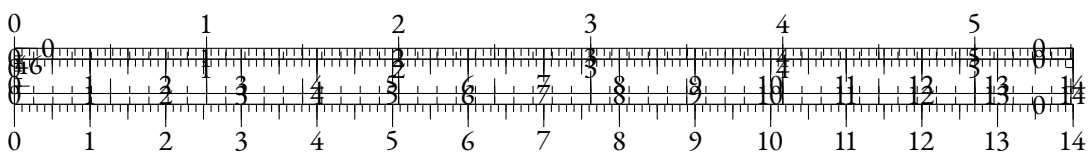


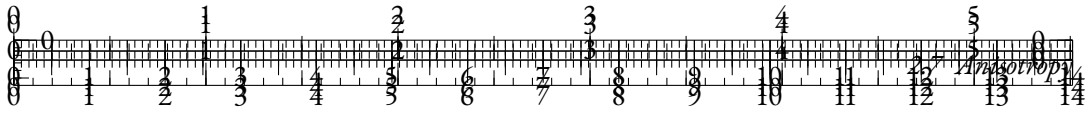
Figure 2.18: Average cosine-similarity between hidden representations across layers for Speech and Vision modalities. We observe that across both modalities, several models display significant levels of anisotropy.

We’ve 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. However, it may be argued that anisotropy is related to linguistic properties. 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).

As in subsection 2.7.3, 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 2.18.





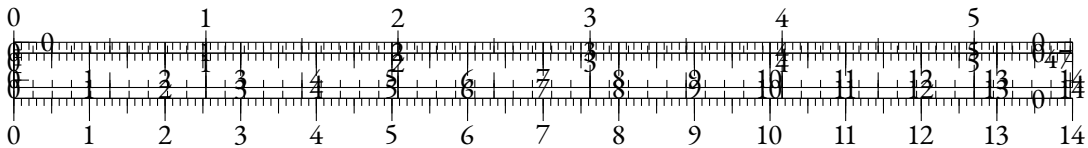
Once again, almost every model shows a significant level of anisotropy on some of its layers. Notably, speech models seem to have very anisotropic representations, as every layer of every model outputs an average cosine-similarity of at least 0.2. We find some exceptions among vision models, since the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

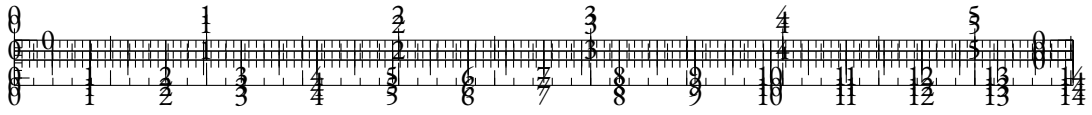
We also conduct the same experiment for convolution-based networks in the vision modality. The models at glance are ResNet (He et al., 2016), EfficientNet (Tan and Le, 2019), CvT (Wu et al., 2021), ConvNeXt (Liu et al., 2022), and VAN (Guo et al., 2023). For these networks, we flatten convolution maps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



Figure 2.19: Average cosine-similarity between hidden representations across layers for convolution-based vision models.

We observe in Figure 2.19 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.





## TO DRIFT OR NOT TO DRIFT?

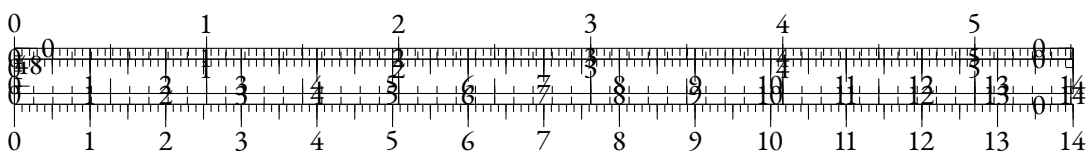
Related works (Biś et al., 2021; Gao et al., 2019b) show that anisotropy in subword-level language models is caused by a drift of the hidden representations in a shared direction. In this section, we try to extend this observation to other modalities.

We study the correlation between the uniformly measured cosine-similarity, and the norm of the average hidden representation  $||\bar{x}||_2$  for each layer. If anisotropy could be directly explained by the drift effect, we would expect a monotonic relation between  $||\bar{x}||_2$  and the average cosine-similarity. To verify this, we apply a Spearman correlation test on these two metrics for every model from subsection 2.7.3 and subsection 2.7.3, along with some token-level language models, namely T5 (Raffel et al., 2020a), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and GPT-2 (Radford et al., 2019).

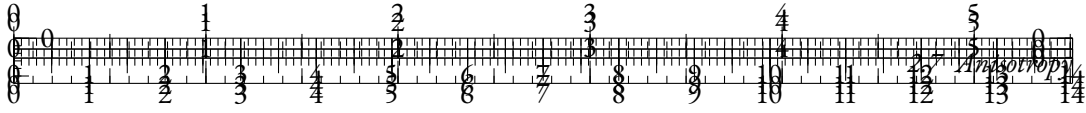


Figure 2.20: p-value of the Spearman correlation test between the norm of the average representation and the cosine-similarity averaged over all layers, across modalities. For models above the red dotted line, there is no significant ( $p > 0.05$ ) correlation between the drift effect and the anisotropy level.

In Figure 2.20, 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, except for GPT-2 for which the Spearman correlation metric may not be appropriate. We provide a similar analysis based on the Pearson correlation test and discuss the relevance of each statistic in Section 2.7.7.







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 2.20 also shows a correlation for all character-based models but Canine-C and MANTa-base.

#### 2.7.4 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  $\mathbf{x}$ . This allows us to mimic the common drift as mentioned in Biš et al. (2021) and to identify some properties induced by this artificial drift on the output representations.

##### 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  $\mathbf{x}$  of length 512 uniformly from  $E$ . To account for a drift component of norm  $N \in \mathbb{R}$ , we generate a vector  $b_u \sim \mathcal{N}(0, I_d)$ , which we normalize into  $b = \frac{b_u}{\|b_u\|_2} \times N$ . We finally compute  $T(\mathbf{x}_i + b)$  for every sequence  $x_i$ , and study the resulting distributions.

Specifically, we study the average norm of the input representations  $\mathbb{E}(\|\mathbf{x}_i + b\|_2)$  against the average norm of the output representations  $\mathbb{E}(\|T(\mathbf{x}_i + b)\|_2)$  in Figure 2.40b. We also retrieve the self-attention scores before the softmax operation, namely  $\frac{QK^T}{\sqrt{d_k}}$ , along with the corresponding  $Q$  and  $K$  matrices. We study some of their properties in Figure 2.22 and Figure 2.23.

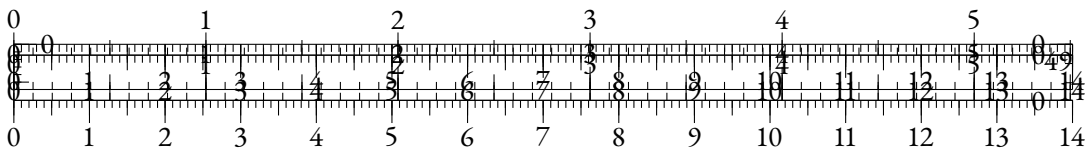
##### INPUT VS. OUTPUT ANALYSIS

In Figure 2.21a, 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.

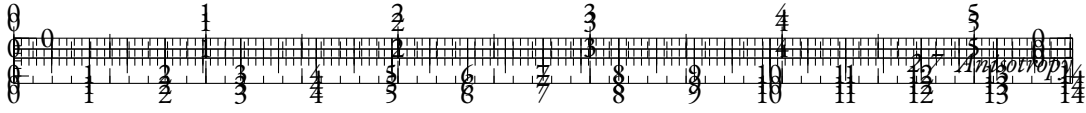
Interestingly, this shows that there is a *fixed point* in terms of norm in the Transformers function with biased input. More formally, there seems to exist a bias norm  $N^* \in \mathbb{R}_+$  such that:

$$\mathbb{E}_{x, b_{N^*}}(\|x_i + b_{N^*}\|) = \mathbb{E}_{x, b_{N^*}}(\|T(x_i + b_{N^*})\|)$$

Moreover, this fixed point level  $N^*$  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.







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

In Figure 2.26 and Figure 2.27, we compute a SVD of the union of  $Q_s$  and  $K_s$  for all steps  $s$ , so that the projection makes sense for both distributions across steps for visualization purposes<sup>2</sup>. As shown in our selected examples, we observe that the dynamics of  $\bar{Q}_s$  and  $\bar{K}_s$  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 2.27a shows that the common direction dynamics appear in the first few steps, while the opposite direction dynamics of Figure 2.27b only starts after 8% of the total training steps.

To consolidate our observations, we compute the evolution of the cosine-similarity between  $\bar{Q}_s$  and  $\bar{K}_s$  along training in Figure 2.28. We also display some projected  $Q_s$  and  $K_s$  distributions for several  $s$  steps in Figure 2.26.

Figure 2.28 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.

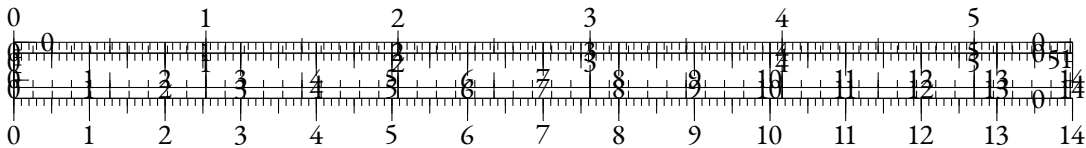
As shown in Figure 2.29, this drift induces an increase in the magnitude of scalar products obtained in the self-attention  $QK^T$  operation, thus facilitating the emergence of sharp patterns where attention focuses on specific tokens.

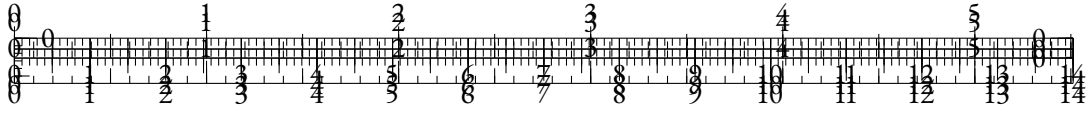
Finally, Figure 2.30 describes the evolution of the average entropy in self-attention distributions. We observe that training induces an overall decay of the entropy for all layers, with different dynamics. This corresponds to sharper self-attention distributions. It is interesting to notice that the distributions in the first layers remain sharper than the ones in the last layers.

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.

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.

<sup>2</sup>We actually uniformly sample 20% of the whole set of representations to compute the SVD under reasonable memory constraints.





## 2.7.6 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 subsection 2.7.4 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 subsection 2.7.5 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.

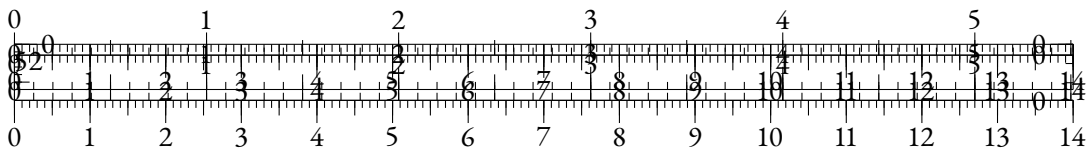
**CONSISTENCY OF THE SVD** In subsection 2.7.5, we use an SVD on the *union* of  $Q_s$  and  $K_s$  for visualization purposes (see Figure 2.26 and Figure 2.27). It may be argued that this approach favors the emergence of a discriminative singular direction, that helps distinguish between keys and queries, thus supporting the findings in a less convincing way. To address this concern, we display alternative projections in subsection 2.7.9, where we compute the SVD on  $Q_s$  or  $K_s$  only, and then project all representations using this SVD. Our observations show that our findings are consistent for these alternative projections.

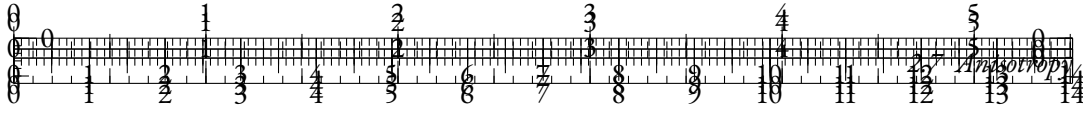
**HARMFULNESS OF ANISOTROPY** Even though anisotropy has not been shown to be an issue in language modeling, previous works have advocated that removing anisotropy in output representations leads to better sense disambiguation abilities (Bihani and Rayz, 2021; Biś et al., 2021). Isotropic models could also improve cross-lingual alignment in multilingual language models (Hämmerl et al., 2023). Nevertheless, concurrent works have suggested that anisotropy may not hurt the quality of the representations (Ait-Saada and Nadif, 2023; Rudman and Eickhoff, 2024). 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  $QK^T$  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 the Discussion section, we acknowledge that subsection 2.7.4 does not take into account the training dynamics, and only exposes some properties of the Transformer layer at initialization. We also notice that the Spearman correlation test used in Figure 2.20 may not be well-suited for such noisy observations, as the high p-value of the GPT-2 model shows. We provide a similar graph based on the Pearson correlation in Section 2.7.7.

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.

## ETHICS STATEMENT

To the best of our knowledge, our work does not raise any ethical concern. However, as noted in Zhou et al. (2021a), we believe that distortions in the embedding space may be related to bias in the training data, whether it is inherent to the structure of the modality (e.g. the Zipfian distribution of words), or due to human factors (e.g. geographical considerations).

## ACKNOWLEDGEMENTS

This work was funded by the last authors' chair in the PRAIRIE institute funded by the French national agency ANR as part of the "Investissements d'avenir" programme under the reference ANR-19-P3IA-0001. This work was granted access to the HPC resources of IDRIS under the allocation 2023-AD011013680R1 made by GENCI.

We would like to thank Roman Castagné for useful discussions that led to focusing on observing the effect of anisotropy in the self-attention process.

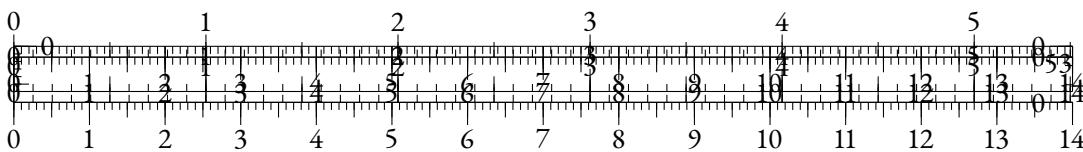
### 2.7.7 PEARSON CORRELATION OF THE DRIFT NORM AND ANISOTROPY

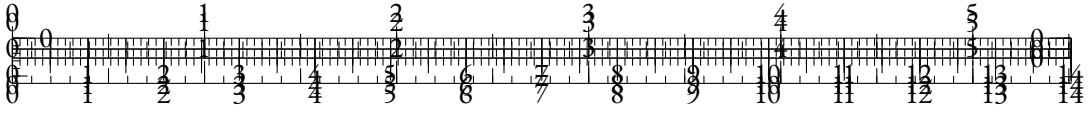
The Pearson test measures a linear correlation between random variables, while the Spearman test measures a monotonic correlation. As there is no specific argument in favor of a linear relationship between the measured distributions (average cosine-similarity and norm of the average representation), we decided to use 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).

### 2.7.8 COSINE-SIMILARITY AND ANISOTROPY

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 2.32, 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 attained 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 2.33, 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.

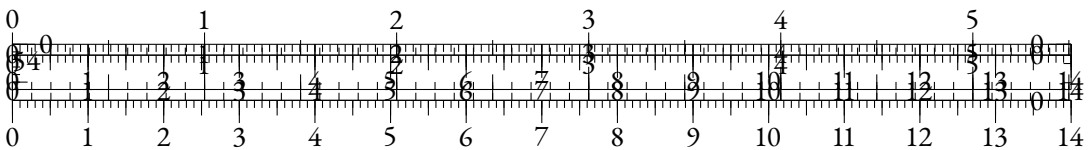
Nevertheless, as we go towards higher dimensional spaces for bigger models (e.g. Llama-65B from Touvron et al. (2023) has 8192 hidden dimensions), we believe that it may be relevant to introduce isotropy metrics that are grounded to isotropic cosine-similarity distributions. We leave this question for future works.

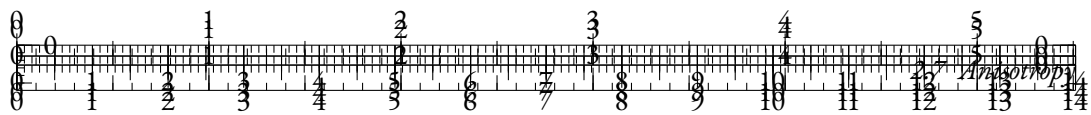
### 2.7.9 OTHER PROJECTIONS FOR $Q_s$ AND $K_s$

As mentioned in the Discussion (subsection 2.9.6), we reproduce visualizations from subsection 2.7.5 using different projection choices. Namely, we compute the SVD on  $K_s$  only in Figure 2.34 and Figure 2.36, and on  $Q_s$  only in Figure 2.35 and Figure 2.37.

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 2.34 and Figure 2.35 support our conclusion that the drift directions of keys and queries are aligned during training.

### 2.7.10 STABILITY ACROSS MULTIBERT SEEDS



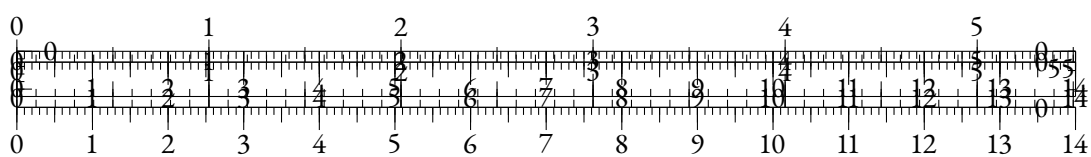


(a) Cosine similarity



(b) Norm

Figure 2.21: Input/Output comparison of a Transformer block from BERT-base as the bias norms increases.



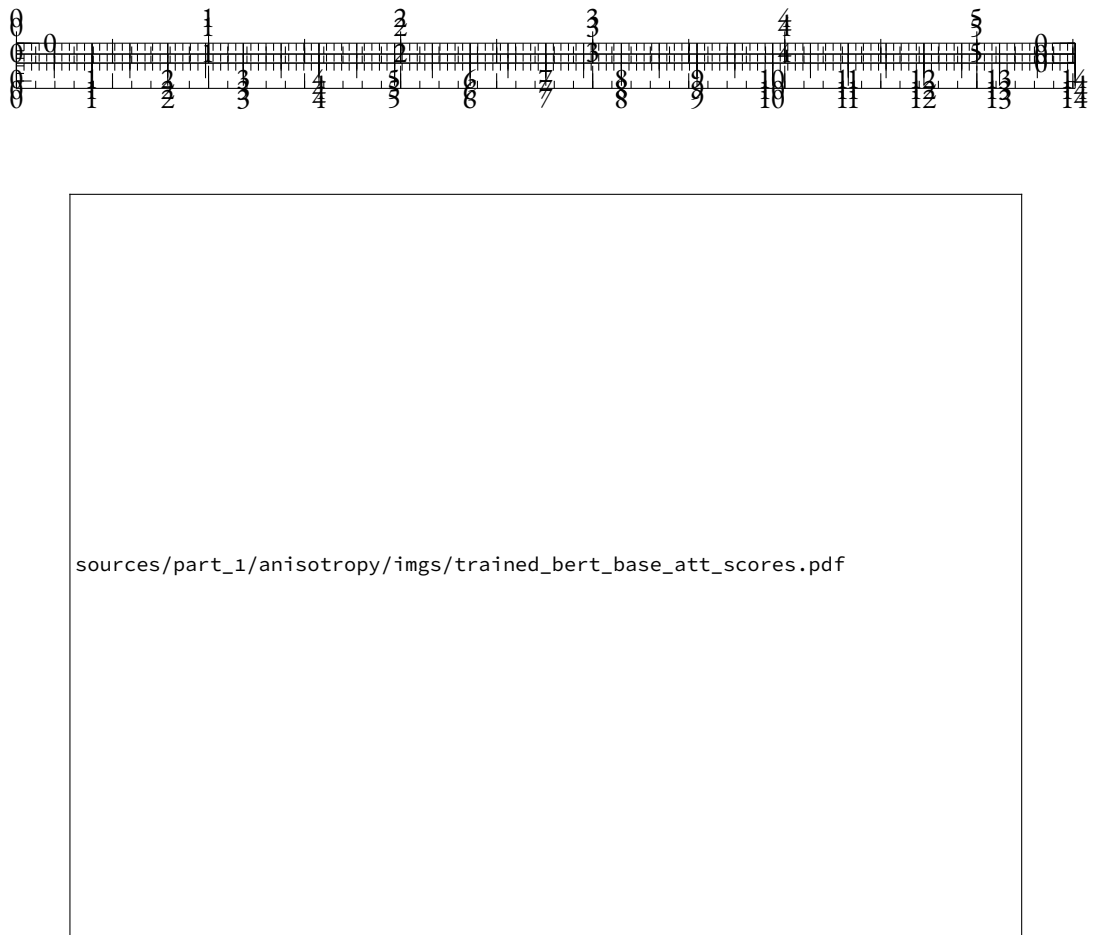
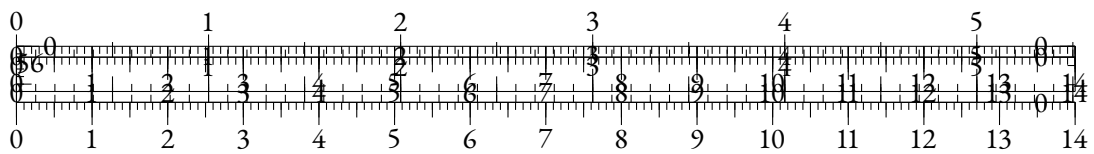


Figure 2.22: Histograms of the pre-softmax attention scores as the input bias norm increases. Other initializations of the layer and of the bias direction  $b_u$  led to a general *increase* of the attention scores instead.



Figure 2.23: Analysis of the self-attention query and key distributions





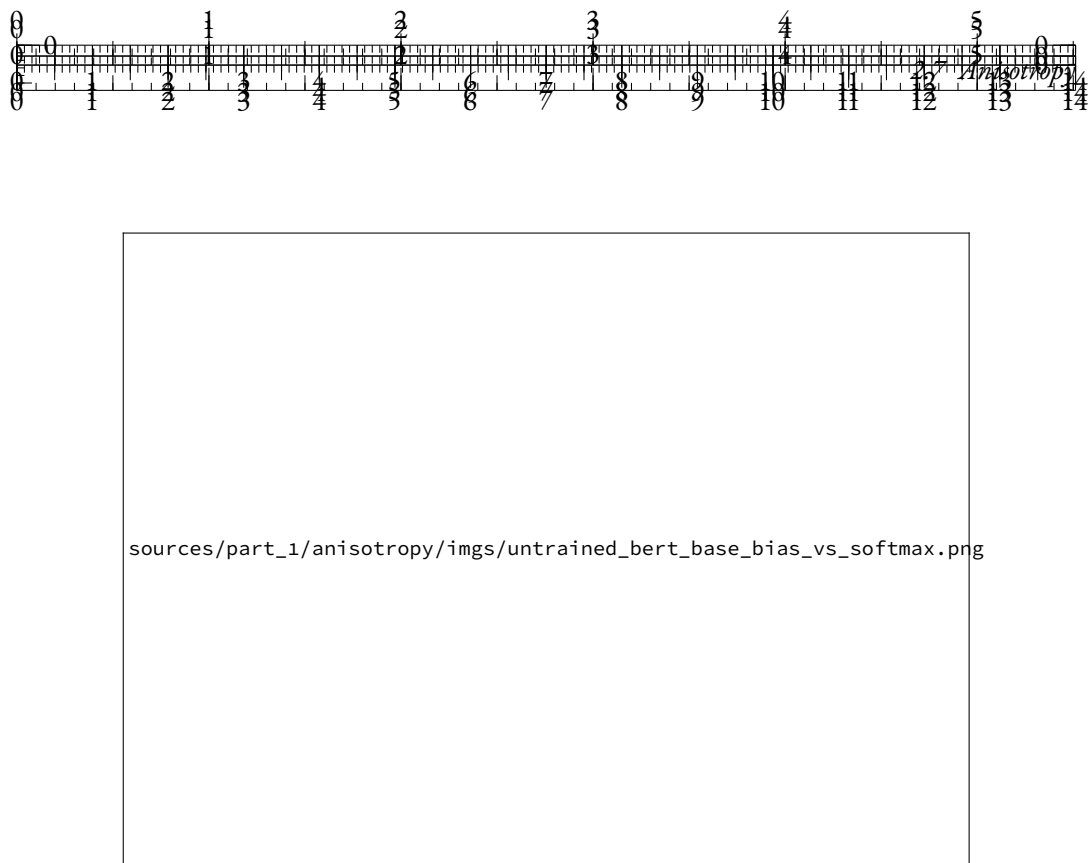
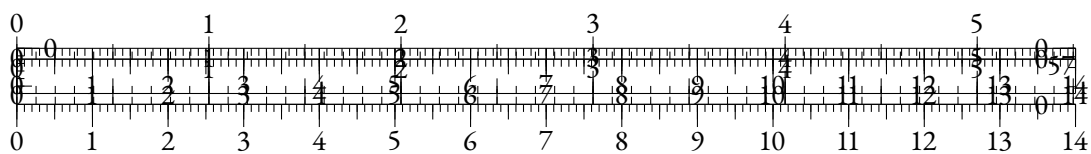


Figure 2.24: Evolution of the self-attention softmax values as the input bias norm increases.



Figure 2.25: Comparison of the extreme values of each sequence averaged over the batch as the bias norm increases.



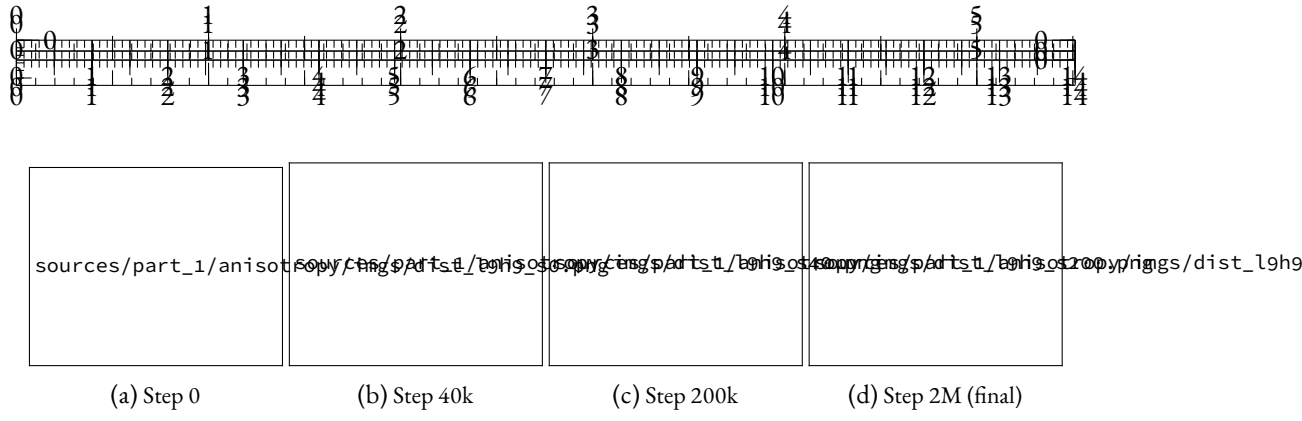


Figure 2.26: Evolution of  $Q_s$  and  $K_s$  distributions along training. Vectors are projected using a common SVD.



Figure 2.27: Evolution of  $\bar{Q}_s$  and  $\bar{K}_s$  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.

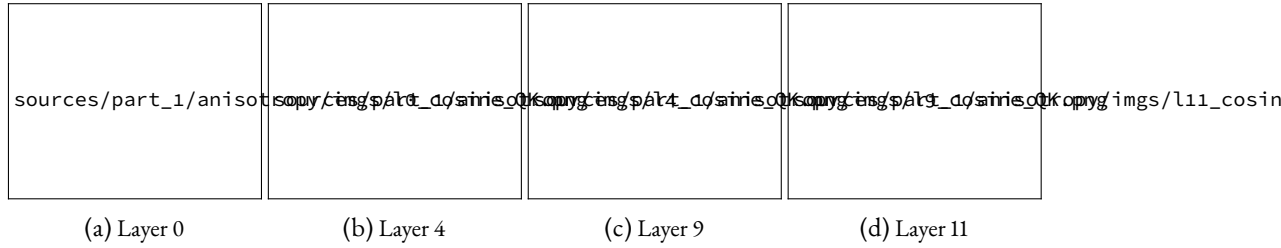
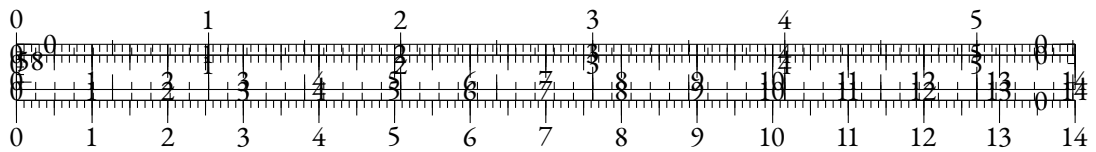


Figure 2.28: Evolution of cosine-similarity between  $\bar{Q}_s$  and  $\bar{K}_s$  along training. Each color represents one self-attention head. Steps are counted in thousands. We generally observe that almost all heads see  $\bar{Q}_s$  and  $\bar{K}_s$  align in common or opposite directions along 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 2.38, and obtain comparable results.



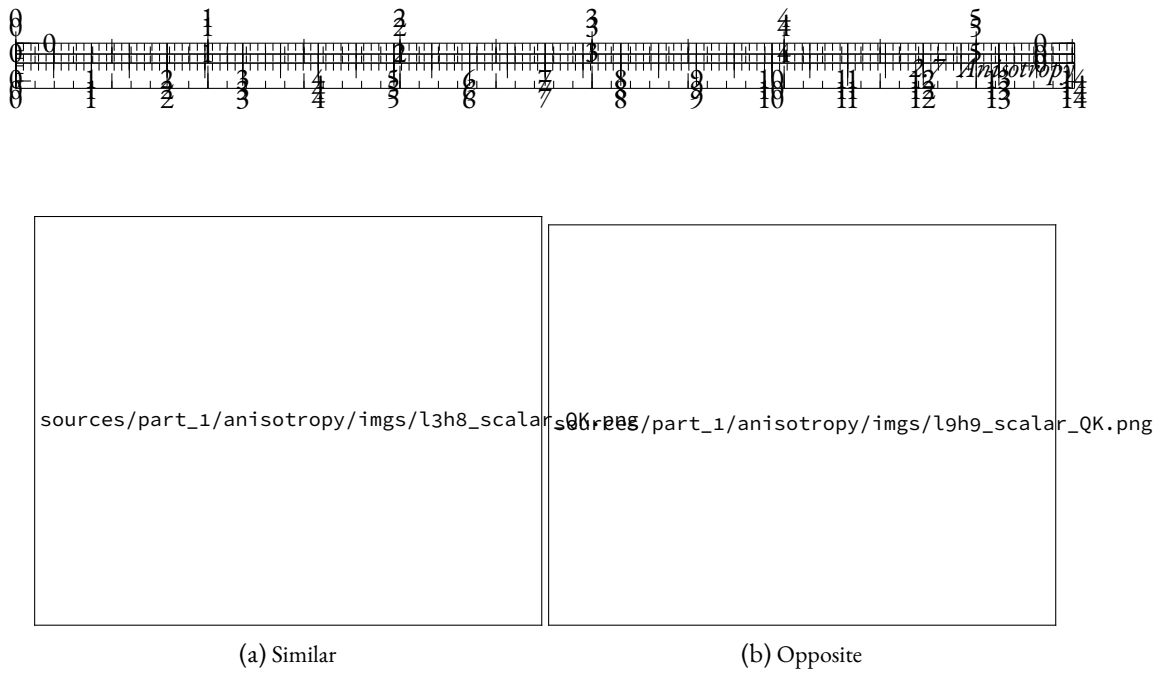
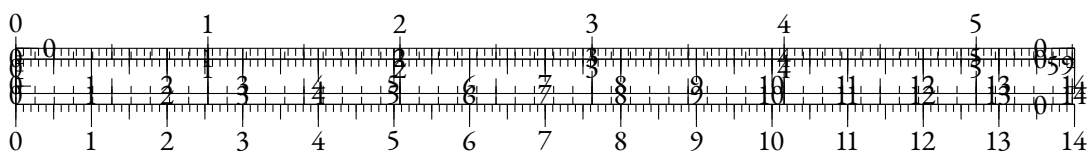


Figure 2.29: Evolution of the scalar product between  $\bar{Q}_s$  and  $\bar{K}_s$  along training. Steps are in thousands.



Figure 2.30: Average entropy of the probability distributions corresponding to self-attention rows along training. Each curve corresponds to one layer.



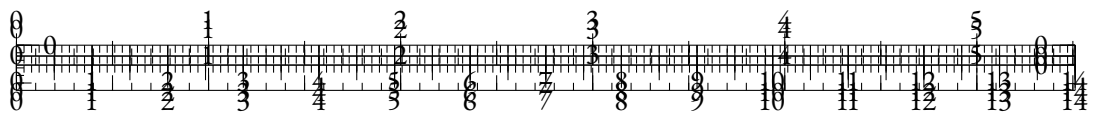
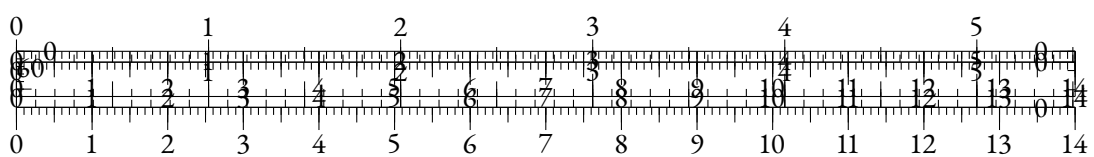


Figure 2.31: p-value of the Pearson correlation test between the norm of the average representation and the cosine-similarity averaged over all layers, across modalities. Models above the red dotted line are not significantly affected by the drift effect.



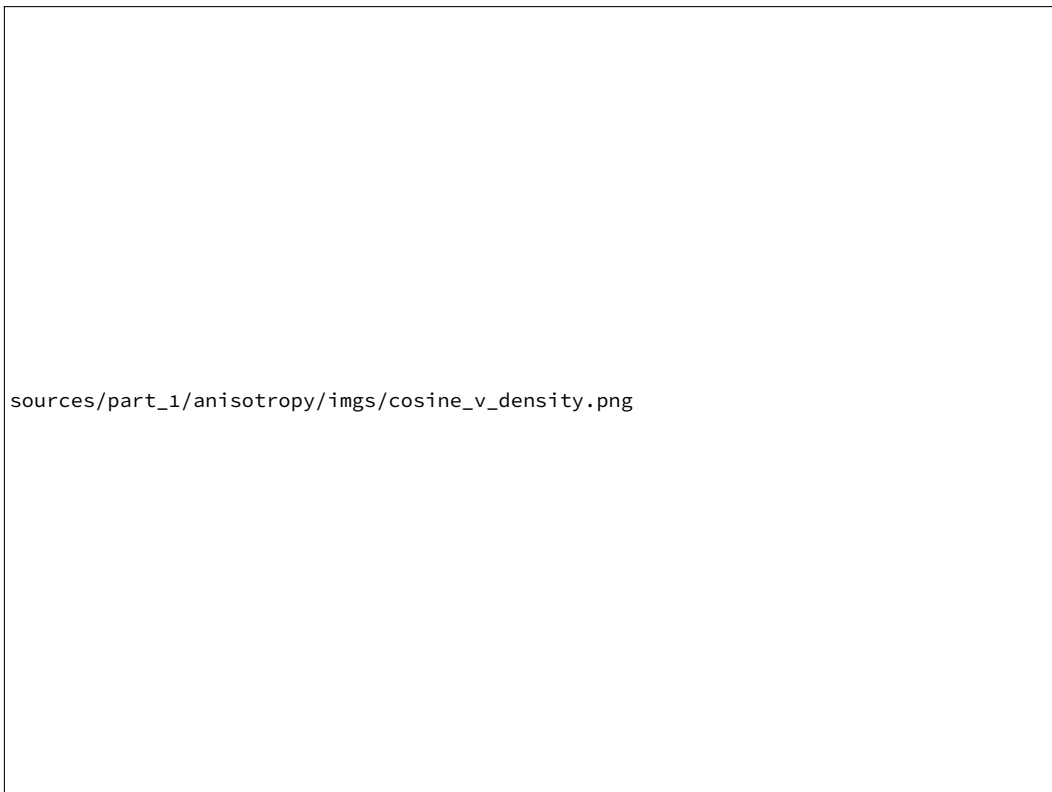
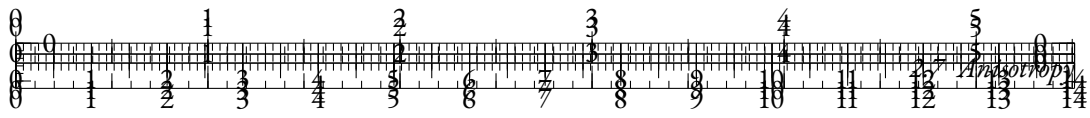


Figure 2.32: Density function of cosine-similarity for a normal distribution as the dimension increases.

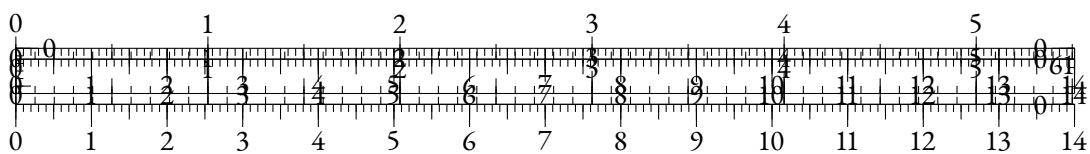




Figure 2.33: 95th quartile of the cosine-similarity distribution on a normal distribution as the dimension increases. We add points for the average cosine-similarity level of Transformers models for several modalities.

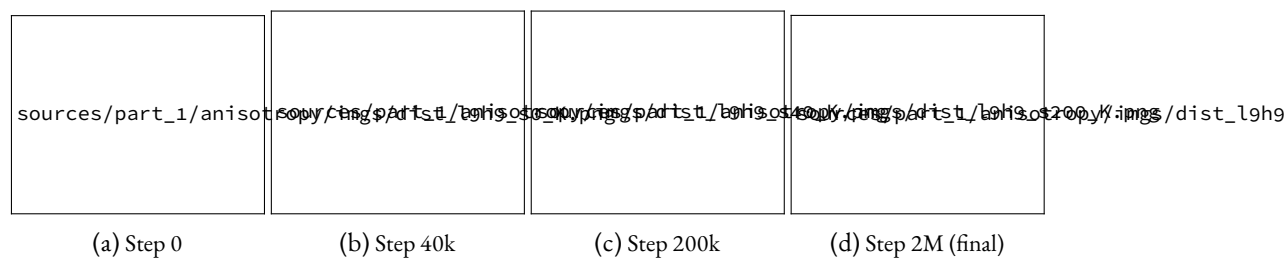
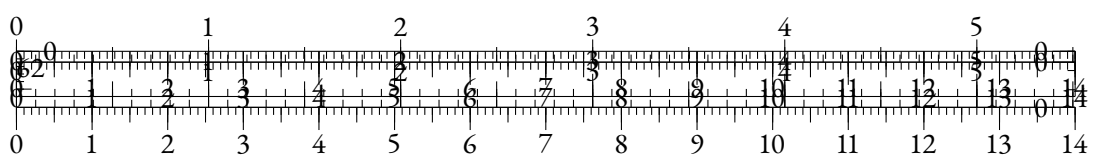


Figure 2.34: Evolution of  $Q_s$  and  $K_s$  distributions along training. Vectors are projected using the SVD computed on  $K_s$ .



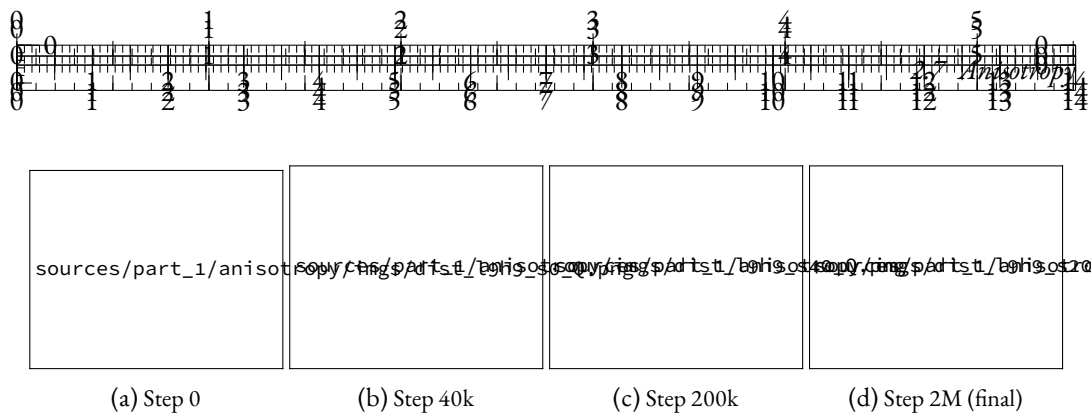


Figure 2.35: Evolution of  $Q_s$  and  $K_s$  distributions along training. Vectors are projected using the SVD computed on  $Q_s$ .

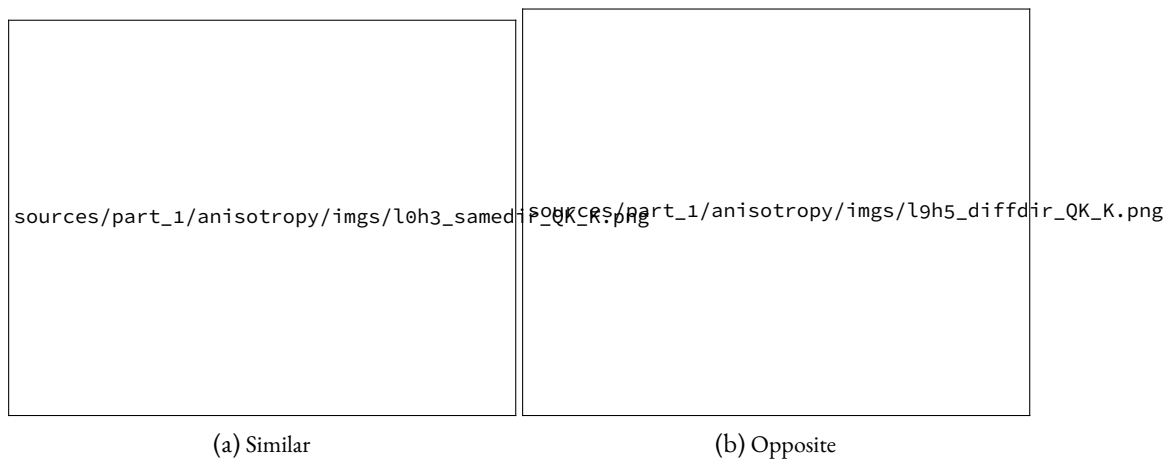
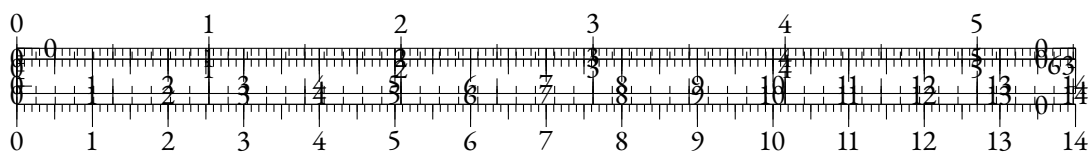


Figure 2.36: Evolution of  $\bar{Q}_s$  and  $\bar{K}_s$  along training for two different heads in the network, projected via the SVD of  $K_s$ .



Figure 2.37: Evolution of  $\bar{Q}_s$  and  $\bar{K}_s$  along training for two different heads in the network, projected via the SVD of  $Q_s$ .



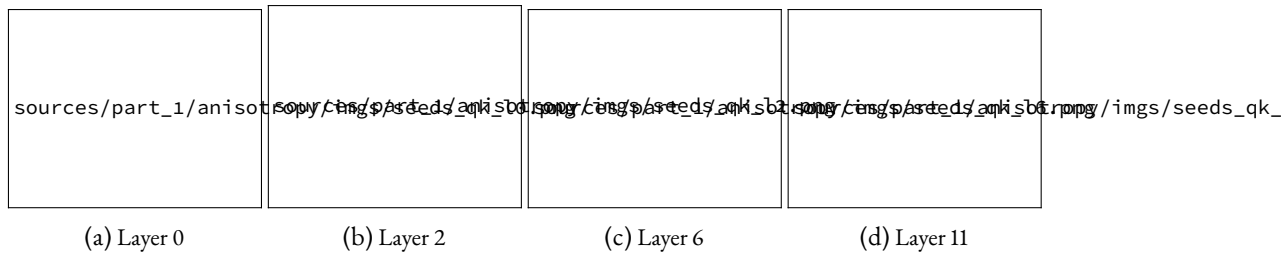
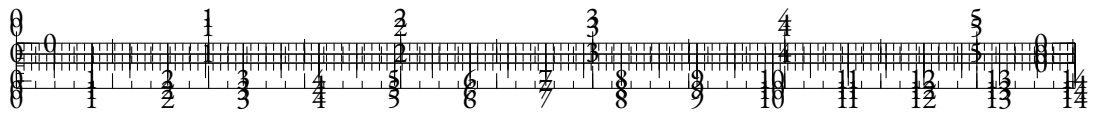
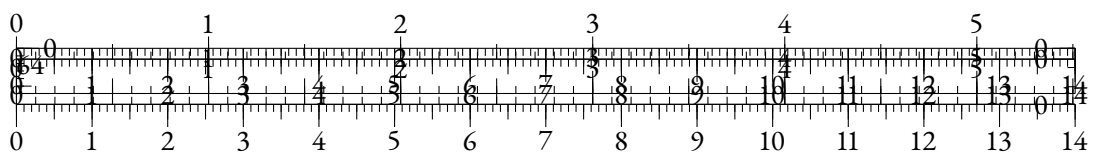
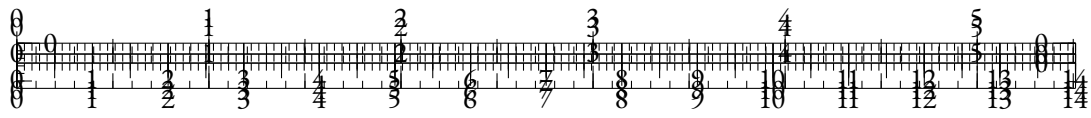


Figure 2.38: Evolution of cosine-similarity between  $\bar{Q}_s$  and  $\bar{K}_s$  along training for various initialization seeds. Representations are concatenated across heads, and each color represents one seed of the MultiBERT models. We observe similar trends across seeds.



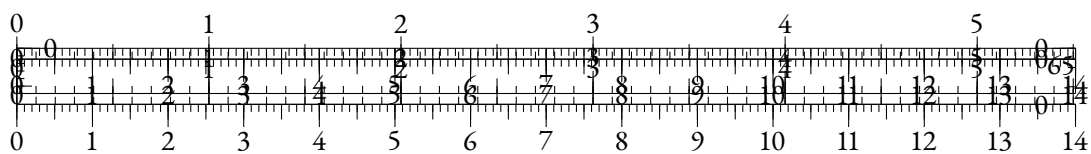




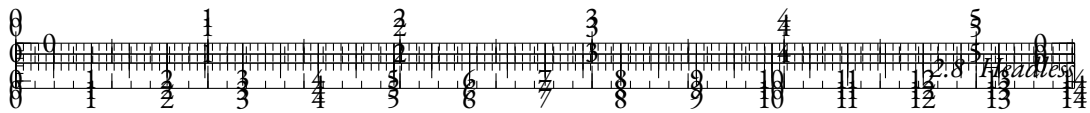
## PART III

### A GOOD PART

You can also use parts in order to partition your great work into larger ‘chunks’. This involves some manual adjustments in terms of the layout, though.







## 2.8 HEADLESS

### 2.8.1 INTRODUCTION

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 paper, 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).

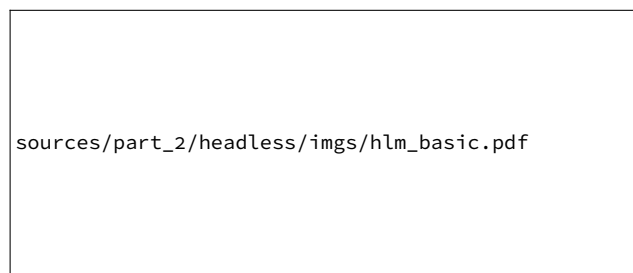
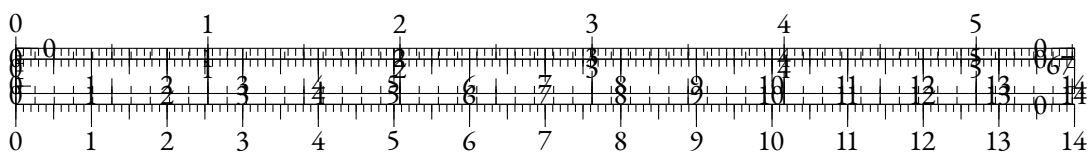
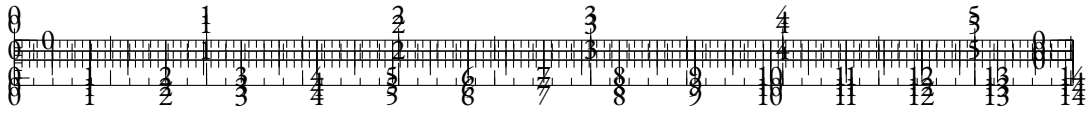


Figure 2.39: Masked Headless Language Modeling (HLM) using Contrastive Weight Tying. The CWT objective aims to contrastively predict masked input representations using in-batch negative examples.

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>3</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 article:

- 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.

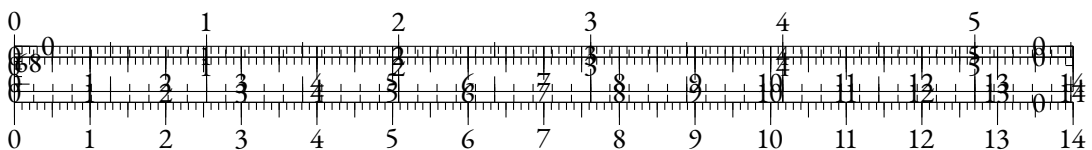
### 2.8.2 RELATED WORK

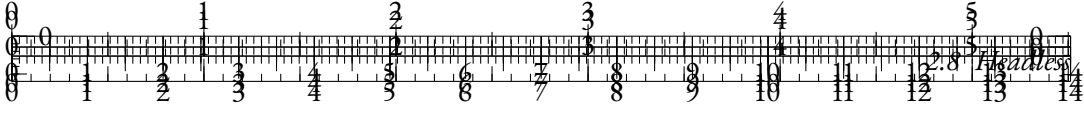
**EFFICIENT PRE-TRAINING** With the dawn of pretrained language models, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), GPT-2 (Radford et al., 2019) or T5 (Raffel et al., 2020a), improving training efficiency has become an important stake in NLP. Subsequent works have focused on changing the training objectives to improve performance. ELECTRA (Clark et al., 2020b) uses Replaced Token Detection as the unsupervised training task, and substantially improves data-efficiency, compute-efficiency, and downstream performance. Their work has also been extended using energy-based models (Clark et al., 2020a) or disentangled weight sharing (He et al., 2021).

**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., 2019) 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

<sup>3</sup>Our pretraining and fine-tuning code is published in <https://github.com/NathanGodey/headless-lm>





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 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.

**TOKENIZATION AND FREQUENCY** The importance of tokenization for language models has been discussed by several works (Rust et al., 2021; Zouhar et al., 2023). As discussed in Zouhar et al. (2023), tokenization choices impact token probability distributions both at contextual and general scales. It has been shown that skewed token distributions can impact the quality of representations (Gao et al., 2019b; Zhou et al., 2021c; Puccetti et al., 2022; Yu et al., 2022). Removing the language modeling head could mitigate these issues. In the case of multilingual models, Liang et al. (2023) have shown that increasing the vocabulary size leads to better performance, at the cost of added time and memory complexity.

### 2.8.3 METHOD

#### CLASSICAL FRAMEWORK

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

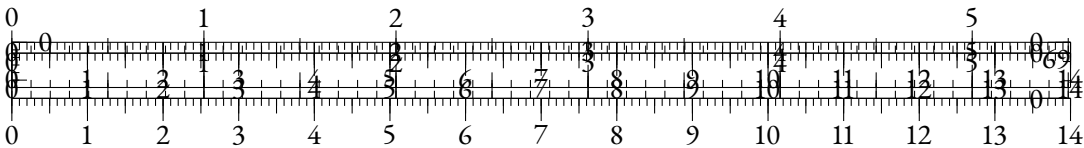
A classical language modeling approach consists in selecting a subset of tokens  $X_S = (x_{i,j})_{i,j \in S}$ , and then estimating a probability distribution over the token vocabulary for these tokens from the  $(\tilde{x}_{i,j})$  sequences, using  $e_\theta$  and  $T_\theta$ . Learning occurs as  $X_S$  is partially altered in  $(\tilde{x}_{i,j})$  (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  $X_S$ .

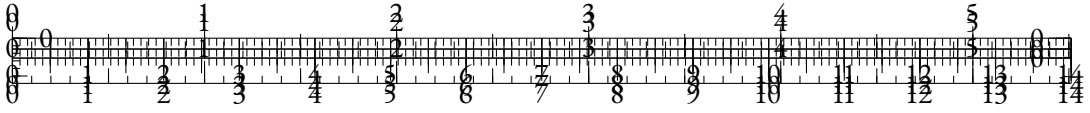
A trick that has been used in many such approaches relies on using  $e_\theta$ 's transpose ( $e_\theta^T$ ) as a projection from the output space of  $T_\theta$  to  $\mathbb{R}^V$ . This approach, called weight tying, can be written for a given sequence at index  $i \in [1, N]$  as:

$$\hat{p}_{i,j} = \text{softmax}(e_\theta^T(T_\theta(e_\theta(\tilde{x}_i))_j))$$

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

$$\mathcal{L}(\theta, X, \tilde{X}) = -\frac{1}{|S|} \sum_{i,j \in S} \mathbf{1}_{x_{i,j}} \cdot \log(\hat{p}_{i,j})$$





## 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  $e_{\theta}(x_{i,j})$  and the output representation at the same position  $o_{i,j}^{\theta} = T_{\theta}(e_{\theta}(\tilde{x}_i))_j$ , 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:

$$\mathcal{L}_c(\theta, X, \tilde{X}) = -\frac{1}{|\mathcal{S}|} \sum_{i,j \in \mathcal{S}} \frac{e^{o_{i,j}^\theta \cdot e_\theta(x_{i,j})}}{\sum_{k,l \in \mathcal{S}} e^{o_{i,j}^\theta \cdot e_\theta(x_{k,l})}}$$

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 paper, 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.

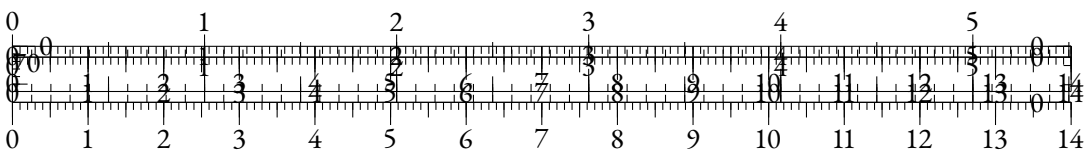
## THE CASE OF DECODERS: CAUSAL FINE-TUNING

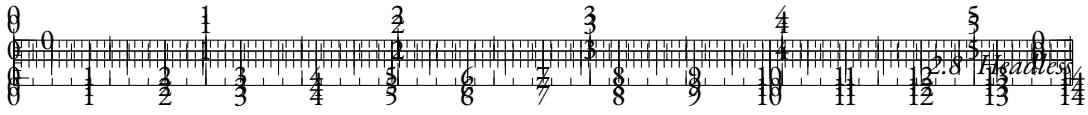
We can easily adapt the Causal Language Modeling (CLM) objective using the Contrastive Weight Tying approach. Negative samples correspond to every input embedding at a different position in the batch. However, the resulting model is not directly able to generate text, as it has no projection head towards  $\mathbb{R}^V$ . 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 2.4). 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.

## 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  $O(KDV)$  operations while our headless approach performs  $O(K^2D)$  operations<sup>4</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

<sup>4</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  $\mathcal{S}_N = \mathcal{S}$ .





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  $K < V$ .

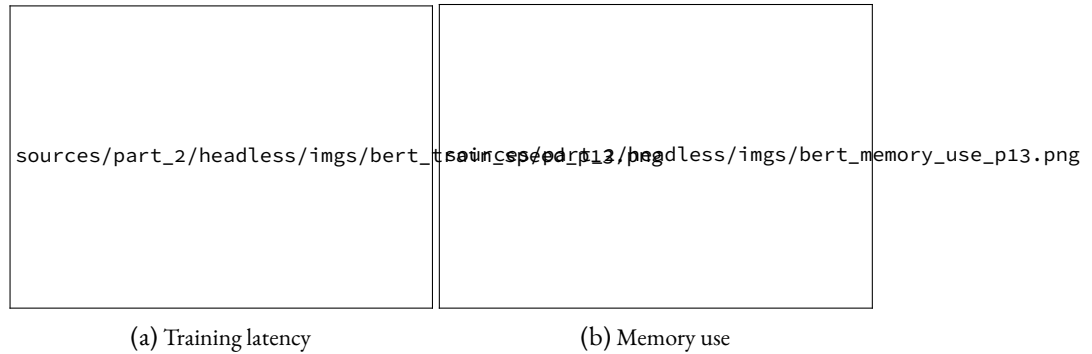


Figure 2.40: Comparison of time and memory complexities of a BERT-base model on a single RTX 8000 GPU.

In Figure 2.40, 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.

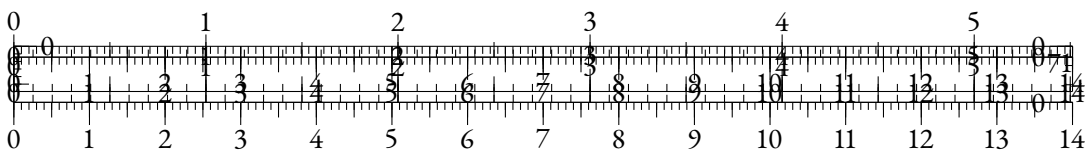
## 2.8.4 EXPERIMENTS

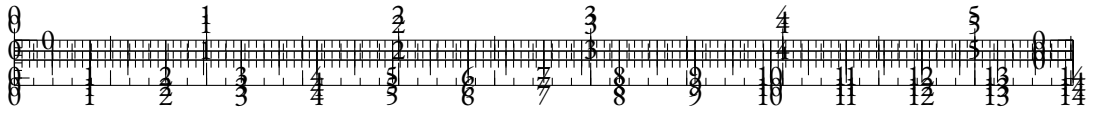
We use the Contrastive Weight Tying objective for medium-scale pre-training experiments in different contexts. We focus on monolingual encoder and decoder architectures, but we also train one multilingual encoder as we believe the uniformity brought by our contrastive objective may improve cross-lingual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

### 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., 2019). 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 Section 2.8.4.

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 (Lefauveux 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





MLM type	Tokens (B)	GPU hours	MRPC	COLA	STS-B	SST2	QNLI	QQP	MNLI	Avg.
Vanilla	4.1	989	<u>85.87</u>	54.66	83.7	92.45	88.38	89.57	82.4	82.43 ( $\pm 0.12$ )
Headless	4.1	444	85.31	58.35	84.54	<b>93.23</b>	89.49	89.62	82.54	83.29 ( $\pm 0.15$ )
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 2.2: 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 2.3: 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.

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.

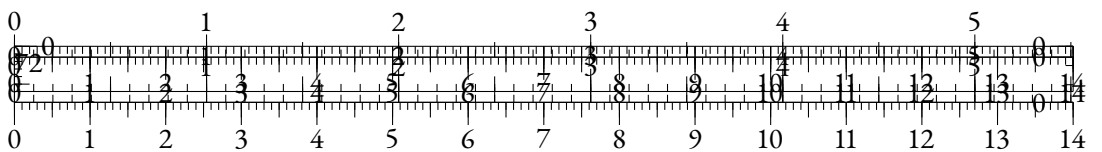
We evaluate on the GLUE benchmark, where we exclude the RTE dataset due to high standard deviations in the obtained scores. We fine-tune our models for 10 epochs on every dataset, and compute validation metrics once every fine-tuning epoch. We use the AdamW optimizer with a learning rate of  $10^{-5}$ , a weight decay of 0.01 and a balanced cross-entropy loss objective. See Section 2.8.5 for more details.

In Table 2.2, 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 2.41. 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.

## 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.

As mentioned in Section 2.8.3, 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-





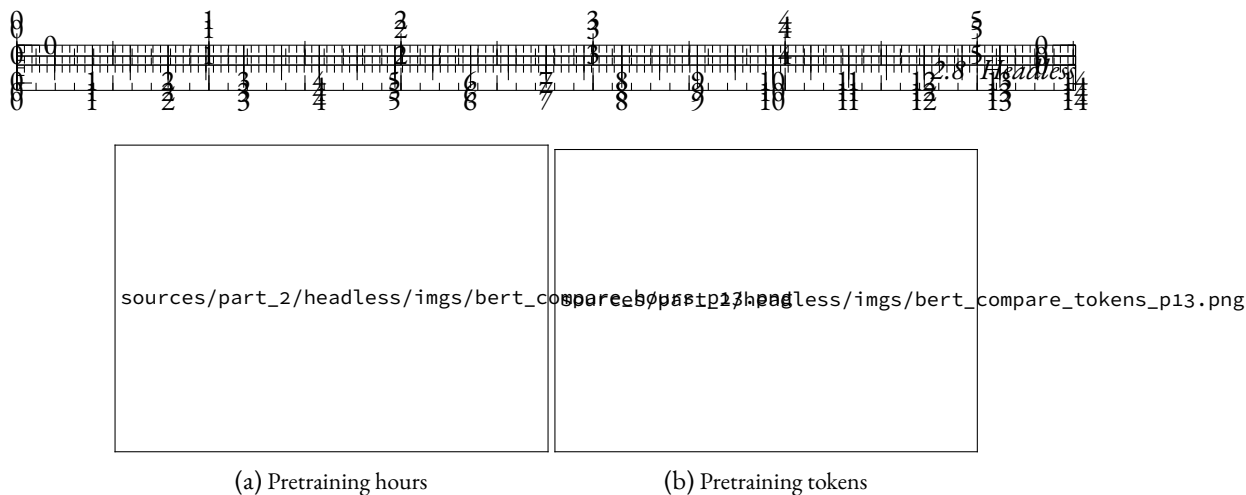


Figure 2.41: Comparison of GLUE average scores along pretraining.

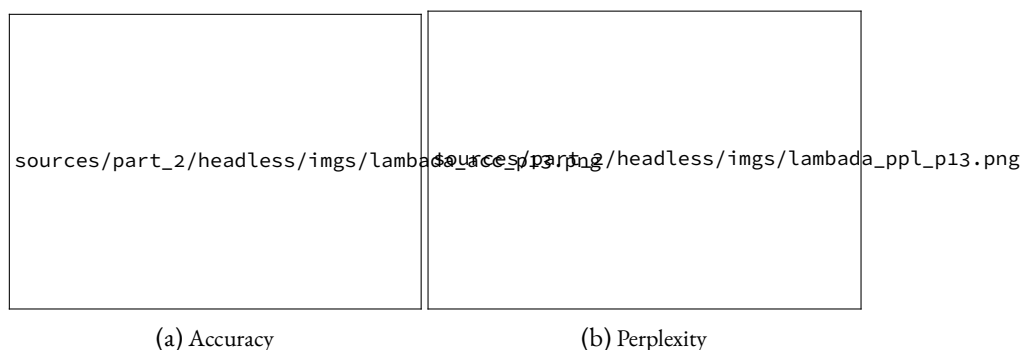


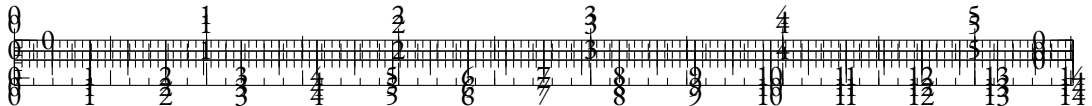
Figure 2.42: Comparison of LAMBADA metrics along pretraining. We display results for vanilla causal language modeling and headless models before and after causal LM fine-tuning. The pretraining token count for the fine-tuned HLM takes fine-tuning tokens into account.

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 2.42.

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 2.4. 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 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 2.5.

In Table 2.5, 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).



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

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

Table 2.5: 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. <sup>†</sup>: best scores that are significantly better than the second best score according to a one-tailed t-test with power 0.95.

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 study shows that language generation needs to be considered as a *downstream task* for HLMs, as they are designed to generate representations instead of words.

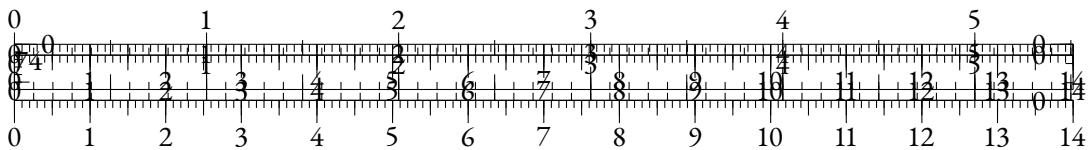
### 2.8.5 MULTILINGUAL ENCODER

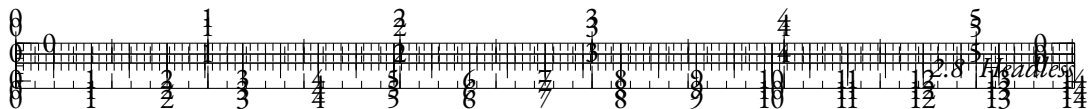
In this section, we pretrain small multilingual MLMs and evaluate their performance on the XNLI dataset (Conneau et al., 2018b). Due to compute limitations, we consider architectures similar to the distilled multilingual BERT<sup>5</sup> trained by Sanh et al. (2019). This model has 137M parameters, and uses a vocabulary of 119k tokens. As in Section 2.8.4, 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 Section 2.8.4.

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 Section 2.8.5.

We display final results in Figure 2.43. 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.

<sup>5</sup>Available at <https://huggingface.co/distilbert-base-multilingual-cased>





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

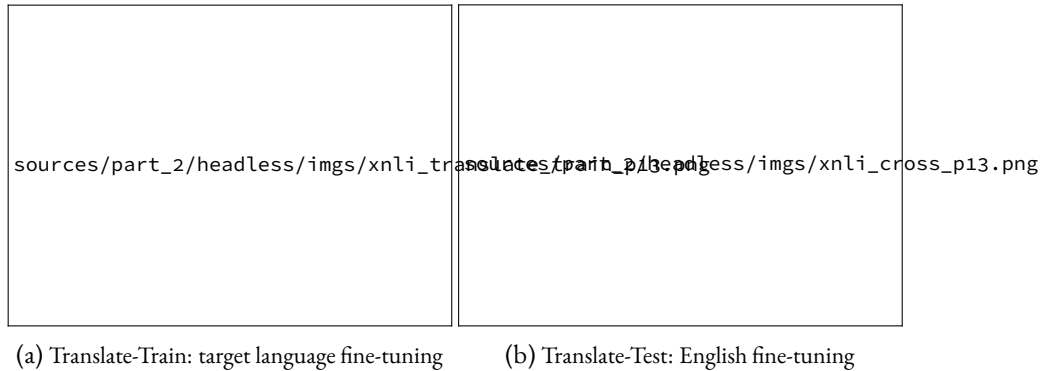


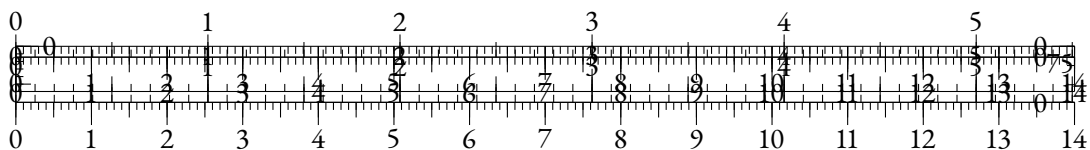
Figure 2.43: Comparison of XNLI average scores along pretraining for different setups. Models are fine-tuned/evaluated in Arabic, German, English, Spanish, French, Hindi and Chinese.

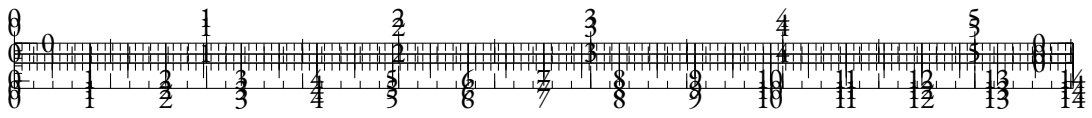
In Figure 2.43, we evaluate the models at intermediate pretraining checkpoints and plot the XNLI average score as a function of used GPU hours. We observe that our HLM finishes training within 45% of the time required by the vanilla mode, and that its performance level outperforms the fully trained vanilla model after only using 5% as much compute in Figure 2.43a, and 22% in Figure 2.43b.

## 2.8.6 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. (2020) and the CC-News dataset (Hamborg et al., 2017). Hyperparameter details can be found in Section 2.8.4. 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.

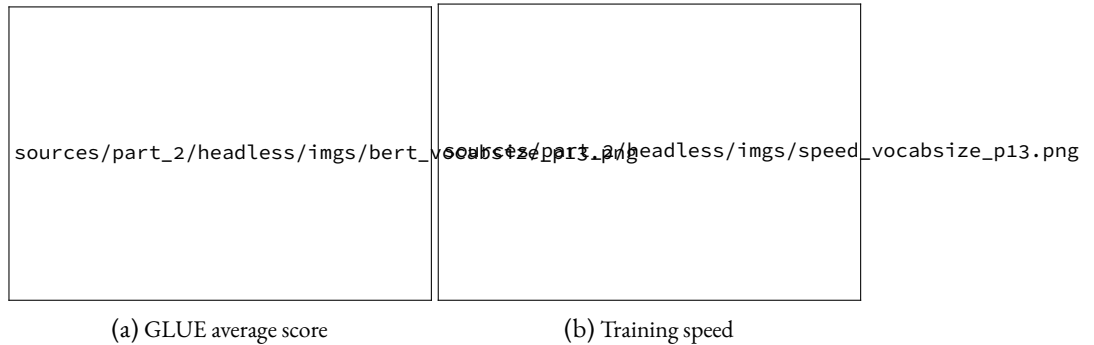


Figure 2.44: Comparison of downstream performance and training speed for small models trained using different token vocabulary sizes.

Figure 2.44 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 2.44b 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 2.8.3, 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 Section 2.8.4. HLMs are fine-tuned similarly to Section 2.8.4.

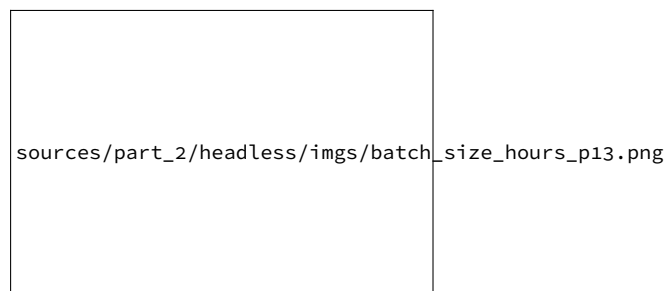
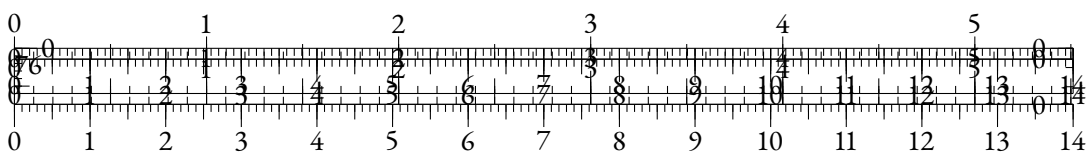
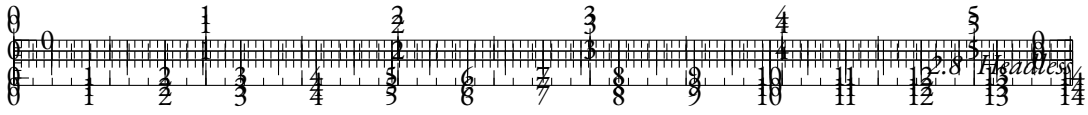


Figure 2.45: LAMBADA accuracy along pretraining for different batch sizes.





In Figure 2.45, 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 2.7, we conduct an ablation study by training small models using the hyperparameters described in Section 2.8.4 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.

Objective	Parameters	Throughput $\uparrow$	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 2.7: 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).

## CONCLUSION

In this paper, 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.

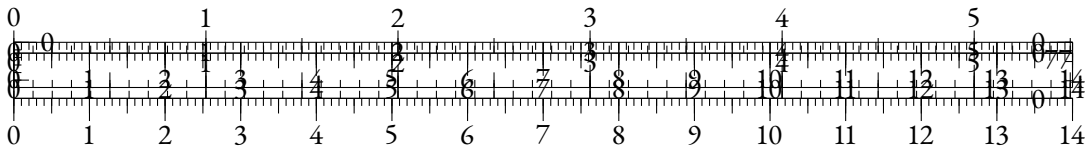
Using our contrastive objective, we pretrain headless monolingual and multilingual encoders, and a headless monolingual decoder. We demonstrate that headless pretraining is significantly more compute-efficient, data-efficient, and performant than classical predictive methods.

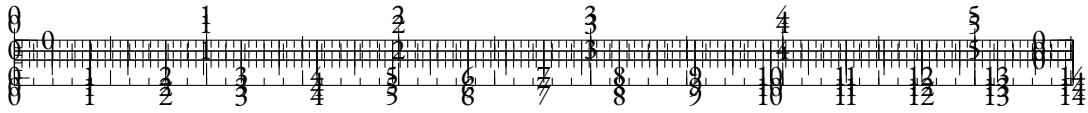
A major advantage of our approach is that it enables the use of very large token vocabularies at virtually no increased cost. We believe that this paper paves the way for the exploration of contrastive techniques as a replacement of cross-entropy based pretraining objectives for NLP.

## ACKNOWLEDGEMENTS

We thank our colleagues Arij Riabi and Roman Castagné for their advice and for the helpful discussions. We are grateful to Robin Algayres for his enlightening question “*But what is the difference with softmax?*”, in the hope that this paper is a satisfying answer.

This work was funded by the last author’s chair in the PRAIRIE institute, itself funded by the French national agency ANR as part of the “Investissements d’avenir” programme under the reference ANR-19-P3IA-0001.





This work was granted access to the HPC resources of IDRIS under the allocation 2023-AD011013680R1 made by GENCI.

### 2.8.1 MODELING CONSIDERATIONS

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 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 2.46.

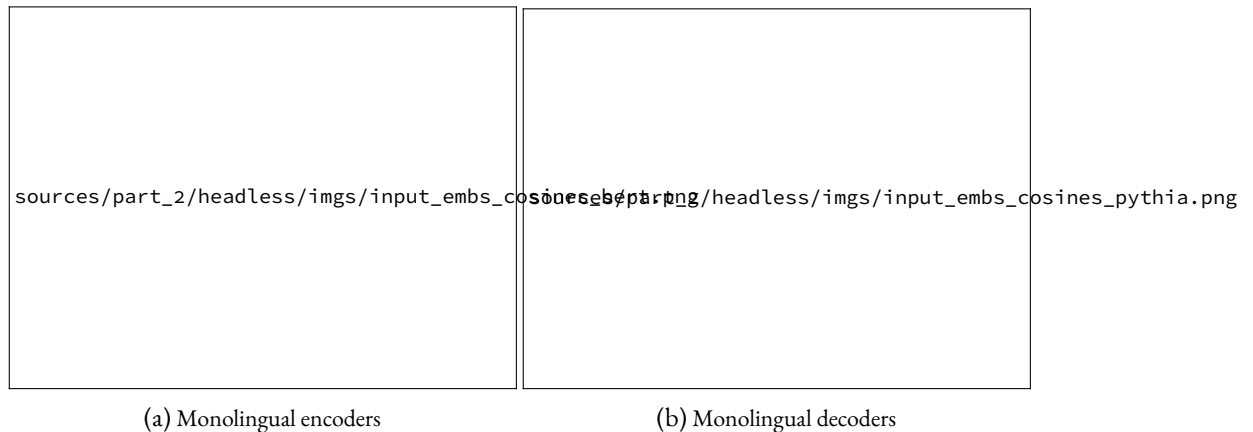
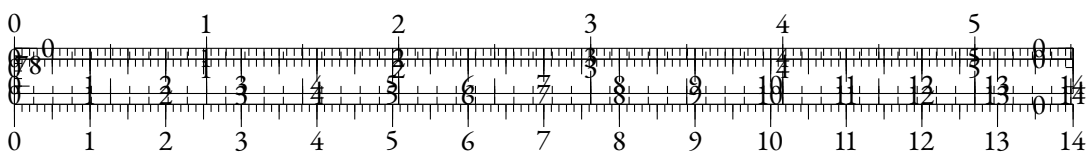


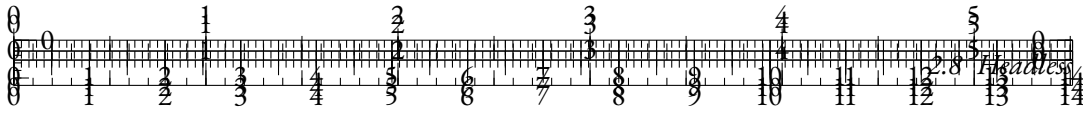
Figure 2.46: Cosine-similarity distributions for pairs of WordNet synonyms.

In Figure 2.46, 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.

### 2.8.2 LIMITATIONS

One key limitation of this paper 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 paper 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 in this paper are still sufficiently promising to be communicated and useful to the community. We leave the scaling of these techniques to future work.

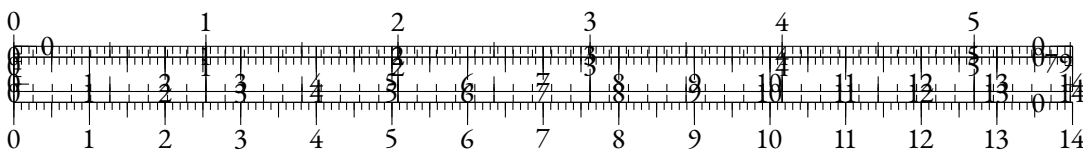
It could be opposed to this paper that as architectures grow in size, the proportion of compute that is associated with the output vocabulary projection shrinks. While we acknowledge that this effect may reduce the advantage of HLMs in terms of training throughput, our experiments show that HLMs are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

### 2.8.3 ETHICS STATEMENT

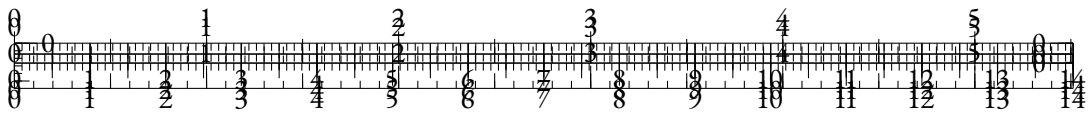
To the best of our knowledge, this paper does not raise any specific ethical concern that is not already inherent to the open-data pre-training paradigm. Our results on the CrowS-Pairs dataset indicate that headless language modeling may mitigate some of the biases that are measured in this task. Due to considerations that are discussed in Zhou et al. (2021c), and for reasons evoked in Section 2.9.6, we believe that alternatives to cross-entropy as an objective for language modeling could mitigate some of the biases that are observed in LLMs, and hope that our work can pave the way for such alternatives.











## 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 2.12: Pre-training hyperparameters used for the small monolingual decoders used in Figure 2.45. These models rely on the GPT-2 architecture with a few changes. These changes scale down the model size to 11M parameters.

## 2.8.5 FINETUNING HYPERPARAMETERS

### BALANCED CROSS-ENTROPY

We have 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  $(w_c)_{c \in [1, C]}$ . We can group the cross entropy loss  $\mathcal{L}_{ce}$  as such:

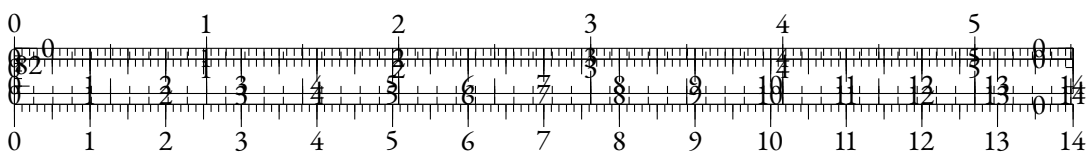
$$\mathcal{L}_{ce}(X, Y) = \sum_{c=1}^C \mathcal{L}_c(X, Y)$$

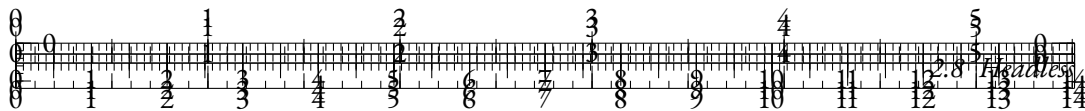
where

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

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

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





In practice, we approximate the  $(w_c)$  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 2.13).

Optimizer	AdamW
Learning rate	5e-6
Weight decay	0.01
Batch size	32
LR schedule	Constant
Linear warm-up	10%
Epochs	10

Table 2.13: Fine-tuning hyperparameters for monolingual encoder models trained with regular cross-entropy on the GLUE benchmark.

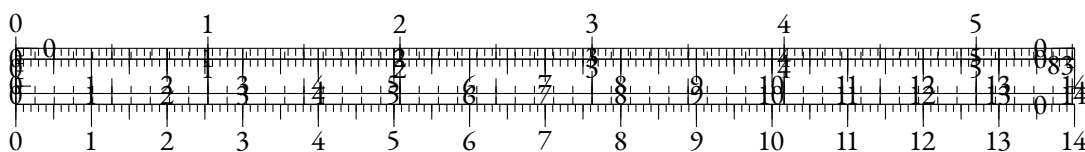
MLM type	MRPC	COLA	STS-B	SST2	QNLI	QQP	MNLI	Avg.
Vanilla	<b>86.27</b>	49.33	82.06	92.37	88.62	89.49	82.35	81.5 ( $\pm 0.14$ )
Headless	85.8	<b>56</b>	<b>84.85</b>	<b>93.23</b>	<b>89.67</b>	<b>89.77</b>	<b>83.05</b>	<b>83.19</b> ( $\pm 0.09$ )

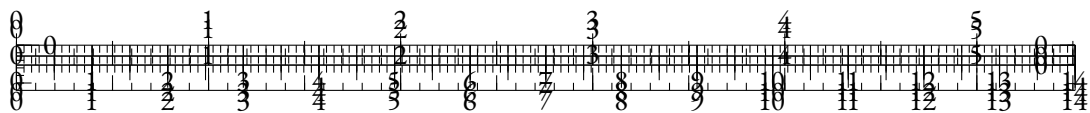
Table 2.14: Results of Masked Language Models (MLMs) on the dev sets of the GLUE benchmark for the regular cross-entropy loss. Results are averaged over 3 runs.

## 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 2.15: Fine-tuning hyperparameters for monolingual encoder models trained with balanced cross-entropy on the GLUE benchmark.





## 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 2.16: Fine-tuning hyperparameters for the headless monolingual decoder model using the causal language modeling objective.

## 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 2.17: Fine-tuning hyperparameters for the multilingual encoder models in Translate-Train and Translate-Test scenarios.

## 2.8.6 REPRESENTING SYNONYMS

In this section,

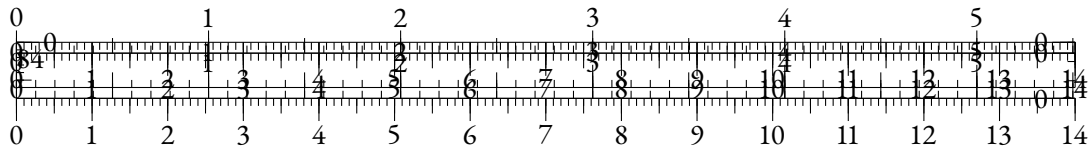
## 2.8.7 IMPLEMENTATION

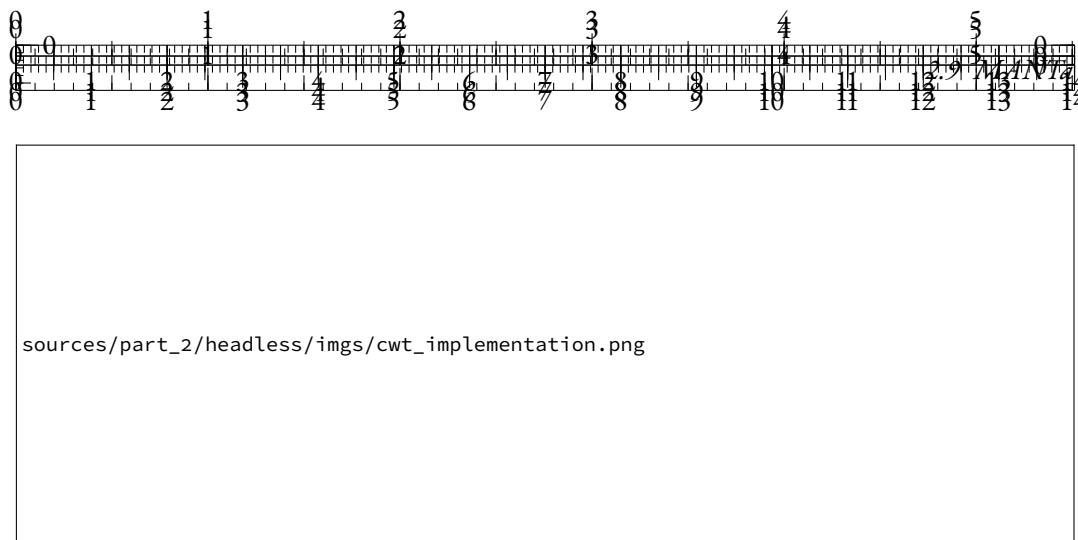
The figures were generated using the Carbon tool (<https://carbon.now.sh/>).

## 2.9 MANTA

### 2.9.1 INTRODUCTION

In order to improve Language Models (LMs), the Natural Language Processing field has removed most of the system-induced biases in the last few years. For instance, practices that were once standard such as lemmatization, stemming and feature engineering have progressively disappeared in favor of general architectures trained on huge amounts of data, learning end-to-end which features may be leveraged to attain better performances. However, one essential part of LMs has seen little evolution: tokenization. Tokenizers convert sequences of characters into sequences





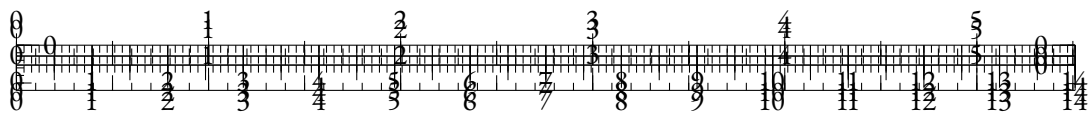
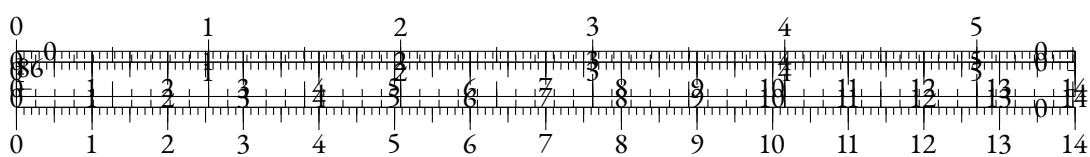


Figure 2.48: PyTorch implementation of the computation of the training loss for headless causal LMs. The implementation of the MLM equivalent is straightforward.



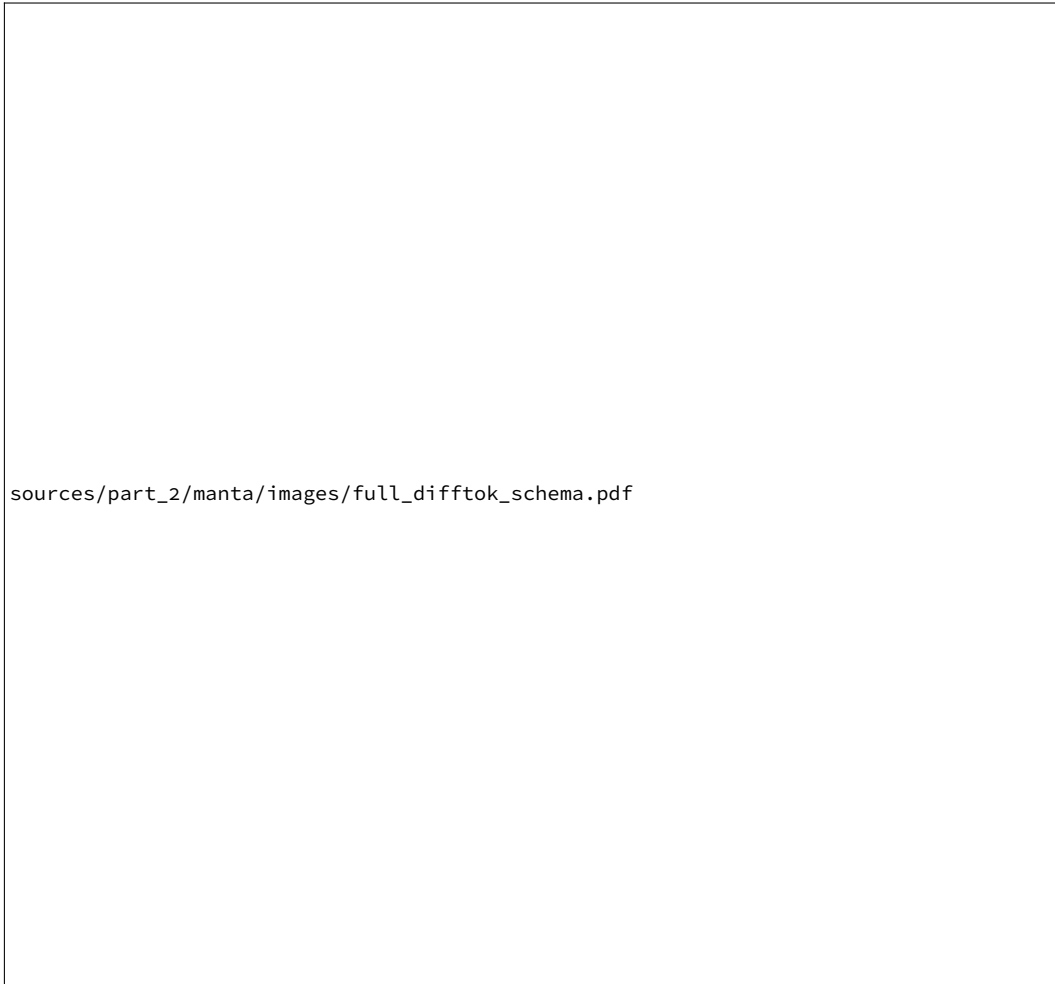
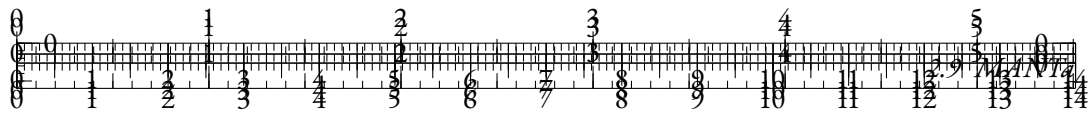
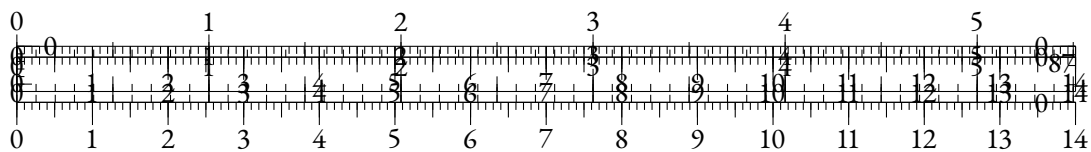
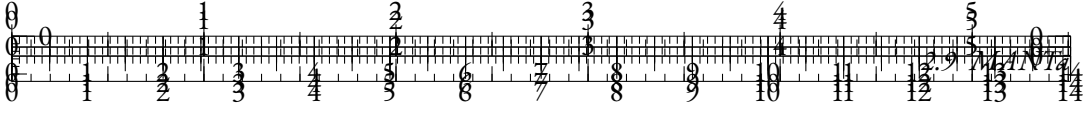


Figure 2.49: The differentiable tokenization scheme of MANTa-LM. Input bytes are first assigned a *separation probability* using a Sliding Window Attention Transformer. These probabilities are used to compute the contribution of each byte embedding in the pooled representations of the *blocks*. The block embeddings are fed to the Encoder-Decoder layers which predict the masked bytes. All the components are optimized with the LM objective.









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.

### 2.9.3 MANTa

#### 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  $p_{F_i}$  to each input byte  $b_i$  of being a frontier;<sup>6</sup>
- Building a byte-block unnormalized joint distribution using the frontier probabilities  $(p_{F_i})_{i \in [1, L]}$  corresponding to a soft assignment from bytes to blocks;
- Pooling byte representations for each block  $B_j$  weighted by the probability of each byte to belong in the current block  $P(b_i \in B_j)$ .

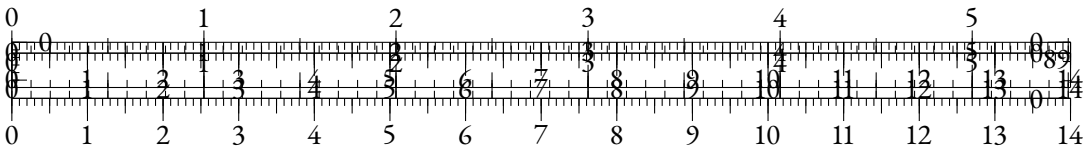
This process results in a sequence of embeddings that can be given directly to the encoder-decoder model. We provide an overview of the entire model in Figure 2.49. We also summarize the process of mapping byte embeddings to block embeddings in appendix 2.9.4.

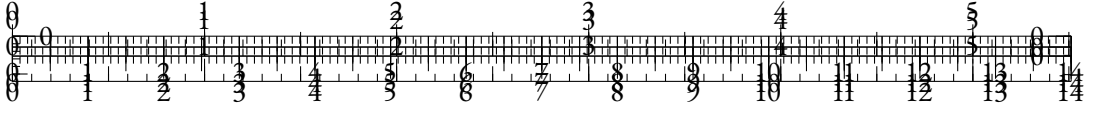
#### PREDICTING SUBWORD FRONTIERS

Our frontier predictor consists in a parameterized module mapping each byte  $b_i$  to the probability of being a block frontier  $p_{F_i}$ . In a first part, we embed each byte  $b_i$  to an embedding  $e_{b_i}$ . Working with bytes instead of characters allows modeling a larger array of symbols while having very small embedding matrices with  $256 \times \text{hidden size}$  parameters. 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.

<sup>6</sup>  $F$  in  $p_{F_i}$  stands for *Frontier*.





### MODELING THE BYTE-BLOCK ASSIGNMENT

Once the frontier probabilities  $(p_{F_i})_{i \in [1, L]}$  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  $(B, b_i)$ , the slot random variables for each byte  $b_i$ , describing the position of the block containing  $b_i$  in the block sequence. In other words, the event  $(B = k, b_i)$  describes the fact that the  $i$ -th byte belongs in the  $k$ -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  $(B, b_i)$  as a *cumulative sum* of the random variables  $F_i$ : the position of the block in which a byte belongs is exactly the number of frontier bytes before this one.

Since  $F_i \sim \mathcal{B}(p_{F_i})$ , 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:  $(B, b_i) \sim \mathcal{PB}((p_{F_k})_{k \leq i})$ . 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  $(B, b_i)$  probability mass function:

$$\forall k \in [1, L_B], P(B = k, b_i) \simeq P_{k,i} \triangleq \frac{1}{Z} G_{\mu_i, \sigma_i}(k)$$

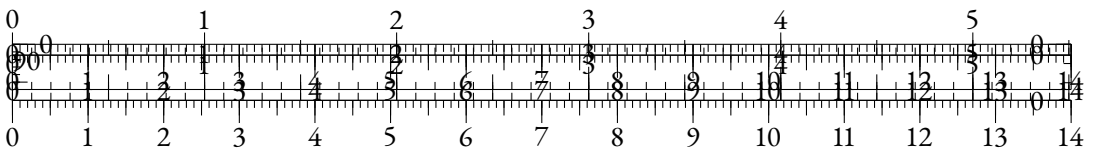
where  $Z = \sum_{1 \leq k \leq L_B} G_{\mu_i, \sigma_i}(k)$  is a normalization term, and:

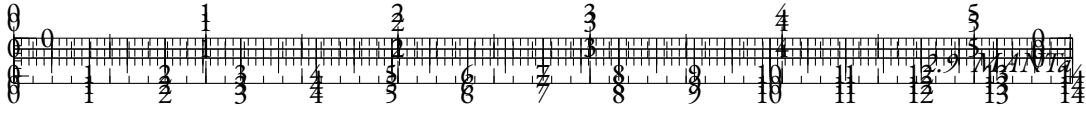
$$\begin{cases} L_B = \min(L, (\mu_L + 3\sigma_L)) \\ \mu_i = \sum_{k=1}^i p_{F_k} \\ \sigma_i = \sqrt{\sum_{k=1}^i p_{F_k}(1 - p_{F_k})} \end{cases} \quad (2.5)$$

We denote by  $P_{k,i}$  the approximation of the probability of membership of the byte  $i$  to block  $k$ . We display an example of this map at different steps during training in Figure 2.50. We truncate the block sequences after  $(\mu_L + 3\sigma_L)$  since all the probabilities beyond this position are negligible.

### 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  $k \in [1, L_B]$ , this map actually provides an unnormalized contribution  $(P_{k,i})_{i \in [1, L]}$  of each byte in this block. We can then use the byte embeddings  $e_b$  from the frontier predictor described in Section 2.9.3 and, for the  $k$ -th block, build a block embedding where each byte  $b_i$  contributes based on its probability of being in this block  $P_{k,i}$ .





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

Table 2.18: Results on dev sets for the GLUE benchmark for small models following our pre-training procedure.

To build  $e_k^B$ , the embedding of block  $B_k$  in  $k$ -th position, we first compute the weighted byte embeddings  $(P_{k,i} \times e_i^b)_{i \in [1,L]} \in \mathbb{R}^H$ , with  $H$  the hidden size of the byte embeddings. To make the block embeddings aware of the ordering of the bytes (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. We discuss our efficient implementation of these operations in Appendix 2.9.1.

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 2.9.6.

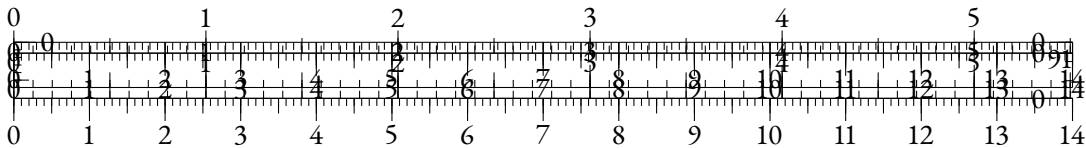
## MODEL TRAINING

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).

## 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 tokenizer-free models trained using it such as Charformer. Since this dataset is not available publicly,



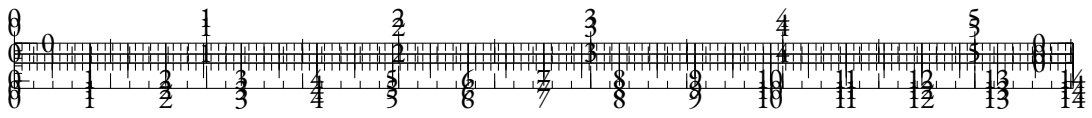


Table 2.19: Results on dev sets for the GLUE benchmark. † indicates results obtained by Tay et al. (2021), which are very similar to our models in terms of compute, but use a smaller batch size which may enhance their performance. § indicates results obtained by Ma et al. (2020). The top section concerns model trained using a subword tokenizer.

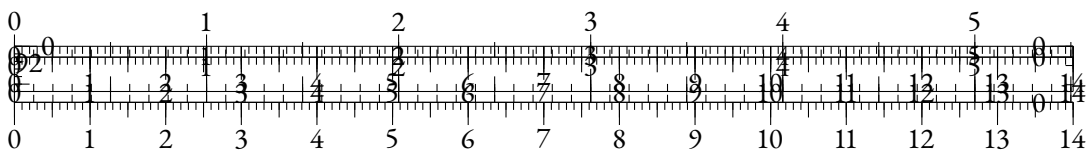
we use the English split of the mC4 distributed by AllenAI. We filter long documents containing more than  $2^{15}$  bytes, which is a simple proxy to remove important quantities of unwanted code data.

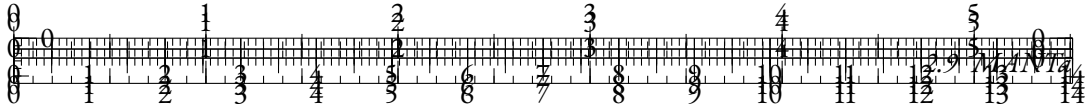
**HYPERPARAMETERS** We pre-train two versions of our model: MANTa-LM<sub>Small</sub> and MANTa-LM<sub>Base</sub>. Each of them stacks a MANTa<sub>Small</sub> (resp. MANTa<sub>Base</sub>) tokenizer and embedding module and a T5<sub>Small</sub> (resp. T5<sub>Base</sub>) encoder-decoder model stripped of its tokenizer and subword embedding matrix. Details about MANTa hyperparameters can be found in Appendix 2.9.2.

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 2.20: 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.

## 2.9.4 EXPERIMENTS AND RESULTS

### 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.

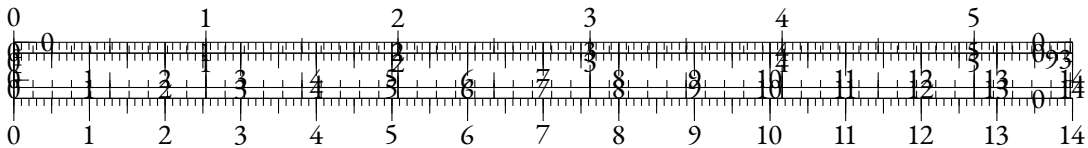
We compare our model to an encoder-decoder model with subword tokenization (pre-trained with the same denoising objective as T5) and a fully byte-level encoder-decoder, similar to ByT5. We compare *Small* models with our pre-trained versions, and *Base* models with results mentioned in Tay et al. (2021). 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.

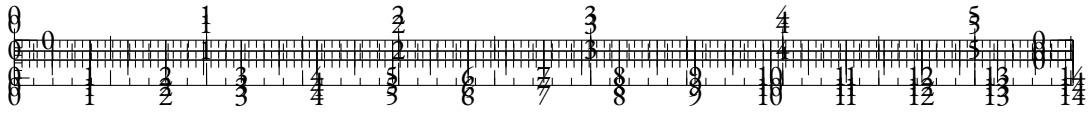
**RESULTS** Results can be found on Tables 2.18 and 2.19. Overall, MANTa-LM exhibits a performance slightly below Charformer but stays within a small margin on average (1.1 points below). Nonetheless, the main objective of our method is to balance decent performance with robustness and speed which we show in the following sections.

### 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





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 2.20. 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.

## 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.

## NATURALLY OCCURRING NOISE

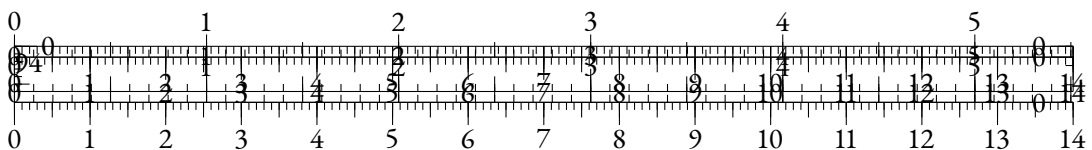
**SETUP** Similarly to Tay et al. (2021), we test our model on a toxicity detection task constructed with user generated data. We use the `TOXICCOMMENTS` dataset (Wulczyn et al., 2017) which contains 223,549 sentences annotated with a binary label indicating whether each sentence can be classified as 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 2.21 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.

## 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** 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.



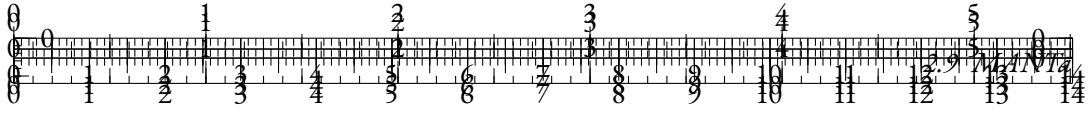


Table 2.21: Results on the TOXICCOMMENTS dataset. Results indicated by † are from Tay et al. (2021).

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 2.22: 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

### 2.9.5 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).

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.

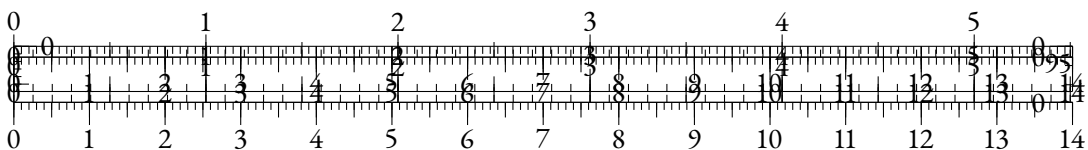
### 2.9.6 DISCUSSION

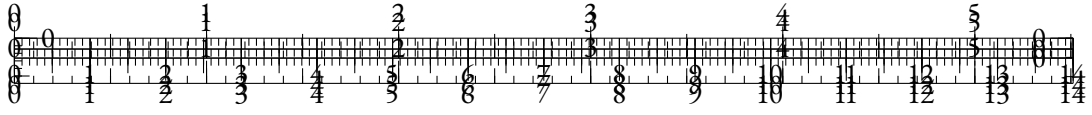
## TRUNCATING EMBEDDING SEQUENCES

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 2.9.3. 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





<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 d al ising? 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 op h ed  at  0 .01 mcg/kg/min .

Table 2.23: Examples of segmentations produced by our module (pre-trained only) and by T5’s BPE tokenizer. The sentences are samples from TOXICCOMMENTS and MEDNLI.

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.

#### LEARNT BLOCK SEGMENTATION

Segmentation examples can be found in Table 2.23. 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  $\min(p_{F_i}, 1 - p_{F_i})$  which always converges towards values of magnitude  $10^{-5}$ .

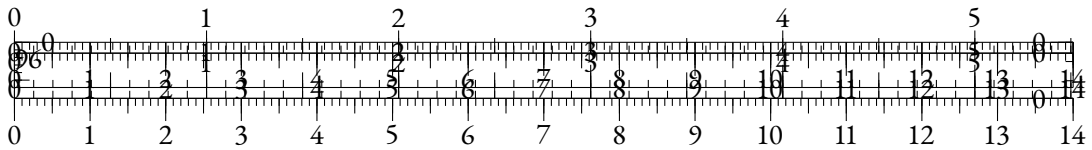
#### GRADIENT-BASED SEGMENTATION

We employ a radically different downsampling approach compared to other gradient-based tokenization methods such as CANINE (Clark et al., 2022b) 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.

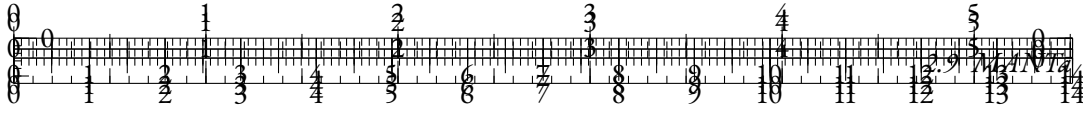
We also argue that our method yields more explainable pooled representations as the segmentation can be explicitly derived from the outputs of MANTa. Indeed, contrary to CANINE and Charformer, MANTa disentangles the segmentation of blocks from their representations, allowing to study each part separately.

#### 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.

### 2.9.7 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.

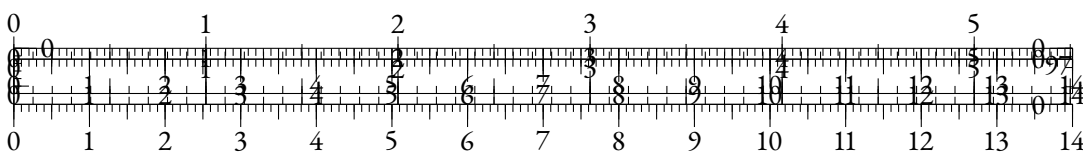
Beyond the noisy and out-of-domain settings, we believe that our approach could lead to interesting results for a number of languages, especially those whose writing system do not use whitespace separators, such as Chinese.

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.

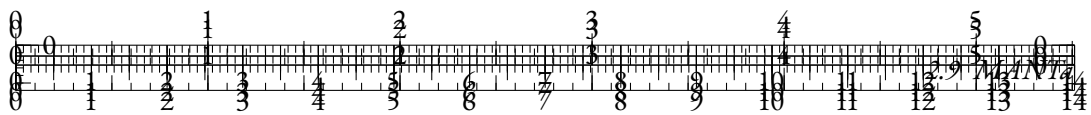
## LIMITATIONS

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. (2022b); 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.







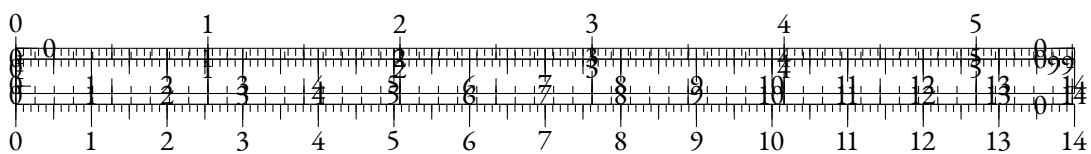
## 2.9.2 HYPERPARAMETERS

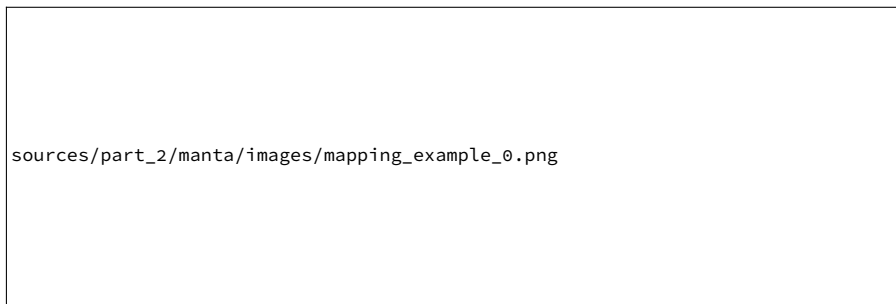
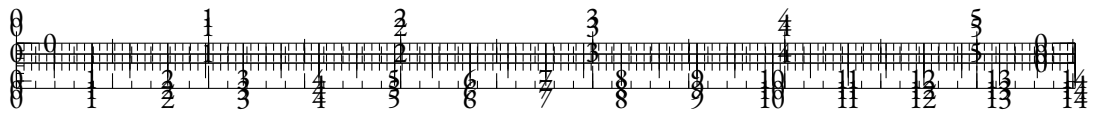
Hyperparameter	MANTa <sub>Small</sub>	MANTa <sub>Base</sub>
Input Embeddings size	64	128
Num. layers	1	2
Num. heads	8	8
Attention window	16	16
Convolution kernel size	3	3

Table 2.24: Hyperparameters for MANTa

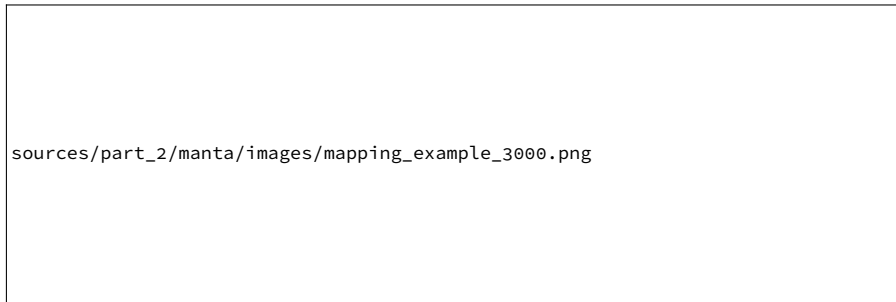
Hyperparameter	ByT5 <sub>Small</sub>	T5 <sub>Small</sub>	T5 <sub>Base</sub>
Hidden size	1472	512	768
Num. layers (encoder)	12	6	12
Num. layers (decoder)	4	6	12
Num. heads	6	8	12
Feed-forward dim.	3584	2048	3072
Dropout rate	0.1	0.1	0.1

Table 2.25: Hyperparameters for encoder-decoders

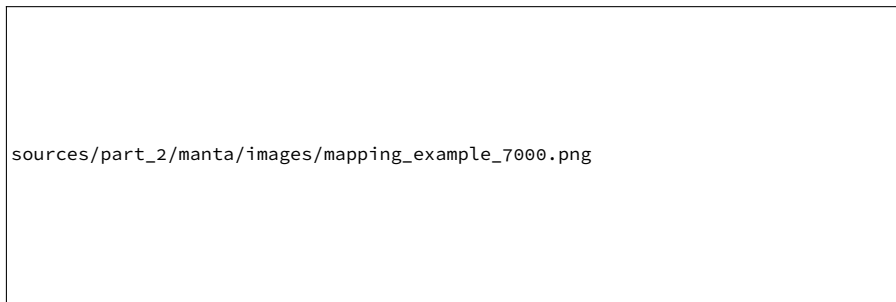




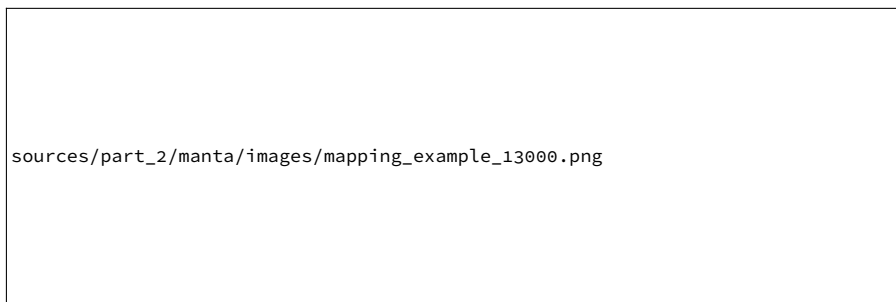
Step 0



Step 3,000

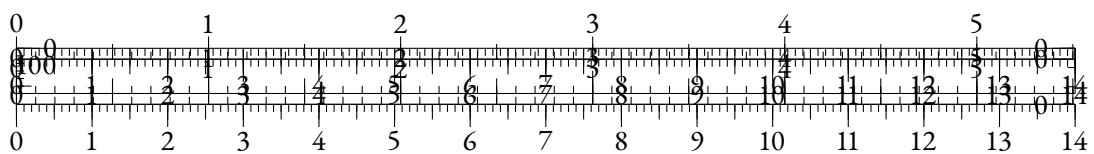


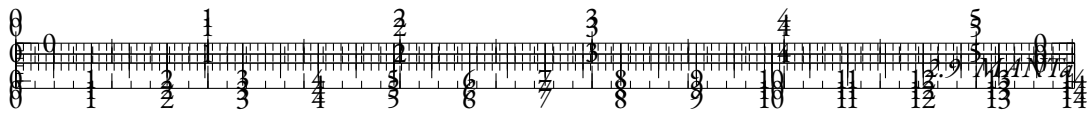
Step 7,000



Step 13,000

Figure 2.50: 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.



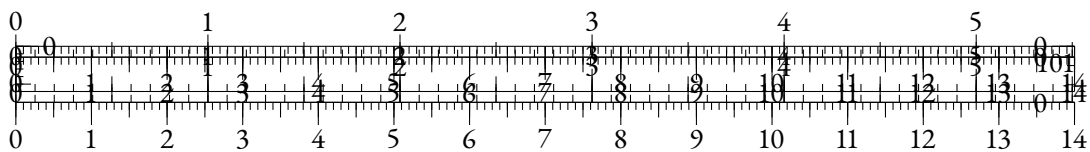


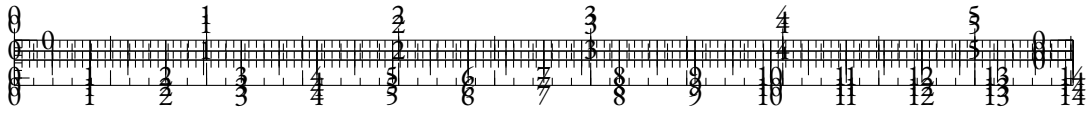
(a) DEV-ONLY



(b) TRAIN-DEV

Figure 2.51: 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.





### 2.9.3 ADDITIONAL RESULTS

We include here the scores obtained by MANTa-LM on the GLUE test sets for reproducibility and future comparisons. The development sets are used in the main body to allow a fair comparison, as the test scores are not reported in Charformer (Tay et al., 2021) and CharBERT (Ma et al., 2020).

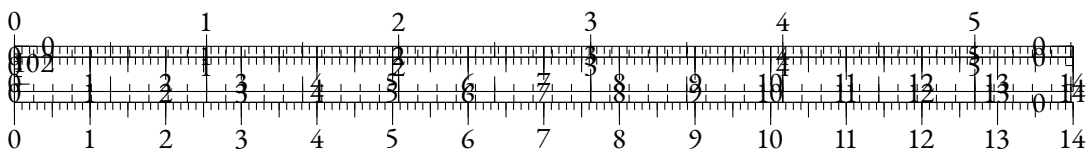
Model	$ \theta $	MNLI	QNLI	MRPC	SST-2	QQP	STSB	COLA	AVG
MANTa-LM <sub>Base</sub> (ours)	200M	78.1/78.2	88.6	83.6/88.6	91.0	70.7/88.6	74.1	45.0	78.7

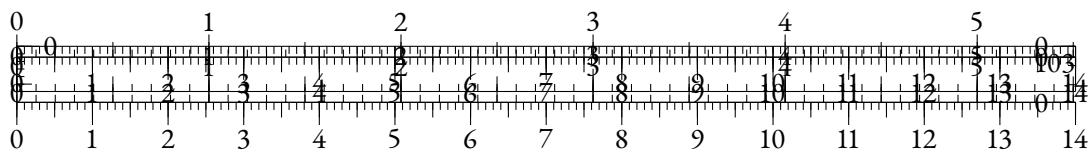
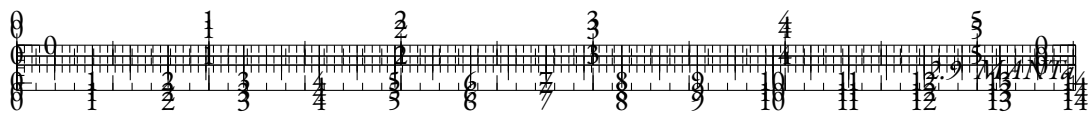
Table 2.26: Results on test sets for the GLUE benchmark.

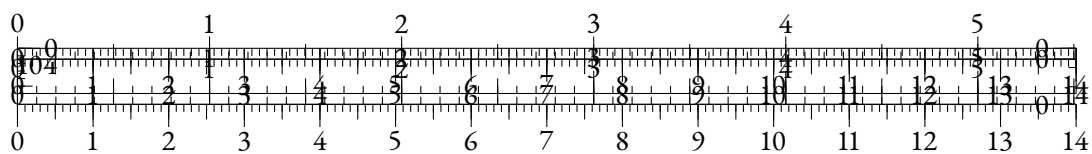
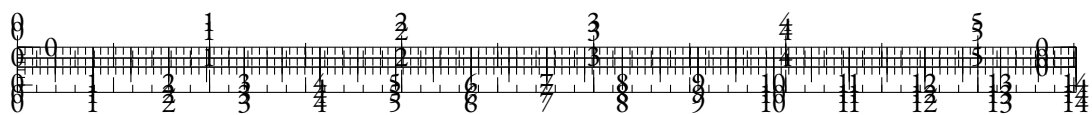
### 2.9.4 MANTa MODULE



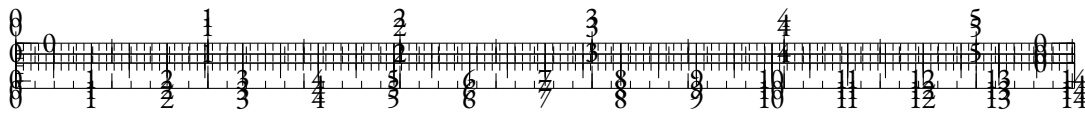
Figure 2.52: A detailed view of the MANTa module described in section 2.9.3. We denote by  $h_b$  the dimension of the byte embeddings, by  $h_{LM}$  the dimension of the block embeddings that will be fed to the encoder-decoder model,  $L$  the length of the input sequence and  $L_B$  the length of the block sequence. We omit batch sizes for simplicity.











## BIBLIOGRAPHY

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.

Ebtesam 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.

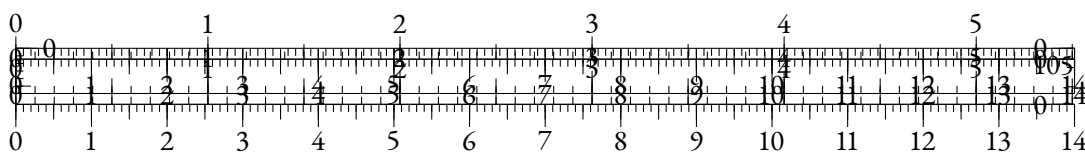
Jordan Ash, Surbhi Goel, Akshay Krishnamurthy, and Dipendra Misra. 2022. Investigating the role of negatives in contrastive representation learning. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 7187–7209. PMLR.

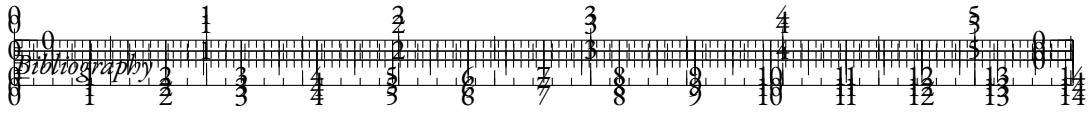
Pranjal Awasthi, Nishanth Dikkala, and Pritish Kamath. 2022. Do more negative samples necessarily hurt in contrastive learning? In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 1101–1116. PMLR.

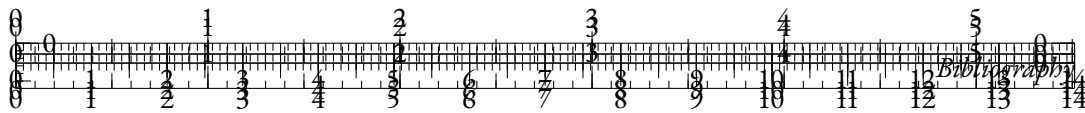
Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020a. wav2vec 2.0: A framework for self-supervised learning of speech representations.

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.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2022. BEit: BERT pre-training of image transformers. In *International Conference on Learning Representations*.







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. 2020. Language models are few-shot learners.

Francesco Camastra and Antonino Staiano. 2016. Intrinsic dimension estimation: Advances and open problems. *Information Sciences*, 328:26–41.

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.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*.

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.

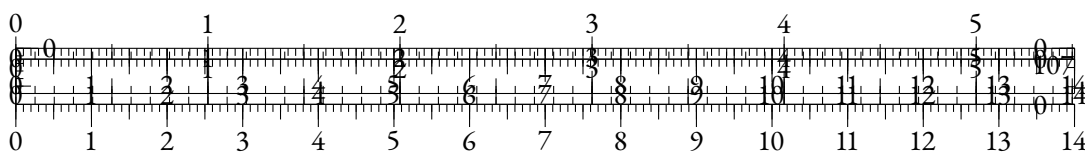
Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022a. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

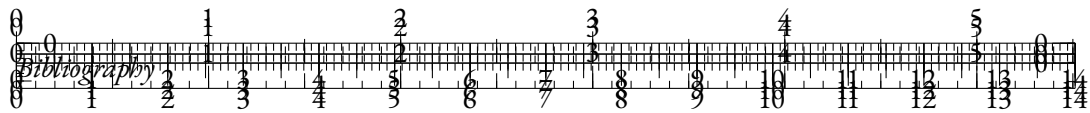
Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022b. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

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 Le, and Christopher D. Manning. 2020a. Pre-training transformers as energy-based cloze models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 285–294, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020b. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.





Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018a. What you can cram into a single  $\&\!#\ast$  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.

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.

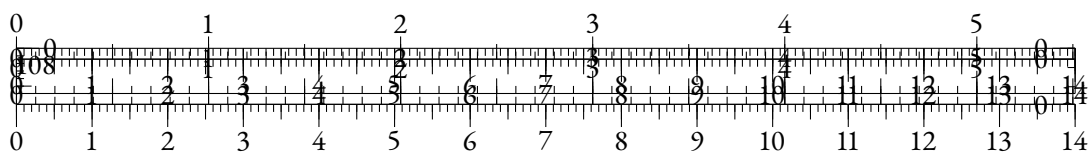
Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

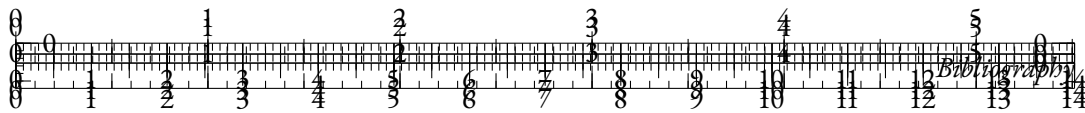
Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun’ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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.

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.

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. Croissantllm: A truly bilingual french-english language model.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.

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*.

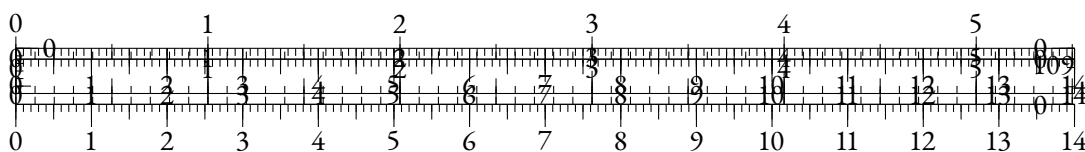
Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish 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.

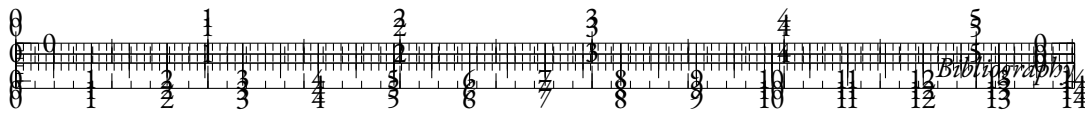
Xavier Garcia, Noah Constant, Ankur Parikh, and Orhan Firat. 2021. Towards continual learning for multilingual machine translation via vocabulary substitution. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1184–1192, Online. 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.

Nathan Godey, Roman Castagné, Éric de la Clergerie, and Benoît Sagot. 2022. MANTa: Efficient gradient-based tokenization for end-to-end robust language modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2859–2870, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.







*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.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhota, 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.

Dieuwke Hupkes and Willem Zuidema. 2018. Visualisation and ‘diagnostic classifiers’ reveal how recurrent and recursive neural networks process hierarchical structure (extended abstract). In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5617–5621. International Joint Conferences on Artificial Intelligence Organization.

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.

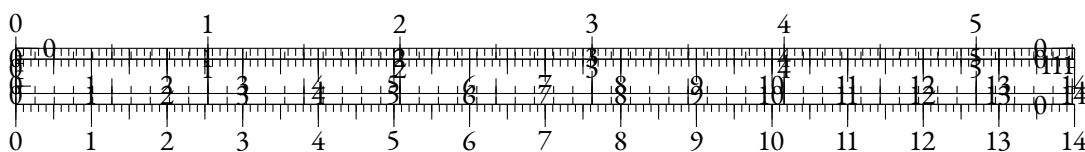
Shadi Iskander, Kira Radinsky, and Yonatan Belinkov. 2023. Shielded representations: Protecting sensitive attributes through iterative gradient-based projection. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5961–5977, Toronto, Canada. Association for Computational Linguistics.

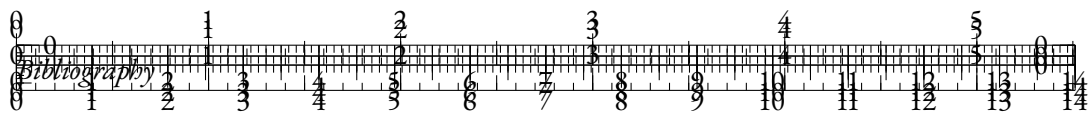
Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. 2021. Perceiver IO: A general architecture for 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.

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, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. Mistral 7b.

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 J  zefowicz, Oriol Vinyals, Mike Schuster, Noam M. Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *ArXiv*, abs/1602.02410.

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.

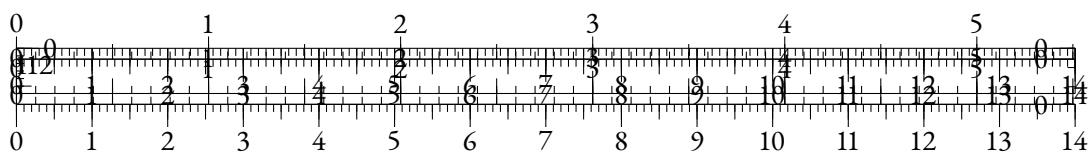
Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *ArXiv*, abs/2001.08361.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI conference on artificial intelligence*.

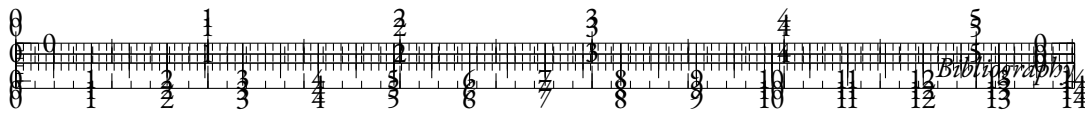
Tassilo Klein and Moin Nabi. 2023. miCSE: Mutual information contrastive learning for low-shot sentence embeddings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6177, Toronto, Canada. Association for Computational Linguistics.

Arne K  hn. 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.

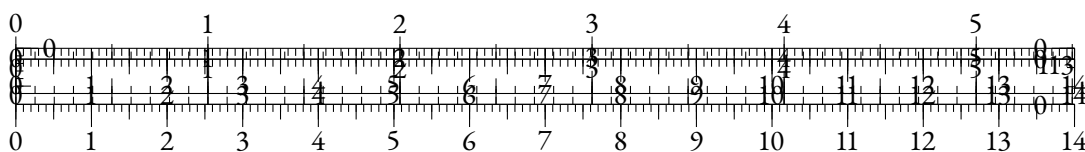
Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

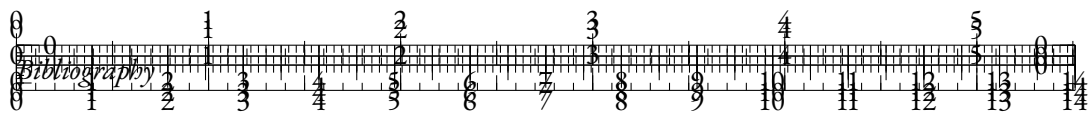






- N. Kishore Kumar and J. Schneider. 2017. Literature survey on low rank approximation of matrices. *Linear and Multilinear Algebra*, 65(11):2212–2244.
- Sachin Kumar and Yulia Tsvetkov. 2019. Von mises-fisher loss for training sequence to sequence models with continuous outputs. In *Proc. of ICLR*.
- Wen Lai, Alexandra Chronopoulou, and Alexander Fraser. 2023. Mitigating data imbalance and representation degeneration in multilingual machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14279–14294, Singapore. Association for Computational Linguistics.
- Benjamin Lefaudeux, 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>.
- 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.
- Ying-Chen Lin. 2021. Breaking the softmax bottleneck for sequential recommender systems with dropout and decoupling.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. 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.
- 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.

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.

Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, et al. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models.

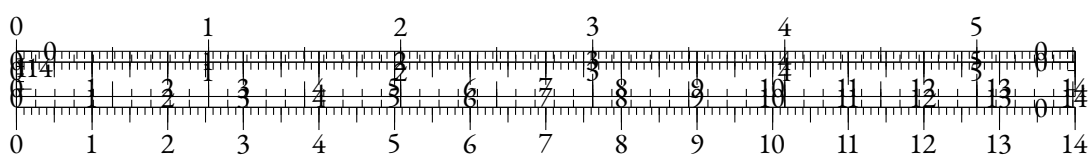
Md Mofijul Islam, Gustavo Aguilar, Pragaash Ponnusamy, Clint Solomon Mathialagan, Chengyuan Ma, and Chenlei Guo. 2022. A vocabulary-free multilingual neural tokenizer for end-to-end task learning. *arXiv e-prints*, pages arXiv–2204.

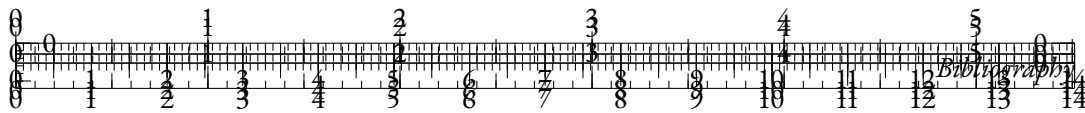
Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.

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.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernandez. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. 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.

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.

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.

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, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

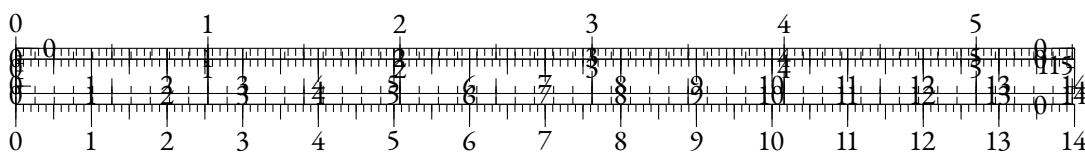
Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020a. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

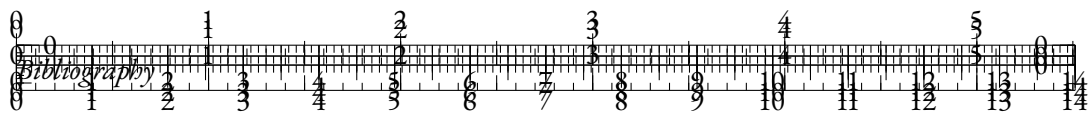
Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020b. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Sara Rajae 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 Rajae 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.

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.





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. 2024. 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.

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.

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*.

Nikhil Sardana and Jonathan Frankle. 2023. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws.

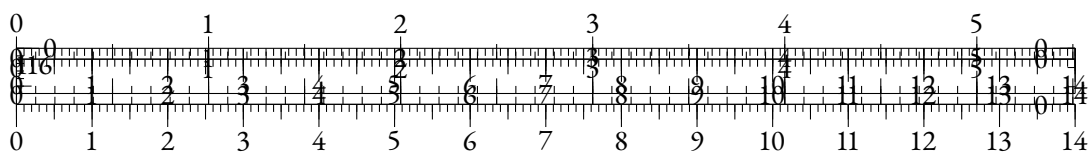
Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. 2019. wav2vec: Unsupervised Pre-Training for Speech Recognition. In *Proc. Interspeech 2019*, pages 3465–3469.

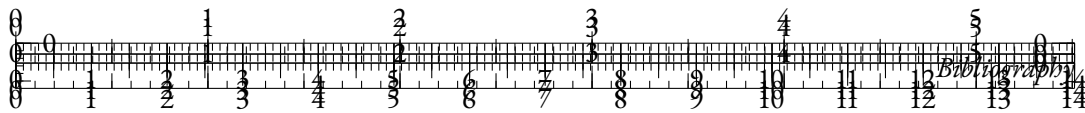
Thibault Sellam, Steve Yadlowsky, Ian Tenney, Jason Wei, Naomi Saphra, Alexander D’Amour, Tal Linzen, Jasmijn Bastings, Iulia Raluca Turc, Jacob Eisenstein, Dipanjan Das, and Ellie Pavlick. 2022. The multiBERTs: BERT reproductions for robustness analysis. In *International Conference on Learning Representations*.

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.

Utkarsh Sharma and Jared Kaplan. 2022. Scaling laws from the data manifold dimension. *Journal of Machine Learning Research*, 23(9):1–34.





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. 2024. mgpt: Few-shot learners go multilingual. *Transactions of the Association for Computational Linguistics*, 12:58–79.

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.

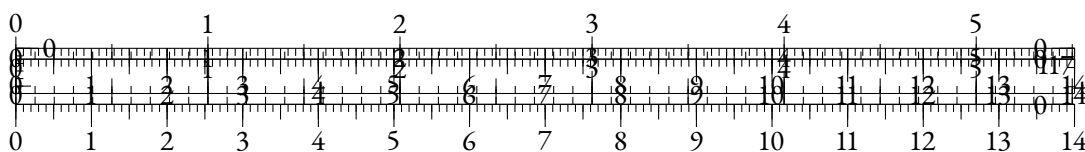
Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.

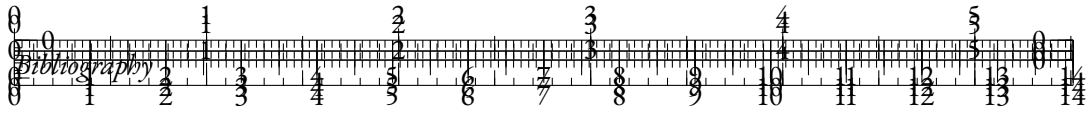
Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.

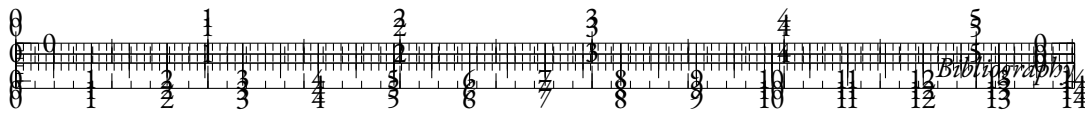
Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. 2022. Scale efficiently: Insights from pretraining and finetuning transformers. In *International Conference on Learning Representations*.

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 Hussenot, 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, 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 Mikula, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem,







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.

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*.

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.

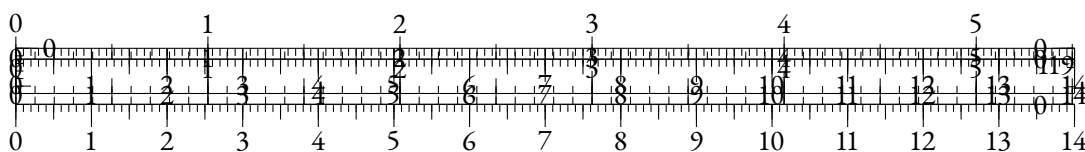
Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022b. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

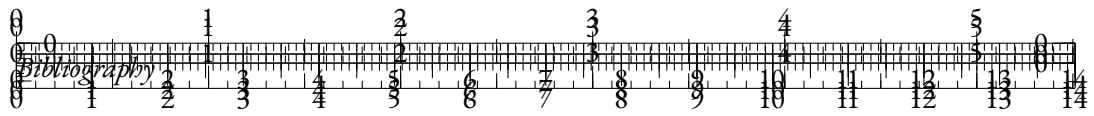
Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference 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*.

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.

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, Shuohui 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.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. 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.

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. *CoRR*, abs/2104.08465.

Kaitlyn Zhou, Kawin Ethayarajh, and Dan Jurafsky. 2021c. Frequency-based distortions in contextualized word embeddings.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. 2023. Tokenization and the noiseless channel. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5184–5207, Toronto, Canada. Association for Computational Linguistics.

