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 Villemonte 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	Examinateur
Christopher Columbus	Kingdom of Castile	Invited member
Margaret Hamilton	University of Michigan	Rapporteur
Emmy Noether	Georg-August-Universität Göttingen	Examinateur
Jürgen Schmidhuber	IDSIA	Directeur
Jean LeCun	Facebook AI	Co-directeur
Claude Shannon	MIT	Examinateur
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.

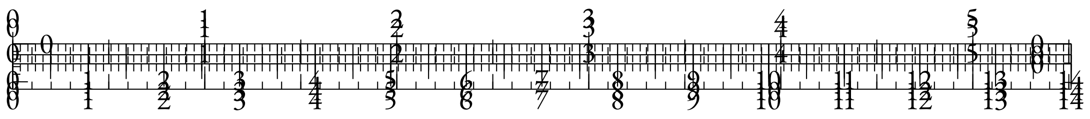
In this thesis, we explore high-level properties of the features (or *representations*) that are extracted by these language models, and we leverage these properties to improve language systems and to quantify their limitations. This work is two-fold, as we first focus on the learnings that result from 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.

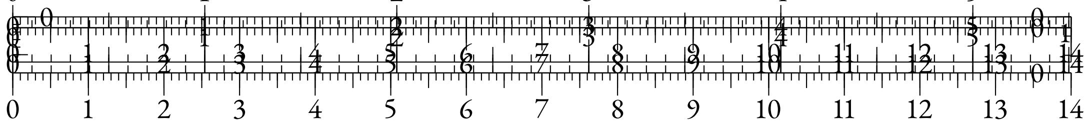
Contents



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 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 ay 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

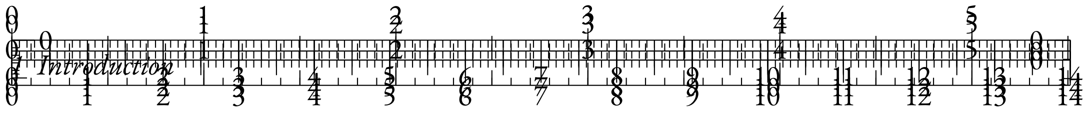
```
\documentclass{mimosis}
```

at the beginning of your document. This is sufficient to use the class. It is possible to build your document using either LaTeX|, XeLaTeX, or LuaLaTeX. 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 XeLaTeXor LuaLaTeXas 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:



Package	Purpose
`amsmath`	Basic mathematical typography
`amsthm`	Basic mathematical environments for proofs etc.
`booktabs`	Typographically light rules for tables
`bookmarks`	Bookmarks in the resulting PDF
`dsfont`	Double-stroke font for mathematical concepts
`graphicx`	Graphics
`hyperref`	Hyperlinks
`multirow`	Permits table content to span multiple rows or columns
`paralist`	Paragraph ('in-line') lists and compact enumerations
`scrlayer-scrpage`	Page headings
`setspace`	Line spacing
`siunitx`	Proper typesetting of units
`subcaption`	Proper sub-captions for figures

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

```
\setmainfont{Baskerville}
\setsansfont[Scale=MatchLowercase]{IBM Plex Sans}
\setmonofont[Scale=MatchLowercase]{IBM Plex Mono}
```

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 finds a combination that you enjoy.

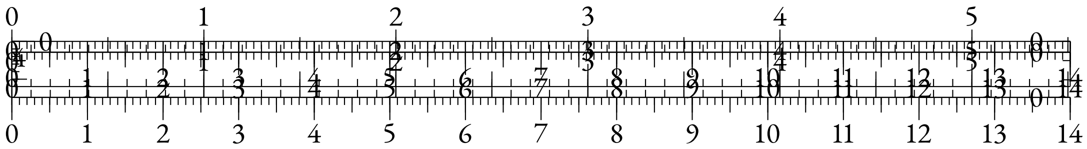
The template automatically imports numerous convenience packages that aid in your typesetting process. Table 1.1 lists the most important ones. Let's briefly discuss some examples below. Please refer to the source code for more demonstrations.

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.

Likewise, as a small pet peeve of mine, I offer specific operators for *ordinals*. Use `\th` to typeset things like July 4$^{\text{th}}$ correctly. Or, if you are referring to the 2$^{\text{nd}}$ edition of a book, please use `\nd`. Likewise, if you came in 3$^{\text{rd}}$ in a marathon, use `\rd`. This is my 1$^{\text{st}}$ rule.

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.

2 Related Works

2.1 Language Modeling

2.1.1 Introduction

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

$$P(w_T|w_{\neq T})$$

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

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

2.1.2 Methods

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

First, many works have built linguistically-grounded sets of textual units that should be considered as microscopic to some extent, such as the Morphosyntactic Annotation Framework (Clément and Villemonte de La Clergerie, 2005), which defines a token as a *non-empty contiguous sequence of graphemes or phonemes in a document*. More generally, the domain of morphology, well defined in Aronoff and Fudeman (2022), has led to several morphologically-informed tokenizers (Saleva and Lignos, 2021; Grönroos et al., 2018).

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

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

sources/related_works/imgs/token_graph.pdf

Figure 2.1

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

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

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

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

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

The statistical approaches are widely used in modern LMs, sometimes jointly with techniques such as subword regularization (Provilkov et al., 2020) that diversify segmentation results.

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

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

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

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

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

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

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

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

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

2.1.3 Architectures

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

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

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

We can extend this idea by estimating the *2-gram* distribution $P(w_{t-1}w_t)$. To do so, we count the occurences of the subsequence $w_{-1}w_0$ in the training dataset for each pair of tokens $w_{-1}, w_0 \in \mathcal{V}^2$. Doing so, we can build a $V \times V$ stochastic matrix - i.e. which coefficients are non-negative reals that sum to 1 column-wise - containing the observed frequencies for w in pairs of the form $w_{-1}w$:

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

Then, the language model ϕ_θ becomes:

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

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

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

The associated language model ϕ_θ can then be written:

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

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

Neural Methods Bengio et al. (2000) introduce the idea of training ϕ_θ as a neural network with parameters θ. The model is composed of three separate layers:

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

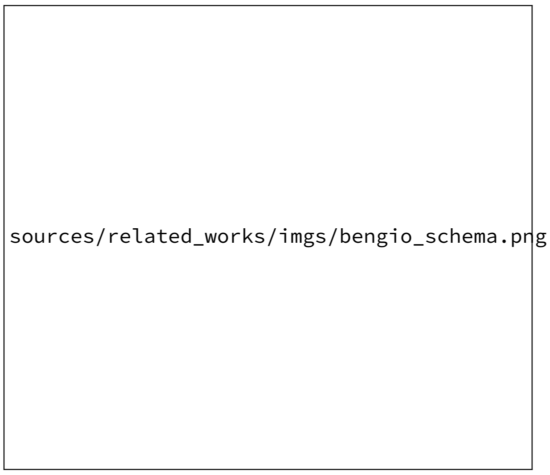


Figure 2.2: Schema of a basic neural network architecture for language modeling (from Bengio et al. (2000)).

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

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

The model is then trained to optimize the log-likelihood using stochastic gradient descent (see **??**) on the parameters θ of the neural network.

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

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

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

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

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

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

$$(o_t, (h^1_t, ...h^k_t)) = v(x_t, (h^1_{t-1}, ...h^k_{t-1}))$$

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

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

Transformers Bahdanau et al. (2015) popularized the use of *attention* in neural machine translation models as a method that lets the model select relevant tokens from the source sequence at a given prediction step. Such models, that previously used the last hidden states of a source-processing RNN (called *encoder*) as the input to a target-generating RNN (called *decoder*), suffered from an information bottleneck due to sharing only last-step representations between source and target sequences. The attention mechanism allowed to ease this bottleneck, by providing a *direct path of interaction* between the source tokens and the decoder.

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

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

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

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

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

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

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

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

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

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

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

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

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

[1]For the sake of simplicity, we consider unbiased linear layers for the rest of this section.

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

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

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

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

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

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

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

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

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

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

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

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

where $\bar{e_t} = \frac{1}{L}\sum_{j=1}^{L} e_{t,j}$ and γ and β are network parameters.

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

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

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

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

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

In Vaswani et al. (2017), the main task is Neural Machine Translation, which leads to the use of an *encoder-decoder* architecture. The source sequence is processed through an encoder Transformers, and the target sequence logits are generated using causal self-attention blocks with an added cross-attention layer. Cross-attention is similar to self-attention, but uses Q^h and K^h representations from one sequence and V^h from another sequence. In the terms of Equation (2.1), cross-attention between sequences x and y can be written:

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

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

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

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

$$(Q^h {K^h}^T)_{ij} = \langle \omega_i(Q^h_i), \omega_j(Q^h_j) \rangle + \beta_{ij} \tag{2.3}$$

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

$$\begin{cases} \omega_i(x) = \mathbf{R_i^{d_m}} x \\ \beta_{ij} = 0 \end{cases}$$

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

$$\mathbf{R_i^{d_m}} = \begin{pmatrix} R_{i,\theta_1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & R_{i,\theta_{d_m/2}} \end{pmatrix}$$

where $R_{i,\theta} = \begin{pmatrix} \cos i\theta & -\sin i\theta \\ \sin i\theta & \cos i\theta \end{pmatrix}$ and $\theta_d = 10000^{\frac{-2(d-1)}{d_m}}$.

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

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

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

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

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

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

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

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

GPT (for *Generative Pre-trained Transformer*) is a Transformers-based architecture that trains a decoder model for causal language modeling in a straightforward way. The training set is BookCorpus (Zhu et al., 2015), which contains unpublished books from a variety of genres. The authors use a 12-layer decoder-only Transformer with $d_m = 768$, $n_h = 12$ attention heads and a vocabulary of $V = 40000$ tokens. Overall, GPT counts 110 million parameters.

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

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

- Replace it with a specific mask token 80% of the time;
- Replace it with a random token 10% of the time;
- Leave it unchanged 10% of the time.[2]

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

T5 (for *Text-to-Text Transfer Transformer*) is a model suite that aims for a different approach when it comes to downstream tasks. Raffel et al. (2020a) argue that downstream tasks can be rephrased as natural language samples. For instance, the Corpus of Linguistic Acceptability, or CoLA (Warstadt et al., 2018) is a sentence-level classification task where a grammatical acceptability

[2]In that case, the training task is not language modeling but simply learning the identity mapping.

label (acceptable or unacceptable) is given to each sentence. The STSB subset of GLUE (Wang et al., 2018) is a sentence-pair classification task where a similarity score in $[1, 5]$ is given to a pair of sentences. The authors argue that these tasks can be rephrased as language modeling tasks where the labels or scores are tokens that the model is expected to generate at inference time. The main advantage of this approach is that it does not require to have task-specific classification heads, which implies that the model can be fine-tuned on all tasks at once.

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

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

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

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

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

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

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

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

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

GPT-3 (Brown et al., 2020a) introduces a much larger architecture, using 175 billion parameters, while relying on a training procedure and architecture that do not significantly differ from the original Transformers-based GPT. The authors claim that this model displays a zero-shot downstream performance level that is close to fine-tuned counterparts. Similar results are reproduced through the OPT initiative (Zhang et al., 2022). Subsequently, the 540 billion parameters PaLM model (Chowdhery et al., 2024) was released, leading to similar conclusions. A multilingual counterpart to this line of work is BLOOM (Le Scao et al., 2023), a 176 billion parameters model trained in 46 languages.

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

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

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

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

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

However, Hoffmann et al. (2022) suggest that the empirical results from Kaplan et al. (2020) were not accurate. They first notice that Kaplan et al. (2020) used intermediate checkpoints taken before complete cooldown, implying underestimated low-data performance. They also underline that there is a slight curvature of the scaling law by exploring larger model sizes to interpolate their scaling law. Their scaling law can be written:

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

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

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

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

2.1.4 Limitations & Extensions

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

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

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

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

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

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

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

They proceed to argue that $\text{rank}\, A$ should be very high for natural language. In practice, measuring $\text{rank}\, A$ would imply getting access to A which is impossible. However, it can be argue that the distribution of tokens is highly context-dependent, as one single token (e.g. a negation marker) can imply a complete shift in token distribution at later positions. It is also unlikely that $\text{rank}\, A$ is low as it does not seem plausible that only a few hundred of bases can express the whole diversity of contexts in language. These arguments hint towards higher rank values for A.

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

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

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

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

Tokenization Some of the induced biases of tokenizers can be harmful for modelization. One such limitation lies in their brittleness to character deformations which are commonly found in real world, noisy data. For instance, BERT's tokenizer (Devlin et al., 2019) encodes "performance" as `["performance"]` but "perfonmance" as `['per', '##fo', '##n', '##man', '##ce']`, which makes it hard for the model to behave similarly in both cases. Moreover, the tokenizer is fixed after its training and is therefore impossible to update without retraining, for instance to reflect new domains (El Boukkouri et al., 2020) where tokenization might over-segment specific or technical terms. Clark et al. (2022a) list other issues emerging when using static subword tokenizers, especially when modeling languages with a more complex morphology than English.

Tokenizers are also a limitation when it comes to multilingual models. Rust et al. (2021) show that training models using monolingual tokenizers systematically leads to better performance compared with multilingual ones. Petrov et al. (2023) show that a single sentence can be 15 times shorter than its translation in another language. Moreover, the use of multilingual tokenizers often leads to the use of larger vocabularies which results in more weights being assigned to input embeddings. Nevertheless, Liang et al. (2023) show that larger vocabularies lead to better downstream performance, thus indicating that multilingual models should ideally be able to handle such vocabularies at reduced cost.

Character-level Models Several alternative methods have been proposed to mitigate these tokenization issues. This line-of-work suggests to learn character-level or byte-level representation for LMs instead of subword-level ones. These methods improve the robustness of LMs to naturally occurring noise as well as their expressiveness when dealing with out-of-domain or multilingual data. In order to cope with increased input lengths, some of these methods compress sequences with constant reduction rates obtained using specialized modules (Clark et al., 2022a; Tay et al., 2021), subsequently removing any notion of subwords.

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

Nevertheless, they still rely on fixed tokenization heuristics (for instance segmenting using whitespaces) which may not be suited to some languages or certain types of language variations. Several works have tried to remove these induced biases by working purely with characters or bytes as input. CANINE (Clark et al., 2022b) downsamples contextualized character representations via a strided convolution before feeding them to a Transformers. It can be trained either with a subword-based objective (CANINE-s) or with a character-level one (CANINE-c). Tay et al. (2021) design a similar model by replacing convolutions with efficient Transformers (Beltagy et al., 2020). A more direct approach, ByT5 (Xue et al., 2022a) is a version of T5 that is trained at byte-level. Finally, YU et al. (2023) introduce the MEGABYTE model, by using a basic strided pooling approach to apply a Transformer architecture on 4-bytes representations before using a more efficient model to decode these representations back to byte-level.

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

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

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

Notably, Beltagy et al. (2020) propose several alternatives in their Longformer architecture. First, they suggest only computing coefficients (i, j) where $|i - j| < \delta$, thus resulting in a *sliding window* pattern which gives its name to the method later used in Mistral models (Jiang et al., 2023). They proceed to present two variations of the sliding window attention: the first relies on a dilated pattern, and the second one allows a portion of the positions to use global attention, that is for a portion of i values, the attention map value is computed at all positions (i, j) for $j \in [1, L]$. These alternatives naturally come with their causal counterpart, for which the non-causal interactions are not included in the targeted (i, j) positions. The Longformer attention maps can be computed in $O(\delta L)$, which greatly accelerates inference in training when $\delta \ll L$.

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

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

During generation, the KV cache grows linearly in size and it represents a total of :

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

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

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

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

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

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

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

Biases & Ethical Concerns Bender et al. (2021) notoriously present issues related with the increasing scale of language models. They argue that, as language models grow larger and more performant, their financial and ecological costs should be considered as a potential concern in the long run, as has been explored in other works (Ligozat et al., 2022; Rillig et al., 2023). They add that these commercial large language models (or LLMs) being trained on large web-scraped datasets that are poorly curated, they may contain dangerous and socio-culturally biased information that is then reproduced by the model at inference. These biases may include sterotypical views about gender (Kotek et al., 2023), religious groups (Abid et al., 2021) or race (Nadeem et al., 2021), among others.

2.2 Representation Learning for Natural Language

2.2.1 Introduction

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

- Bengio et al. (2013)

It is often stated informally that

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

For tabular data, choosing representations to feed a statistical or neural model is straightforward, as they are readily available in a numerical format and can be vectorized directly. However, for modalities such as natural language, the representation step is crucial and raises various questions. Text can be seen as a sequence of symbols that follow some hierarchical patterns (Longacre, 1970), which makes harder to translate to vector representations, especially as these underlying patterns are complex and brittle to subtle changes. Even when the notion of atomic units is defined (see Section 2.1.2), some properties of language demand peculiar attention before tensor representations can be extracted and used in machine learning pipelines.

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

Secondly, textual data is sequential, which complexifies the possible nature of tasks and the granularity of the represented objects. For instance, some models may be designed to classify words, and others to cope with document-level tasks. This implies that textual information of different nature and/or shape (e.g. two documents with different lengths or languages) may need to be represented in similar vector spaces so that a machine learning model can perform predictions about these different pieces of information.

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

2.2.2 Statistical Methods

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

Counting Methods

A naive approach to document representation, called *bag-of-words*, consists in counting the words and using a histogram as a vector representation. Based on the token sequence framework introduced in Section 2.1, we consider a document D as a sequence of tokens $(w_t)_{t \in L_D}$, and we define its bag-of-word representation $x \in \mathbb{N}^V$:

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

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

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

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

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

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

The first term corresponds to Text-Frequency, and can take other forms (e.g. log-regularization). The second term is Inverse-Document-Frequency and computes the rate of documents that contain the word w. This technique yields more *expressive* representations as resulting vectors better capture the specificity of different documents (Ramos et al., 2003).

Topic Modeling Both with BoW and TF-IDF, the resulting representations can be automatically compressed to lower-dimensional vectors at corpora level using *Latent Semantic Analysis* (Deerwester et al., 1990). To do so, LSA relies on the Singular Value Decomposition (or SVD) of the matrix of document representations[3] $X \in \mathbb{R}^{\mathcal{D} \times V}$ to identify components that are shared across documents.

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

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

[3]In the case of BoW, this matrix is also called the *Term-Document* matrix.

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

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

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

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

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

Co-occurence Matrices The key concept for most statistical methods used to obtain word-level representations is the distributional hypothesis (Harris, 1954) that states that words are characterized by the words that appear in their context. (Weaver, 1952) introduces *statistical semantics*, a field that employs statistical methods to analyze word meanings in natural languages. A straightforward application of these theories to the document representation framework consists in considering the columns of X as useful token-level representations, as tokens that appear in the same documents should convey similar meanings.

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

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

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

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

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

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

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

2.2.3 Neural Methods

As computational capabilities improved over the years, and as they became popular in Computer Vision (Krizhevsky et al., 2012), neural approaches were increasingly studied in Natural Language Processing (Billingsley and Curran, 2012; Cho et al., 2014a).

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

GloVe *GloVe* (Global Vectors for Word Representation) was introduced in Pennington et al. (2014). Inspired by the rationale of PMI, they propose to learn a log-bilinear regression on a regularized co-occurrence matrix. Namely, they learn token representations x and $\tilde{x}$ by minimizing the following objective[4]:

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

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

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

[4] In practice, a regularization term is used to account for rare co-occurences.

string itself, and use the Skip-gram objective from Word2Vec at token-level over the summed representations.

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

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

Contextual Embeddings

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

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

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

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

2.2.4 Sentence Embeddings

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

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

Sentence-BERT

Reimers and Gurevych (2019) introduce Sentence-BERT, a model based on BERT that generates sentence-level representations. They use a siamese architecture where a single model encodes two sentences into pooled representations, and a classifier predicts a label that should detect whether two sentences share the same meaning. This siamese network is initialized with BERT or RoBERTa models and is further trained on the SNLI dataset (Hill et al., 2016), a *natural language inference* (or NLI) dataset that comprises pairs of sentences associated with a label indicating a semantic correspondance between the sentences. At inference time, the siamese network produces sentence embeddings and their *cosine similarity* is computed as a similarity score.

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

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

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

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

Latent Regularization Methods

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

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

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

Contrastive Methods

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

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

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

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

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

Similarly to Chen et al. (2020), they leverage the InfoNCE objective (van den Oord et al., 2019a) which can be interpreted as a cross-entropy loss on in-batch classification, i.e. identifying the object in a training batch to which a representation corresponds. Formally, given an object embedding $h \in \mathbb{R}^d$, a positive sample embedding $h_+ \in \mathbb{R}^d$ and a series of negative samples $h_-^i \in \mathbb{R}^d$ for $i \in [1, N_-]$, the InfoNCE objective based on cosine similarity is:

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

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

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

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

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

2.3 Representation Analysis for NLP

GENERATED WITH CHATGPT, NO FEEDBACK EXPECTED

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.

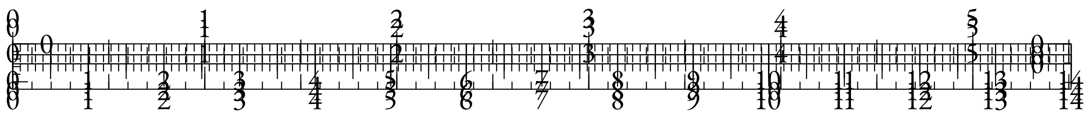
- **Dimensionality Analysis**: Analyzing and managing the dimensionality of the embedding space helps in ensuring that the model makes full use of the available dimensions. Methods like Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) can help in identifying and addressing dimensional 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.

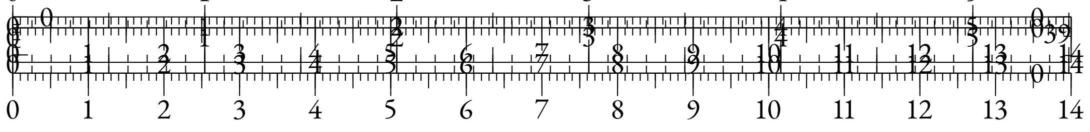
In summary, representation degeneration in the geometric sense involves anisotropy, outlier dimensions, representation collapse, and biases in latent spaces. Addressing these issues is crucial for maintaining the effectiveness of learned embeddings and ensuring that they provide accurate and fair representations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



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.



2.4 Geographical

2.4.1 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-occurence 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.

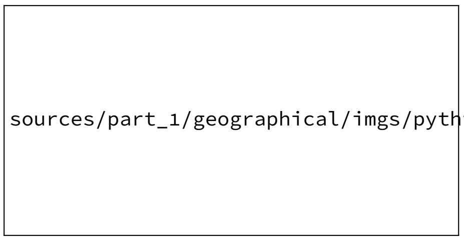
In this work, we propose to extend the analysis by Gurnee and Tegmark (2024) to smaller language models, in order to observe how scale affects the ability of models to implicitly embed geographical information from raw training data. We show that such ability consistently improves with model size, and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

We make several contributions:

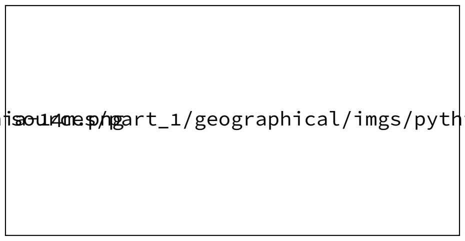
- We show that geographical information can be extracted to a certain extent from representations at every model scale;
- We observe that larger models are more geographically biased than their smaller counterparts;
- We find that the performance of models in terms of geographical probing is correlated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

2.4.2 Scaling Laws of Geographical Probing

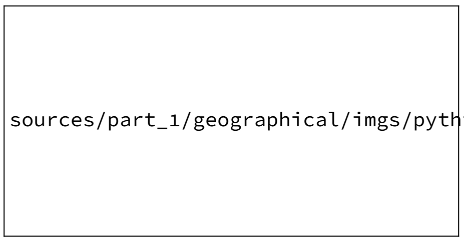
In this section, we train geographical probes for a wide variety of models at different scales.



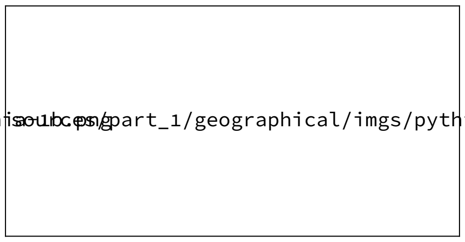
(a) Pythia 14M ($R^2 = 34.34$)



(b) Pythia 160M ($R^2 = 55.28$)



(c) Pythia 1B ($R^2 = 67.94$)



(d) Pythia 2.8B ($R^2 = 74.97$)

Figure 2.3: 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 2.3, 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 2.4 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., 2024b), and Llama-2 (Touvron et al., 2023). We also display results for several encoder models: BERT (Devlin et al., 2019; ?), RoBERTa sources/part_1/geographical/imgs/r_square_world_decoders.png

(a) Decoder models

sources/part_1/geographical/imgs/r_square_world_encoders.png

(b) Encoder models

Figure 2.4: Evolution of the R^2 coefficient on the test set for various model suites.

(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 (?) 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.

2.4.3 Geographical Bias and Scale

In Figure 2.3, 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 2.5. 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.

sources/part_1/geographical/imgs/mse_continent.png

Figure 2.5: 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 2.6 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 2.3, and shows that scale does not solve the geographical discrepancy caused by bias inherent to the training data.

In Figure 2.7, 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 sources/part_1/geographical/imgs/gini_combined.png

Figure 2.6: Gini coefficients of MSE on the test set averaged by country or by continent, as model size increases.

sources/part_1/geographical/imgs/heatmap_100_50.png

Figure 2.7: Test log-MSE for Pythia-1B as plotted on a World map.

sources/part_1/geographical/imgs/correl_size.png

Figure 2.8: 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 [5]. 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 2.8. As in Figure 2.3a, 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.

[5] `https://huggingface.co/datasets/ola13/small-the_pile`

2.4.4 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.

2.5 Softmax Bottleneck

2.5.1 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; Rajaee 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_θ 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

modeling head is a linear layer with parameter $W \in \mathbb{R}^{V \times d}$, where V is the number of possible tokens. The resulting next-token probability distribution is then given by:

$$p(y_i) = \sigma(W f_\theta(y_{<i}))$$

where σ is the softmax function.

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

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

Overall, our contributions can be summarized as:

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

2.5.2 Related Works

Small LMs & Saturation   Biderman et al. (2023b) train Pythia, a suite of models of various sizes on 300B tokens from the Pile (Gao et al., 2020), and release the weights for an exhaustive number of checkpoints during training. They notice that smaller models suffer a performance decrease on the Lambada dataset (Paperno et al., 2016) in late training. The scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022) predict that training smaller models on large corpora is suboptimal in terms of compute. However, recent initiatives (Zhang et al., 2024; Faysse et al., 2024; Team et al., 2024) have pretrained smaller language models on large datasets, motivated by inference cost reduction (Sardana and Frankle, 2023).

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; Rajaee and Pilehvar, 2021) and was proven to be related with the Zipfian shape of token distribution (Gao et al., 2019b; Biś 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.

2.5.3 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 2.9a, 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 2.9b, 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 α) 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):

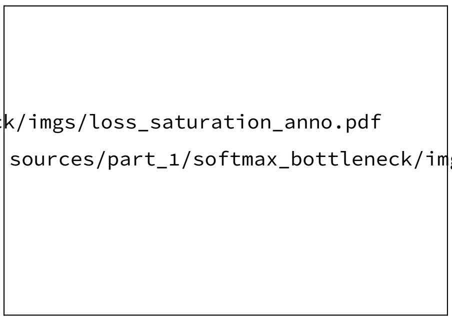
$$L(N, T) = \frac{A}{N^{\alpha}} + \frac{B}{T^{\beta}} + E$$

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.

sources/part_1/softmax_bottleneck/imgs/loss_saturation_anno.pdf

(a) Loss saturation


sources/part_1/softmax_bottleneck/imgs/scaling_laws_unfit_anno.

(b) Loss extrapolation

Figure 2.9: 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	**24.6**	**40.3**	**59.6**	**30.47**	**79.6**	**46.5**
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.

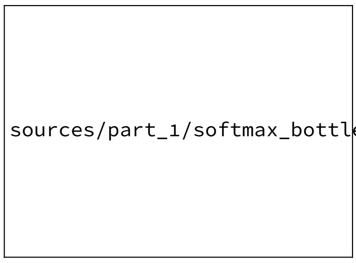
2.5.4 Performance Saturation is Rank Saturation

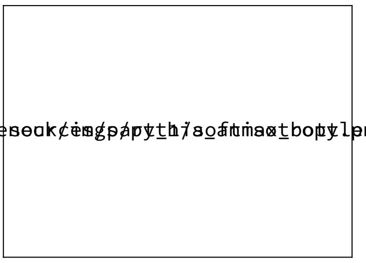
Anisotropy at Scale

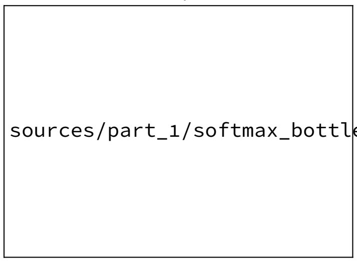
Anisotropy is a common form of representation degeneration that has been observed among various small language models. It consists in reduced angular variability of the representation distribution at a given layer. Previous works (Ethayarajh, 2019; Godey et al., 2024) notice that almost all layers of 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.


(a) Pythia


(b) GPT-2


(c) Gemma

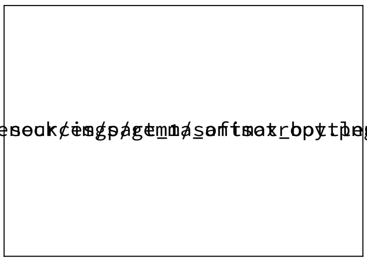

(d) OPT

Figure 2.10: Anisotropy in function of layer depth (i.e. order in the forward pass).

In Figure 2.10, we observe that most layers of Transformers models are anisotropic to some extent, regardless of the scale. Nevertheless, there seems to be a dichotomy in the last layer, where models are either nearly isotropic or highly anisotropic. Interestingly, we notice that the dichotomy aligns with the one of the saturation phenomenon for the Pythia suite, where only models containing 160M or fewer parameters seem affected by last-layer anisotropy.

We thus decide to study the training dynamics of anisotropy for the Pythia suite, and compare them with the saturation phenomenon in Figure 2.11.

Figure 2.11 illustrates a neat correlation between the emergence of the performance saturation phenomenon and the appearance of anisotropy in the last-layer representations of the models. It also shows that anisotropy increases abruptly around the saturation point during training. Moreover, we see here that on a specific in-domain corpus, the models quickly lose performance at saturation and never seem to fully recover from this explosion.

Singular Values Saturation

Average cosine-similarity is a valuable measure of the uniformity of a distribution, but including other metrics can help to better capture the complexity of some manifolds (Rudman et al., 2022). Moreover, it only focuses on the output embeddings of the language models, and not on their weights. In this section, we extend our analysis by studying the singular value distributions of the language modeling heads, to link our empirical observations to our theoretical findings. In Figure 2.12, we display the singular value distributions of the final predictive layer weights W along training.

Figure 2.12 sheds light on a specific pattern of spectral saturation, roughly co-occurring with the performance saturation phenomenon. It shows that the singular value distribution progressively

(a) 14M (b) 31M (c) 70M
(d) 160M (e) 410M

Figure 2.11: 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 2.13 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 2.13: Training dynamics of the singular entropy, for different Pythia models.

(a) 14M (b) 31M (c) 70M

(d) 160M (e) 410M

Figure 2.12: Evolution of the singular value distributions of the LM heads of Pythia models during training, normalized by the maximum singular value.

2.5.5 The Softmax Bottleneck & Language Dimensionality

Inherent Dimensionality of Natural Language

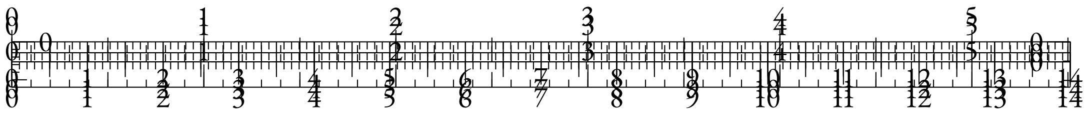
Intuitively, the saturation of the singular values distribution observed only for smaller models in Section 2.5.4 questions the dimensionalities involved in the optimization of the LM head. In this section, we propose to empirically measure a critical value for the rank of the LM head, and to estimate the dimensionality of the contextual probability distribution the head's outputs are supposed to match.

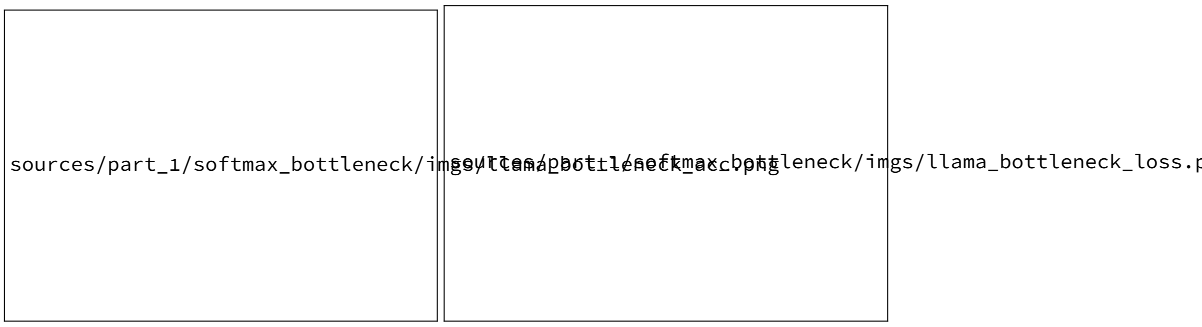
In order to empirically measure the effect of the rank of the linear head, we propose to train rank-constrained heads on pretrained contextual representations from highly-parameterized language models. In order to control the maximum rank r, we consider heads of the form $W = AB \in \mathbb{R}^{V \times d}$, 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 2.5.2).

In Figure 2.14, 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




sources/part_1/softmax_bottleneck/imgs/llama_bottleneck_acc.png
sources/part_1/softmax_bottleneck/imgs/llama_bottleneck_loss.p

(a) Accuracy (b) Cross-entropy

Figure 2.14: Performance of several models as the bottleneck dimension of the head increases.

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 2.15, 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 2.5.2), normalized by the Frobenius norm of W:

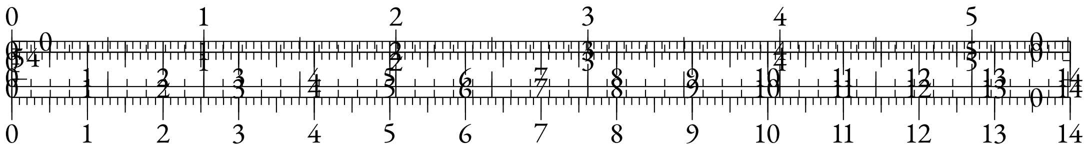
$$W\text{-error}(d) = \frac{||\sigma_{d+1:}||_2}{||W||_F}$$

sources/part_1/softmax_bottleneck/imgs/llama_sv_4gram.png
sources/part_1/softmax_bottleneck/imgs/pythia_sv_4gram.png

(a) Llama-2 tokenizer (b) Pythia tokenizer

Figure 2.15: 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 ϕ^* 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 ϕ^*, 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:

$$W^* = \underset{W\in\mathbb{R}^{V\times\infty}}{\arg\min} \sum_{y\in\mathcal{T}} \sum_i \mathcal{L}(W, x_i^*, y_i) \tag{2.6}$$

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

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

In practice, a neural language model ϕ_θ produces contextual representations $x_i = \phi_\theta(y_{<i})$ of dimension $d \in \mathbb{N}^*$. The linear language modeling head $W_\theta \in \mathcal{R}^{V\times d}$ is trained concurrently with ϕ_θ with the same objective as in Equation 2.6.

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:

$$W_d^* = \underset{W\in\mathbb{R}^{V\times\infty}}{\arg\min} \sum_{y\in\mathcal{T}} \sum_i \mathcal{L}(W, x_i^*, y_i) \text{ s.t. } rank(W) \leq d \tag{2.7}$$

Lemma 2.5.1 shows that by approaching W^* directly, we can asymptotically expect to close the performance gap.

Lemma 2.5.1. *(proof in Section 2.5.1) 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* $\epsilon \to 0$*:*

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

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 2.5.2. *(Eckart–Young–Mirsky theorem) Let's consider* (σ_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* $d < V = rank(W^*)$. *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 2.5.3.

Theorem 2.5.3. *(proof in Section 2.5.1) Let's consider* (σ_i) *the singular values of* W^* *in decreasing order. Then, when* $d \to V$, *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 2.5.3 is not particularly strong, as discussed in Section 2.5.1.

In Figure 2.16, we compare the results of the head bottleneck experiment of the Pythia-1B model in Section 2.5.5 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.

2.5.6 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.

Figure 2.16: 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.

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 2.5.4 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 and find that the emergence of last-layer anisotropy that only affects small models can be correlated with saturation. We also show that the LM heads of small models concurrently suffer from *spectral* saturation, i.e. a uniformization of singular values that leads to a degenerated state.

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

Limitations

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

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

Another limitation of this work is the loose nature of the mathematical connection that we establish between the dimensionality of the ideal language modeling head and the rank-constrained performance (cf. Theorem 2.5.3). Moreover, it can also be argued that considering *ideal* x_i^* representations is an ill-defined notion. We argue that the reasoning behind Theorem 2.5.3 could be applied to any contextual representations, as the *ideal* nature of x_i^* is not necessary in the demonstrations. The word *ideal* reflects that our observations hold for x_i^* representations obtained from *any underlying model*, to an extent that depends on the structure that these representations impose on the W^* matrix for a given training set $\mathcal{T}$.

Acknowledgements

We thank Song Duong for carefully reviewing this article and for his valuable suggestions.

This work was funded by the last author's 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 GENCI operated by IDRIS (allocation 2023-AD011013680R1).

2.5.1 Proofs

Lemma 2.5.1

The proof is mainly based on calculations and limited development:

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

The continuous function $M \longrightarrow \left| -(Mx_i^*)_{y_i} + \frac{\sum_{j\in V} \exp((Mx_i^*)_j)}{\sum_{j\in V} \exp((W^*x_i^*)_j)} \right|$ 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 2.5.3

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:

$$
\left| \sum_{y\in\mathcal{T}} \sum_i \mathcal{L}(W_d^*, x_i^*, y_i) - \mathcal{L}(W^*, x_i^*, y_i) \right| \leq \sum_{y\in\mathcal{T}} \sum_i |\mathcal{L}(W_d, x_i^*, y_i) - \mathcal{L}(W^*, x_i^*, y_i)| \tag{2.8}
$$

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

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

By defining $\varepsilon = W_d - W^*$, we can apply Lemma 2.5.1 and show that:

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

From Equation (2.8), we have that:

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

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

$$0 \leq \sum_{y \in \mathcal{T}} \sum_i \mathcal{L}(W_d^*, x_i^*, y_i) - \mathcal{L}(W^*, x_i^*, y_i) = \left|\sum_{y \in \mathcal{T}} \sum_i \mathcal{L}(W_d^*, x_i^*, y_i) - \mathcal{L}(W^*, x_i^*, y_i)\right|$$

which ends the proof.

Remark The bound used in Equation (2.8) can be rather loose in practice. We can think of no particular reason why approaching W^* directly should be the optimal way to minimize the loss on $\mathcal{T}$. Hence, the presented result should be taken carefully, and we leave the refinement of such an analysis for future work.

2.5.2 Hyperparameters

Constrained head experiments (Figure 2.14)

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

We report the chosen learning rates in Figure 2.17.

2.6 Anisotropy

2.6.1 Introduction

In recent years, deep learning models based on Transformers have led to significant breakthroughs in the field of natural language processing (NLP). These models have demonstrated state-of-the-art performance across a range of tasks, such as language modeling, machine translation, and sentiment analysis. However, despite their successes, they suffer from a phenomenon known as the

Figure 2.17: 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; Biś 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.

2.6.2 Related Work

The general phenomenon of anisotropy in token-based Transformers for language models has been shown in Ethayarajh (2019). Figure 2.18 extends one of their experiment to more architectures. Gao et al. (2019b) shows that the degeneration of representations comes from the distributions of sources/part_1/anisotropy/imgs/cosine_token.png

Figure 2.18: 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; Rajaee and Pilehvar, 2022). Biś 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 (Rajaee and Pilehvar, 2021; Wang et al., 2020a), or at sentence-level (Gao et al., 2021; Yan 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.

2.6.3 Anisotropy in pre-trained Transformers

Character-based NLP

sources/part_1/anisotropy/imgs/cosine_char_based.png

Figure 2.19: 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., 2022b) is downsampling contextualized character representations via a strided convolution before feeding them to a Transformers. It can be trained either with a subword-based objective (CANINE-s) or with a character-level one (CANINE-c).

- MANTa-LM (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.

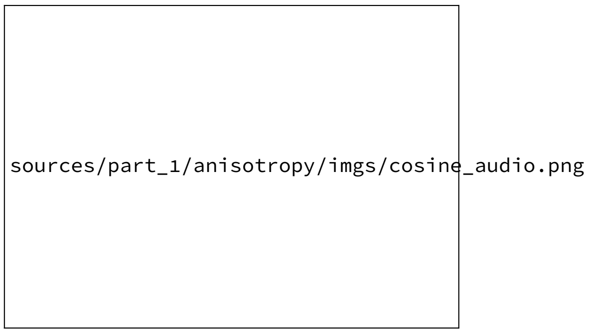
- ByT5 (Xue et al., 2022a) is a version of T5 that is trained at byte-level. To afford for more complex encoding, the authors resize the encoder-decoder architecture.

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.

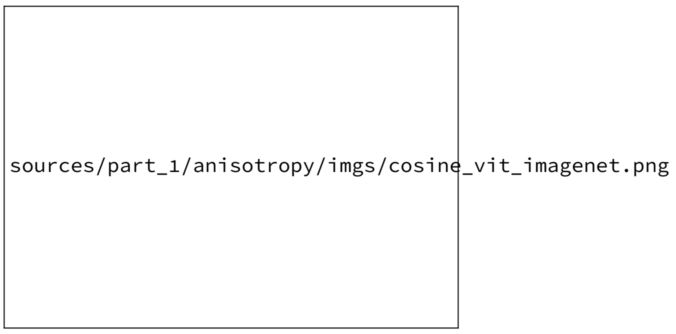
To verify this hypothesis, we compute hidden representations for the validation set of the WikiText-103 corpus (Merity et al., 2017). We then compute the average cosine-similarity between two representations, uniformly taken in the whole validation corpus.

In fact, as shown in Figure 2.19, 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



(a) Speech



(b) Vision

Figure 2.20: 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 subsubsection 2.6.3, 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 2.20.

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

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

sources/part_1/anisotropy/imgs/cosine_cnn_imagenet.png

Figure 2.21: Average cosine-similarity between hidden representations across layers for convolution-based vision models.

We observe in Figure 2.21 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 subsubsection 2.6.3 and subsubsection 2.6.3, 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).

sources/part_1/anisotropy/imgs/pval_vs_cosine_spearman.png

Figure 2.22: 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 2.22, 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 2.6.7.

Interestingly, we notice that the anisotropy affecting most CNN-based vision models is generally not correlated with the drift effect, contrary to Tranformers-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 2.22 also shows a correlation for all character-based models but Canine-C and MANTa-base.

2.6.4 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 Biś 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 2.42b. 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 2.24 and Figure 2.25.

Input vs. output analysis

In Figure 2.23a, 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.

Exploring the Transformer block

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

Figure 2.24 shows that the attention scores tend to move away from zero as the input bias norm increases. Indeed, as the norm of the average $\bar{x}$ of the input embeddings increases, we can expect the query and key vectors Q and K to also display signs of anisotropy. Actually, for each self-attention head, and for all position $i \in [1, L]$, we have:

$$\begin{cases} \mathbb{E}_x(Q_i) = W_Q\bar{x} + b_Q \\ \mathbb{E}_x(K_i) = W_K\bar{x} + b_K \end{cases} \tag{2.9}$$

We can observe in Figure 2.25 that query and key representations indeed increase in norm with the input bias norm. We also notice that the corresponding distributions are anisotropic even when no bias is added, which may be a consequence of BERT's initialization parameters.

Impact of the drift

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

In Figure 2.26, we retrieve softmax values in the self-attention block and for each position, we extract the maximum, the median and the minimum. We then average these values over the whole batch, and repeat for various input bias norm levels. We notice that as the input bias norm increases, the self-attention softmax distributions tend to become less entropic, evolving towards higher maximal probabilities and lower minimal probabilities. In the following analysis, we'll use the term *sharpness* to discuss entropy levels of the self-attention distributions.

This sharpening effect of the attention distributions becomes even clearer if we consider the maximum and minimum values over the whole sequences, as in Figure 2.27.

However, at low anisotropy levels, i.e. when the bias norm is low, we see that the effect is not very important. Figure 2.26 and Figure 2.27 only hint at the fact that the drift of embeddings may help the self-attention to be sharper. Another explanation could be that training favors sharp self-attention patterns, as has been pointed out in previous works (Clark et al., 2019b), which in turn induces a drift in the models' representations. In order to account for that, we need to study the evolution of latent spaces at the self-attention level along training.

2.6.5 Queries and keys: training dynamics

We have established that manually pushing for drift-based anisotropy on *untrained* Transformers models leads to sharper (i.e. low-entropy) self-attention patterns. In this section, we show that this evolution of self-attention values actually takes place during training, and we explore the mechanism behind their appearance. As pointed out in subsection 2.6.4, the self-attention scores result from the QK^T operation, which computes scalar products between query and key representations corresponding to each pair of positions. Thus, in this section, we study the evolution of these query and key representations *along training*, and explore the mechanism behind the increase of the scalar products leading to self-attention scores.

We use the MultiBERT checkpoints (Sellam et al., 2022) with seed 0 to retrieve Q 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 $\hat{Q_s}$ and $\hat{K_s}$ 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 2.28 and Figure 2.29, 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 [6]. 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 2.29a shows that the common direction dynamics appear in the first few steps, while the opposite direction dynamics of Figure 2.29b 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 2.30. We also display some projected Q_s and K_s distributions for several s steps in Figure 2.28.

Figure 2.30 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 2.31, 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 2.32 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.

[6] We actually uniformly sample 20% of the whole set of representations to compute the SVD under reasonable memory constraints.

2.6.6 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 2.6.4 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 2.6.5 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 2.6.5, we use an SVD on the *union* of Q_s and K_s for visualization purposes (see Figure 2.28 and Figure 2.29). 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 2.6.9, 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 2.6.4 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 2.22 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 2.6.7.

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.

2.6.7 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).

2.6.8 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 2.34, 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 2.35, 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.

2.6.9 Other projections for Q_s and K_s

As mentioned in the Discussion (subsection 2.8.6), we reproduce visualizations from subsection 2.6.5 using different projection choices. Namely, we compute the SVD on K_s only in Figure 2.36 and Figure 2.38, and on Q_s only in Figure 2.37 and Figure 2.39.

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 2.36 and Figure 2.37 support our conclusion that the drift directions of keys and queries are aligned during training.

2.6.10 Stability across MultiBERT seeds

`sources/part_1/anisotropy/imgs/scratch_bert_base_input_vs_output.png`

(a) Cosine similarity

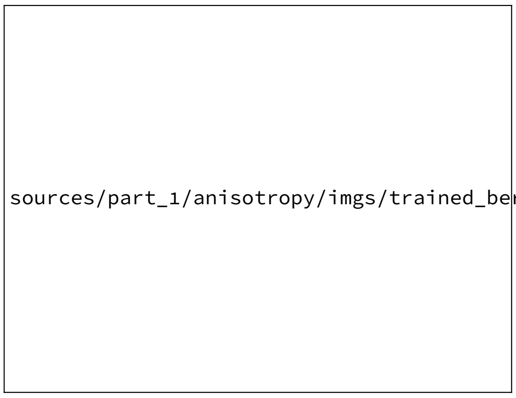
`sources/part_1/anisotropy/imgs/bert_base_norm_v_output.pdf`

(b) Norm

Figure 2.23: Input/Output comparison of a Transformer block from BERT-base as the bias norms increases.

sources/part_1/anisotropy/imgs/trained_bert_base_att_scores.pdf

Figure 2.24: 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.


sources/part_1/anisotropy/imgs/trained_bert_base_bias_vs_kq_cos.png

(a) Cosine sim.

sources/part_1/anisotropy/imgs/trained_bert_base_bias_vs_kq_n

(b) Norm

Figure 2.25: Analysis of the self-attention query and key distributions

sources/part_1/anisotropy/imgs/untrained_bert_base_bias_vs_softmax.png

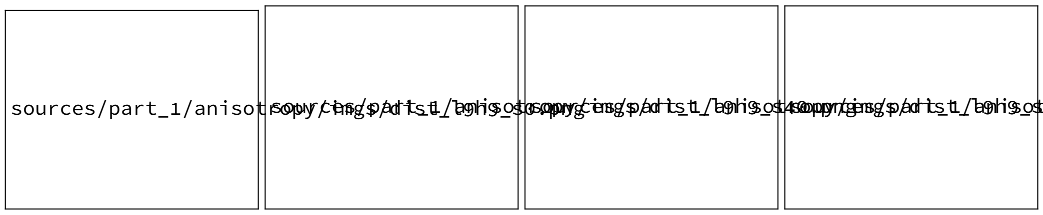
Figure 2.26: Evolution of the self-attention softmax values as the input bias norm increases.

sources/part_1/anisotropy/imgs/untrained_bert_base_bias_vs_max_softmax.png
sources/part_1/anisotropy/imgs/untrained_bert_base_bias_vs_min_softm

(a) Maximum

(b) Minimum

Figure 2.27: Comparison of the extreme values of each sequence averaged over the batch as the bias norm increases.

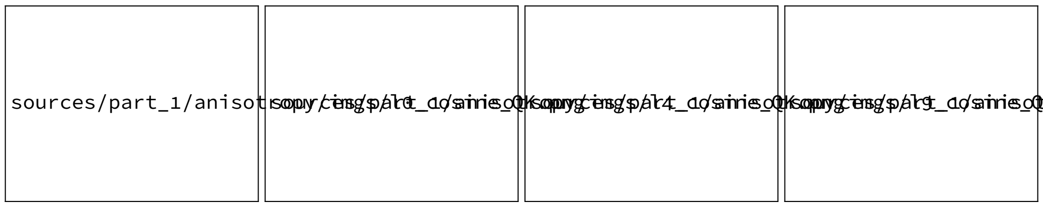


(a) Step 0 (b) Step 40k (c) Step 200k (d) Step 2M (final)

Figure 2.28: Evolution of Q_s and K_s distributions along training. Vectors are projected using a common SVD.

(a) Similar (b) Opposite

Figure 2.29: 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.



(a) Layer 0 (b) Layer 4 (c) Layer 9 (d) Layer 11

Figure 2.30: 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 2.40, and obtain comparable results.

sources/part_1/anisotropy/imgs/l3h8_scalar_QK.png

(a) Similar

sources/part_1/anisotropy/imgs/l9h9_scalar_QK.png

(b) Opposite

Figure 2.31: Evolution of the scalar product between $\bar{Q_s}$ and $\bar{K_s}$ along training. Steps are in thousands.

sources/part_1/anisotropy/imgs/entropy_decay.png

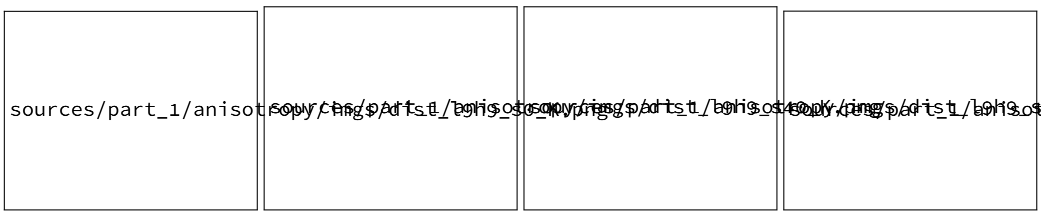
Figure 2.32: Average entropy of the probability distributions corresponding to self-attention rows along training. Each curve corresponds to one layer.

Figure 2.33: 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.

sources/part_1/anisotropy/imgs/cosine_v_density.png

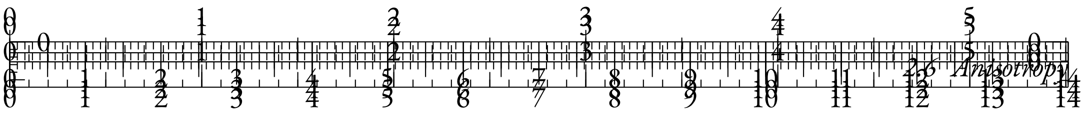
Figure 2.34: Density function of cosine-similarity for a normal distribution as the dimension increases.

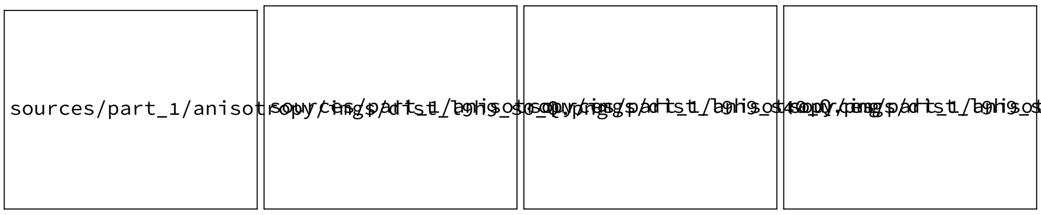
Figure 2.35: 95th quartile of the cosine-similarity distribution on a normal distribution as the dimension increases. We add points for the average cosine-similarity level of Transformers models for several modalities.



(a) Step 0 (b) Step 40k (c) Step 200k (d) Step 2M (final)

Figure 2.36: Evolution of Q_s and K_s distributions along training. Vectors are projected using the SVD computed on K_s.




sources/part_1/anisotropy/imgs/dist_l9h9_s0_Q.png
sources/part_1/anisotropy/imgs/dist_l9h9_s40000_Q.png
sources/part_1/anisotropy/imgs/dist_l9h9_s200000_Q.png
sources/part_1/anisotropy/imgs/dist_l9h9_s2000

(a) Step 0 (b) Step 40k (c) Step 200k (d) Step 2M (final)

Figure 2.37: Evolution of Q_s and K_s distributions along training. Vectors are projected using the SVD computed on Q_s.

sources/part_1/anisotropy/imgs/l0h3_samedir_QK_K.png
sources/part_1/anisotropy/imgs/l9h5_diffdir_QK_K.png

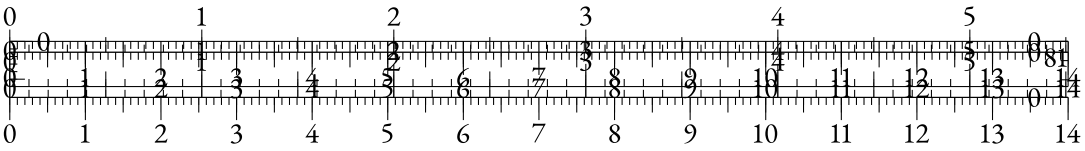
(a) Similar (b) Opposite

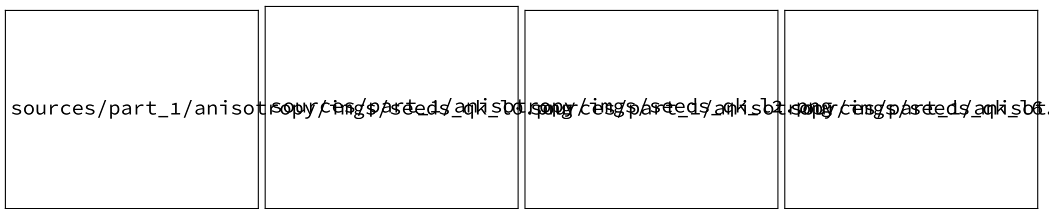
Figure 2.38: Evolution of $\bar{Q_s}$ and $\bar{K_s}$ along training for two different heads in the network, projected via the SVD of K_s.pip3 install ninja einops packaging

sources/part_1/anisotropy/imgs/l0h3_samedir_QK_Q.png
sources/part_1/anisotropy/imgs/l9h5_diffdir_QK_Q.png

(a) Similar (b) Opposite

Figure 2.39: 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.





(a) Layer 0 (b) Layer 2 (c) Layer 6 (d) Layer 11

Figure 2.40: 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.

2.7 Headless

2.7.1 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 2.41: 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× acceleration of the training throughput, and up to 20× 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[7], 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.

2.7.2 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., 2019b) initiated the use of pretraining approaches based on a contrastive learning objective, an idea that has obtained success in many modalities over the years (Sermanet et al., 2018; Schneider et al., 2019; Baevski et al., 2020b; Algayres et al., 2022). In NLP, contrastive learning has

[7]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.

2.7.3 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 $T_\theta : \mathbb{R}^{N\times L\times D} \to \mathbb{R}^{N\times L\times D}$ both based on a set of parameters $\theta \in \mathbb{R}^P$.

A classical language modeling approach consists in selecting a subset of tokens $X_\mathcal{S} = (x_{i,j})_{i,j\in\mathcal{S}}$, and then estimating a probability distribution over the token vocabulary for these tokens from the $(\tilde{x}_{i,j})$ sequences, using e_θ and T_θ. Learning occurs as $X_\mathcal{S}$ is partially altered in $(\tilde{x}_{i,j})$ (e.g. in Masked Language Modeling) or internally in T_θ (e.g. decoder models), and contextual information is essential for e_θ and T_θ to accurately estimate the tokens in $X_\mathcal{S}$.

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

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

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:

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

Headless modeling

While weight tying does not use additional parameters, the projection e_θ^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_θ^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

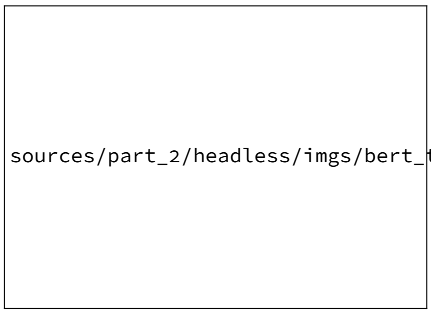
We can easily adapt the Causal Language Modeling (CLM) objective using the Contrastive Weight Tying approach. Negative samples correspond to every input embedding at a different position in the batch. However, the resulting model is not directly able to generate text, as it has no projection head towards $\mathbb{R}^V$. A way to retrieve language generation capacities is to use the input embedding matrix transpose e_θ^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_θ^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[8]. 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

[8]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_θ^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 $K < V$.



(a) Training latency

(b) Memory use

Figure 2.42: Comparison of time and memory complexities of a BERT-base model on a single RTX 8000 GPU.

In Figure 2.42, 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.

2.7.4 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 2.7.4.

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 (Lefaudeux 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	85.87	54.66	83.7	92.45	88.38	89.57	82.4	82.43 (±0.12)
Headless	4.1	444	85.31	58.35	84.54	**93.23**	89.49	89.62	82.54	83.29 (±0.15)
Headless	8.2	888	**86.89**	**60.72**	**85.98**	92.56	**89.75**	**89.81**	**82.87**	**84.08** (±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	**77.8**	60.2	64.9	67.9 (±0.4)
Headless	**69.8**	74.7	**62.7**	**67.2**	**68.6** (±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.

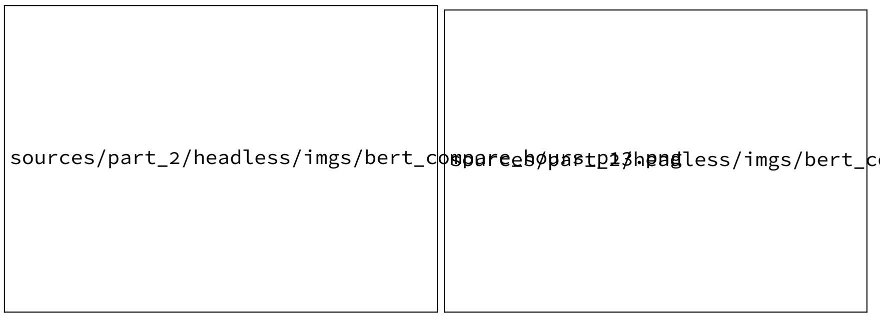
We evaluate on the GLUE benchmark, where we exclude the RTE dataset due to high standard deviations in the obtained scores. We fine-tune our models for 10 epochs on every dataset, and compute validation metrics once every fine-tuning epoch. We use the AdamW optimizer with a learning rate of 10^{-5}, a weight decay of 0.01 and a balanced cross-entropy loss objective. See Section 2.7.5 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 pretraining, 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 2.43. 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

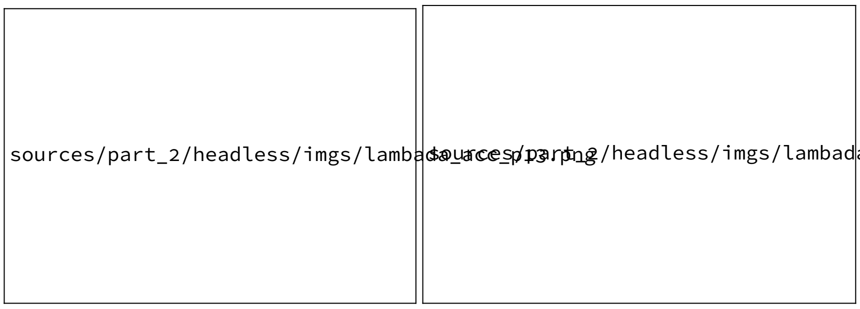
We pretrain Pythia-70M architectures for English, sticking to the Pythia procedure (Biderman et al., 2023b) as much as possible. We use OpenWebText2 as a pretraining dataset. We train on 143,000 batches of 1,024 sequences of length 2,048 split over 16 V100 GPUs. We use exactly the same hyperparameters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

As mentioned in Section 2.7.3, we fine-tune our headless models for CLM with an LM head initialized with e_θ^T for 10000 steps using an effective batch size of 256 ($4\times$ smaller that during pretraining), a learning rate of 10^{-4}, and a constant learning rate schedule with 2000 linear warm-


sources/part_2/headless/imgs/bert_compare_hours_p13.png
sources/part_2/headless/imgs/bert_compare_tokens_p13.png

(a) Pretraining hours (b) Pretraining tokens

Figure 2.43: Comparison of GLUE average scores along pretraining.


sources/part_2/headless/imgs/lambada_acc_p13.png
sources/part_2/headless/imgs/lambada_ppl_p13.png

(a) Accuracy (b) Perplexity

Figure 2.44: 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 2.44.

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	**3.143**	170.23	19.52
Headless	-	524.44	18.26
Headless + FT	3.283	**153.5**	**22.2**

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)	**66.4** (±1.5)	43.8 (±1.6)	41.9 (±4.8)
HLM + FT	1052 (61%)	**53.0**† (±0.9)	**56.0** (±1.2)	**26.0** (±1.4)	64.5 (±1.5)	**47.5**† (±1.6)	**66.0**† (±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. †: 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.

2.7.5 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[9] trained by Sanh et al. (2019). This model has 137M parameters, and uses a vocabulary of 119k tokens. As in Section 2.7.4, 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 2.7.4.

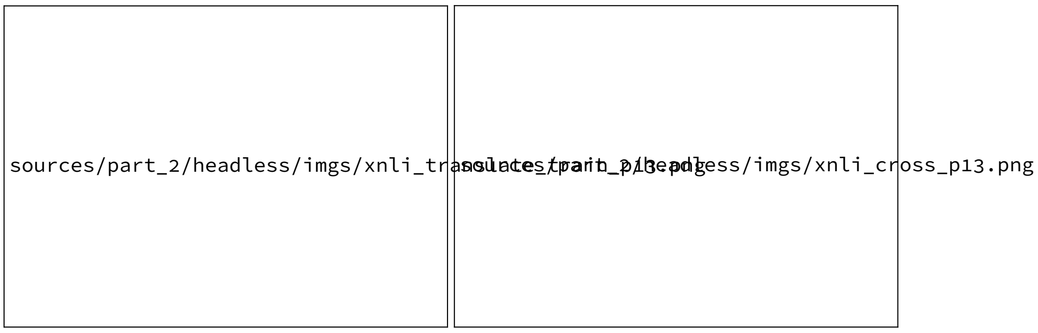
Models are then fine-tuned on the XNLI dataset, for both cross-lingual zero-shot transfer from English and target language fine-tuning. Fine-tuning hyperparameters can be found in Section 2.7.5.

We display final results in Figure 2.45. 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.

[9]Available at `https://huggingface.co/distilbert-base-multilingual-cased`

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

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


sources/part_2/headless/imgs/xnli_tr
sources/part_2/headless/imgs/xnli_cross_p13.png

(a) Translate-Train: target language fine-tuning (b) Translate-Test: English fine-tuning

Figure 2.45: 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 2.45, 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 2.45a, and 22% in Figure 2.45b.

2.7.6 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_θ 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 **?** and the CC-News dataset (Hamborg et al., 2017). Hyperparameter details can be found in Section 2.7.4. 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.

sources/part_2/headless/imgs/bert_vocabsize_p13.png

sources/part_2/headless/imgs/speed_vocabsize_p13.png

(a) GLUE average score

(b) Training speed

Figure 2.46: Comparison of downstream performance and training speed for small models trained using different token vocabulary sizes.

Figure 2.46 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 2.46b 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 2.7.3, 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 2.7.4. HLMs are fine-tuned similarly to Section 2.7.4.

sources/part_2/headless/imgs/batch_size_hours_p13.png

Figure 2.47: LAMBADA accuracy along pretraining for different batch sizes.

In Figure 2.47, 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 2.7.4 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 ↑	GLUE avg.
Cross-Entropy	x1	x1	82.45
Cross-Entropy + CWT	x1	x0.87	82.93
NCE (wo/ WT)	x1.57	**x2.47**	82.91
CWT	x1	**x2.13**	**83.37**

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.

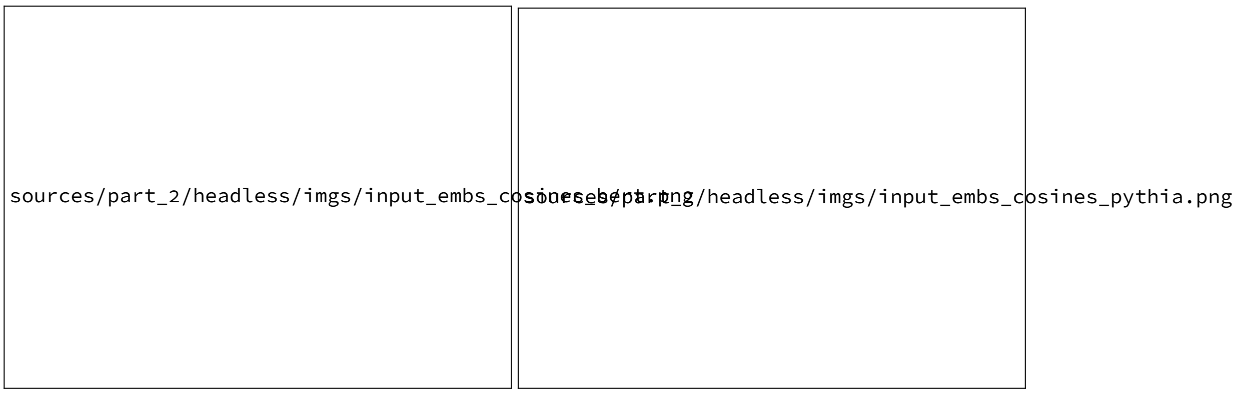
This work was funded by the last author's chair in the PRAIRIE institute, itself funded by the French national agency ANR as part of the "Investissements d'avenir" programme under the reference ANR-19-P3IA-0001.

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

2.7.1 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 2.48.


sources/part_2/headless/imgs/input_embs_cosines_bert.png
sources/part_2/headless/imgs/input_embs_cosines_pythia.png

(a) Monolingual encoders

(b) Monolingual decoders

Figure 2.48: Cosine-similarity distributions for pairs of WordNet synonyms.

In Figure 2.48, 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.

2.7.2 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.

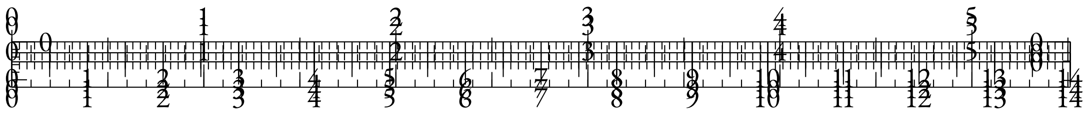
It could be opposed to this paper that as architectures grow in size, the proportion of compute that is associated with the output vocabulary projection shrinks. While we acknowledge that this effect may reduce the advantage of HLMs in terms of training throughput, our experiments show that HLMs are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

2.7.3 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 2.8.6, 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.



2.7.4 Pretraining hyperparameters

Monolingual encoders

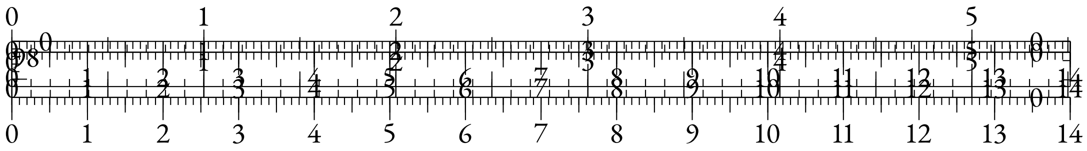
Dataset	OpenWebText2
Architecture	`bert-base-uncased`
Tokenizer	`pythia-70m-deduped`
Optimizer	AdamW
Learning rate	1e-4
Precision	16
Weight decay	0.01
Gradient clipping	1
Device batch size	32 / 64
Batch size	256 / 512
Sequence length	128
LR schedule	Triangular
Warmup steps	10000
Nb. steps	1000000

Table 2.8: Pre-training hyperparameters used for the monolingual encoders. When they differ between vanilla and headless models, we provide separate values formatted as (vanilla / headless). Model names written as `model-name` refer to their HuggingFace release.

Monolingual decoders

Dataset	OpenWebText2
Architecture	`pythia-70m-deduped`
Tokenizer	`pythia-70m-deduped`
Optimizer	AdamW
Adam ϵ	1e-8
Adam (β_1, β_2)	(0.9, 0.95)
Learning rate	1e-3
Precision	16
Weight decay	0.1
Gradient clipping	1
Device batch size	8 / 8
Batch size	1024 / 1024
Sequence length	2048
LR schedule	Cosine
Warmup steps	1430
Nb. steps	143000

Table 2.9: Pre-training hyperparameters used for the monolingual encoders. When they differ between vanilla and headless models, we provide separate values formatted as (vanilla / headless).



Multilingual encoders

Dataset	Wikipedia (multilingual)
Architecture	`distilbert-base-multilingual-cased`
Tokenizer	`distilbert-base-multilingual-cased`
Optimizer	AdamW
Learning rate	2e-4
Precision	16
Weight decay	0.01
Gradient clipping	1
Device batch size	64
Batch size	64
Sequence length	128
LR schedule	Triangular
Warmup steps	10000
Nb. steps	400000

Table 2.10: Pre-training hyperparameters used for the multilingual encoders.

Small monolingual encoders

Dataset	CC-News
Architecture	`google/bert_uncased_L-4_H-512_A-8`
Tokenizer	`google/bert_uncased_L-4_H-512_A-8`
Optimizer	AdamW
Learning rate	2e-4
Precision	16
Weight decay	0.01
Gradient clipping	1
Device batch size	64
Batch size	64
Sequence length	128
LR schedule	Triangular
Warmup steps	10000
Nb. steps	400000

Table 2.11: Pre-training hyperparameters used for the small monolingual encoders used in Figure 2.46.

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 2.47. These models rely on the GPT-2 architecture with a few changes. These changes scale down the model size to 11M parameters.

2.7.5 Finetuning hyperparameters

Balanced cross-entropy

We have noticed that using balanced cross-entropy loss for fine-tuning could further improve the performance of <u>all</u> 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	**86.27**	49.33	82.06	92.37	88.62	89.49	82.35	81.5 ($\pm$0.14)
Headless	85.8	**56**	**84.85**	**93.23**	**89.67**	**89.77**	**83.05**	**83.19** ($\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.

2.7.6 Representing synonyms

In this section,

2.7.7 Implementation

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

2.8 MANTa

2.8.1 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 sources/part_2/headless/imgs/cwt_implementation.png

Figure 2.49: PyTorch implementation of the Contrastive Weight Tying loss.

of tokens (substrings of smaller length) which can then be embedded by the model. Subword tokenization algorithms (Sennrich et al., 2016; Wu et al., 2016; Kudo, 2018) are a specific class of tokenizers designed in such a way that almost every string can be encoded and decoded with very few out-of-vocabulary tokens. They are used in the vast majority of recent LMs, but have been an essential part of NLP systems since much longer (Mielke et al., 2021a).

The success of these algorithms can be attributed to several reasons. Firstly, they produce token sequences whose length is greatly reduced compared to the original character sequence. This characteristic is helpful because limitations in compute power and architectural constraints, such as the quadratic complexity with respect to sequence length of Transformers (Vaswani et al., 2017), prevent models from processing arbitrary long sequences. Secondly, they compress the corpus using occurrence statistics that may help LMs. For instance, if a word appears frequently in the training corpus, it will be encoded as a single token in the vocabulary and the model will be able to build a representation for that particular token more easily.

However, the induced biases of tokenizers are also harmful for modelization. One such limitation lies in their brittleness to character deformations which are commonly found in real world, noisy data. For instance, BERT's tokenizer (Devlin et al., 2019) encodes "performance" as `["performance"]` but "perfonmance" as `['per', '##fo', '##n', '##man', '##ce']`, which makes it hard for the model to behave similarly in both cases. Moreover, the tokenizer is fixed after its training and is therefore impossible to update, for instance to reflect new domains (El Boukkouri et al., 2020) where tokenization might over-segment specific or technical terms. Clark et al. (2022a) list other issues emerging when using static subword tokenizers, especially when modeling languages with a more complex morphology than English.

To overcome these issues, *tokenization-free* models (Clark et al., 2022a; Xue et al., 2022b; Tay et al., 2021) produce character-based or byte-based embeddings for LMs instead of subword embeddings. These methods improve the robustness of LMs to naturally occurring noise as well as their expressiveness when dealing with out-of-domain or multilingual data. In order to cope with increased input lengths, some of these methods compress sequences with constant reduction rates obtained using specialized modules (Clark et al., 2022a; Tay et al., 2021), subsequently removing any notion of subwords.

sources/part_2/headless/imgs/train_implementation.png

Figure 2.50: PyTorch implementation of the computation of the training loss for headless causal LMs. The implementation of the MLM equivalent is straightforward.

sources/part_2/manta/images/full_difftok_schema.pdf

Figure 2.51: 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.

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

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

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

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

2.8.2 Related Work

Non-neural subword-level tokenization methods have dominated in the last few years as the default way to encode textual data, the most used being BPE (Sennrich et al., 2016), WordPiece (Wu et al., 2016) and Unigram (Kudo, 2018). However, they have inherent flaws that limit their multilingual performance (Rust et al., 2021), their adaptability to new languages and new domains after pre-training (El Boukkouri et al., 2020; Garcia et al., 2021) and the downstream performance of language models in general (Bostrom and Durrett, 2020).

To alleviate these issues, tokenization-free (or character-level) models leverage characters instead of subwords to build text representations. Some of the first neural networks for sequence generation used characters directly as inputs (Sutskever et al., 2011; Graves, 2013), and following works modified the approach to create input word representations based on characters (Kim et al., 2016; Józefowicz et al., 2016; Peters et al., 2018). Similar architectures were recently adapted to work with Transformers (El Boukkouri et al., 2020; Ma et al., 2020). Nevertheless, they still rely on fixed tokenization heuristics (for instance segmenting using whitespaces) which may not be suited to some languages or certain types of language variations. Recent works have tried to remove these induced biases by working purely with characters or bytes as input (Clark et al., 2022a; Tay et al., 2021; Xue et al., 2022b). However, they either have to use various tricks to reduce the sequence lengths based on other induced biases like downsampling rates (Clark et al., 2022a; Tay et al., 2021) or have extremely low training and inference speeds (Xue et al., 2022b). Chung et al. (2016) create tokens in a differentiable manner by predicting frontiers and using the representations of each character inside a "token", but it remains unclear how their model could be adapted to be used with newer architectures such as Transformers. Mofijul Islam et al. (2022) propose to segment tokens using a trained "frontier predictor." Nevertheless, this differentiable tokenizer is not trained with the main language model objective but instead mimics a BPE subword tokenizer, carrying some of its flaws.

2.8.3 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;[10]
- 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 2.51. We also summarize the process of mapping byte embeddings to block embeddings in appendix 2.8.4.

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 $256 \times \mathit{hidden\ size}$ 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.

[10] 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: $(B,\ b_i) \sim \mathcal{PB}\Big((p_{F_k})_{k\leq i}\Big)$. 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 $Z = \sum\limits_{1\leq k\leq L_B} G_{\mu_i,\sigma_i}(k)$ is a normalization term, and:

$$\begin{cases} L_B = \min(L, (\mu_L + 3\sigma_L)) \\ \mu_i = \sum_{k=1}^{i} p_{F_i} \\ \sigma_i = \sqrt{\sum_{k=1}^{i} p_{F_i}(1 - p_{F_i})} \end{cases} \tag{2.10}$$

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 2.52. 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 2.8.3 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	$\lvert\theta\rvert$	MNLI	QNLI	MRPC	SST-2	QQP	STSB	COLA	AVG
T5$_{Small}$	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$_{Small}$ (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^B_k, the embedding of block B_k in k-th position, we first compute the weighted byte embeddings $\left(P_{k,i} \times e^b_i\right)_{i\in[1,L]} \in \mathbb{R}^H$, 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 2.8.1.

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 2.8.6.

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,

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

Table 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$_{Small}$ and MANTa-LM$_{Base}$. Each of them stacks a MANTa$_{Small}$ (resp. MANTa$_{Base}$) tokenizer and embedding module and a T5$_{Small}$ (resp. T5$_{Base}$) encoder-decoder model stripped of its tokenizer and subword embedding matrix. Details about MANTa hyperparameters can be found in Appendix 2.8.2.

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$_{Small}$, MANTa-LM$_{Small}$, and MANTa-LM$_{Base}$ 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$_{Small}$ 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$_{Small}$. Hence, our ByT5$_{Small}$ is only trained for 200k steps.

Model	Accuracy
BERT$^{\ddagger}_{Base}$	77.7
CharacterBERT$^{\ddagger}_{Base}$	77.9
T5$_{Small}$	75.3
MANTa-LM$_{Small}$ (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.

2.8.4 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.

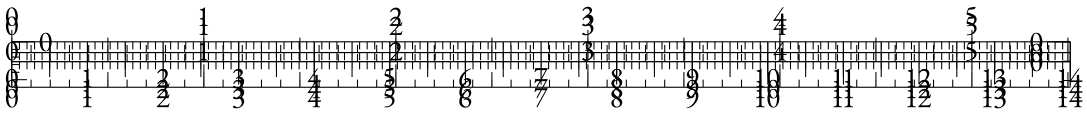
We compare our model to an encoder-decoder model with subword tokenization (pre-trained with the same denoising objective as T5) and a fully byte-level encoder-decoder, similar to ByT5. We compare *Small* models with our pre-trained versions, and *Base* models with results mentioned in Tay et al. (2021). We report the number of parameters given in Tay et al. (2021) for Byte-level T5$_{Base}$, and gather from its low value that their implementation corresponds to a T5$_{Base}$ 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

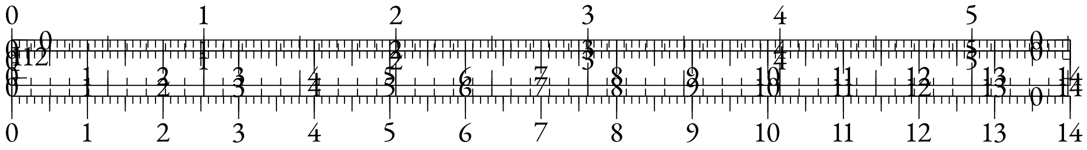
Setup Similarly to Tay et al. (2021), we test our model on a toxicity detection task constructed with user generated data. We use the ToxicComments dataset (Wulczyn et al., 2017) which contains 223,549 sentences annotated with a binary label indicating whether each sentence can be classified as toxic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



Model	Accuracy
$\text{T5}^{\dagger}_{Base}$	91.5
$\text{Charformer}^{\dagger}_{Base}$	92.7
MANTa-LM_{Base} (ours)	**93.2**

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

Model	$\|\theta\|$	Seconds/step
Byte-level T5_{Small}	57M	9.06 ($\times$ 8.0)
MANTa-LM_{Small}	57M	2.61 ($\times$ 2.3)
T5_{Small}	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

2.8.5 Training Speedups

In terms of speed, we compare our model to MANTa-LM_{Small} to T5_{Small} 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_{Small} architecture as described in Xue et al. (2022b).

MANTa-LM is approximately 4 times faster than Byte-level T5_{Small}, and 5 times faster than ByT5_{Small}, 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_{Small} which furthermore benefits from already tokenized sequences at training time.

2.8.6 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 2.8.3. 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

Original	Oh, it's me vandalising?xD See here. Greetings,
MANTa	`Oh|,| it|'s| me| vandalising?|xD| See| here|.| Greetings|,`
T5 tokenizer	`Oh|,| it|'|s| me| van|d|al|ising|?|x|D| See| here|.| |Greeting|s|,`
Original	The patient was started on Levophed at 0.01mcg/kg/min.
MANTa	`The| patient| was| started| on| Levophed| at| 0|.01mcg|/kg|/min|.`
T5 tokenizer	`The| patient| was| |started| on| Le|vo|p|he|d| at| 0.|01|m|c|g|/|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., 2022a) or Charformer (Tay et al., 2021). While CANINE downsamples sequences using a fixed rate after byte contextualization and Charformer's GBST (Gradient Based Subword Tokenizer) pools representations created using various downsampling rates, MANTa only applies downsampling right before the LM to limit the length of block sequences. Hence, our model is able to build word-level representations of *arbitrary length* as long as it divides the whole byte sequence length by a fixed factor.

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

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.

2.8.7 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. (2022a); Xue et al. (2022b); Tay et al. (2021)) which adds an important computational overhead at generation time. Previous works have attempted to reduce this computational cost by decreasing the size of the decoder layers compared to the encoder (Xue et al., 2022b) or by projecting embeddings to a smaller latent space (Jaegle et al., 2021) for the decoding.

Finally, we presented in this work a proof of concept of adaptive segmentation algorithms on relatively small models, ranging from 50M to 200M parameters. Although we hypothesize that our model would scale relatively well since it keeps most of the encoder-decoder architecture untouched, this hypothesis should be tested in a future work.

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 by GENCI to the HPC resources of IDRIS under the allocation 2022-AD011012676R1.

2.8.1 Improving Pooling Speed

Applying the 1-D convolution requires computing and storing $\mathcal{O}(L_B \times L \times H)$ parameters since we apply the 1D-convolution on every row of the weighted embedding map $P(e^b)^T$. Therefore, this operation may be particularly costly, especially if the frontier predictor outputs a high number of blocks. However, we can use the fact that the weighted embedding map has a special form to reduce the memory load when computing the convolution. Let K be the convolution kernel size, $(C_j)_{j\in[1,K]} \in \mathbb{R}^{K\times H}$ the convolution filters and "$\cdot$" denote the element-wise product. Then, omitting padding and biases :

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

Notice how the product between the convolution filters and the byte embeddings $C_j \cdot e_{i+j}^b \in \mathbb{R}^H$ does not depend on the block anymore. We cache this computation, storing $\mathcal{O}(K \times L \times H)$ parameters and only later apply the convolution per block by summing these products with the block-byte membership map P. Caching greatly lowers the speed and memory requirements of MANTa, allowing to save $L_B - 1$ element-wise products.[11] K is usually small, so the products can be stored easily.

[11]This caching would be exactly similar if the convolution was not depthwise.

2.8.2 Hyperparameters

Hyperparameter	MANTa$_{Small}$	MANTa$_{Base}$
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$_{Small}$	T5$_{Small}$	T5$_{Base}$
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

`sources/part_2/manta/images/mapping_example_0.png`

Step 0

`sources/part_2/manta/images/mapping_example_3000.png`

Step 3,000

`sources/part_2/manta/images/mapping_example_7000.png`

Step 7,000

`sources/part_2/manta/images/mapping_example_13000.png`

Step 13,000

Figure 2.52: 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.

sources/part_2/manta/images/noise_curve_dev.pdf

(a) Dev-Only

sources/part_2/manta/images/noise_curve.pdf

(b) Train-Dev

Figure 2.53: 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.

2.8.3 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).

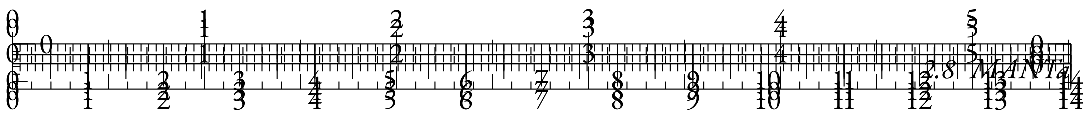
Model	$\|\theta\|$	MNLI	QNLI	MRPC	SST-2	QQP	STSB	COLA	AVG
MANTa-LM$_{Base}$ (ours)	200M	78.1/78.2	88.6	83.6/88.6	91.0	70.7/88.6	74.1	45.0	78.7

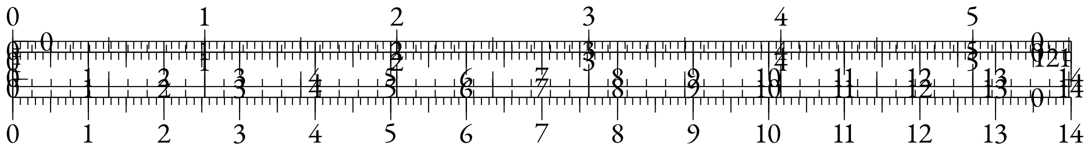
Table 2.26: Results on test sets for the GLUE benchmark.

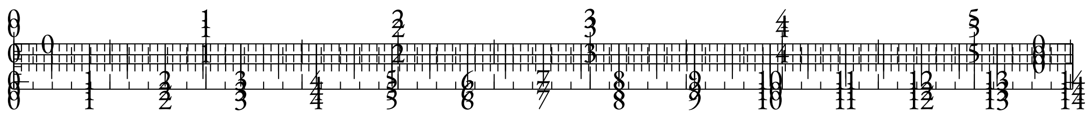
2.8.4 MANTa Module

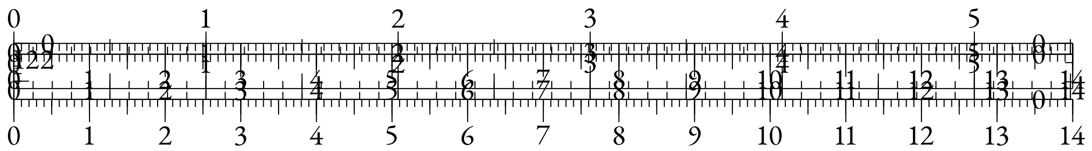
sources/part_2/manta/images/MANTa_Tokenization_Module.pdf

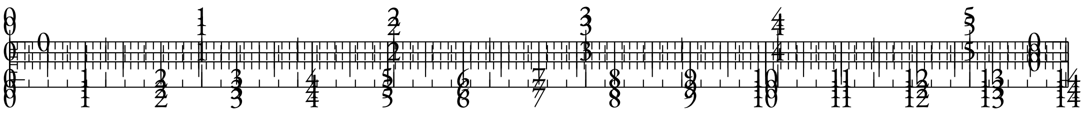
Figure 2.54: A detailed view of the MANTa module described in section 2.8.3. We denote by h_b the dimension of the byte embeddings, by h_{LM} the dimension of the block embeddings that will be fed to the encoder-decoder model, L the length of the input sequence and L_B the length of the block sequence. We omit batch sizes for simplicity.


2.8
MANTa




0 1 2 3 4 5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14


0 1 2 3 4 5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14



Bibliography

Abubakar Abid, Maheen Farooqi, and James Zou. 2021. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, page 298–306, New York, NY, USA. Association for Computing Machinery.

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant Nair, Ilya Soloveychik, and Purushotham Kamath. 2024. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. In *Proceedings of Machine Learning and Systems*, volume 6, pages 114–127.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Mira Ait-Saada and Mohamed Nadif. 2023. Is anisotropy truly harmful? a case study on text clustering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1194–1203, Toronto, Canada. Association for Computational Linguistics.

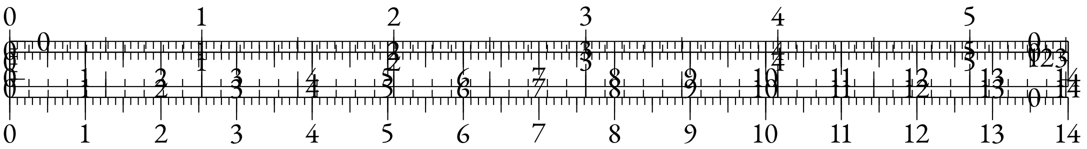
Robin Algayres, Tristan Ricoul, Julien Karadayi, Hugo Laurençon, Salah Zaiem, Abdelrahman Mohamed, Benoît Sagot, and Emmanuel Dupoux. 2022. DP-parse: Finding word boundaries from raw speech with an instance lexicon. *Transactions of the Association for Computational Linguistics*, 10:1051–1065.

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.

Mark Aronoff and Kirsten Fudeman. 2022. *What is morphology?* John Wiley & Sons.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.



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

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

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

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

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020b. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Advances in Neural Information Processing Systems*, volume 33, pages 12449–12460. Curran Associates, Inc.

Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. 2010. A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology*, 1(1):4.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

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

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT ’21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems*, 13.

Yoshua Bengio and Jean-Sébastien Senecal. 2003. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, volume R4 of *Proceedings of Machine Learning Research*, pages 17–24. PMLR. Reissued by PMLR on 01 April 2021.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023a. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

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

Geetanjali Bihani and Julia Rayz. 2021. Low anisotropy sense retrofitting (LASeR) : Towards isotropic and sense enriched representations. In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 81–95, Online. Association for Computational Linguistics.

Richard Billingsley and James Curran. 2012. Improvements to training an RNN parser. In *Proceedings of COLING 2012*, pages 279–294, Mumbai, India. The COLING 2012 Organizing Committee.

Daniel Biś, Maksim Podkorytov, and Xiuwen Liu. 2021. Too much in common: Shifting of embeddings in transformer language models and its implications. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5117–5130, Online. Association for Computational Linguistics.

William Biscarri, Sihai Dave Zhao, and Robert J. Brunner. 2018. A simple and fast method for computing the poisson binomial distribution function. *Computational Statistics & Data Analysis*, 122:92–100.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Rishi Bommasani, Kelly Davis, and Claire Cardie. 2020. Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4758–4781, Online. Association for Computational Linguistics.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4617–4624, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners.

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

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Haw-Shiuan Chang and Andrew McCallum. 2022. Softmax bottleneck makes language models unable to represent multi-mode word distributions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8048–8073, Dublin, Ireland. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014a. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2024. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1).

Gobinda G Chowdhury. 2010. *Introduction to modern information retrieval*. Facet publishing.

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

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Rob Churchill and Lisa Singh. 2022. The evolution of topic modeling. *ACM Computing Surveys*, 54(10s):1–35.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019a. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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*.

Lionel Clément and Éric Villemonte de La Clergerie. 2005. MAF: a Morphosyntactic Annotation Framework. In *2nd Language & Technology Conference (LTC'05)*, 2nd Language & Technology Conference (LTC'05), pages 90–94, Poznan, Poland.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018a. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018b. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

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.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107.

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.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Manuel Faysse, Patrick Fernandes, Nuno M. Guerreiro, António Loison, Duarte M. Alves, Caio Corro, Nicolas Boizard, João Alves, Ricardo Rei, Pedro H. Martins, Antoni Bigata Casademunt, François Yvon, André F. T. Martins, Gautier Viaud, Céline Hudelot, and Pierre Colombo. 2024. Croissantllm: A truly bilingual french-english language model.

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

R. A. Fisher. 1922. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222:309–368.

Hao Fu, Shaojun Zhou, Qihong Yang, Junjie Tang, Guiquan Liu, Kaikui Liu, and Xiaolong Li. 2021. Lrc-bert: Latent-representation contrastive knowledge distillation for natural language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12830–12838.

Kunihiko Fukushima. 1969. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019a. Representation degeneration problem in training natural language generation models.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. 2019b. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Nathan Godey, Éric de la Clergerie, and Benoît Sagot. 2024. Anisotropy is inherent to self-attention in transformers.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Andreas Grivas, Nikolay Bogoychev, and Adam Lopez. 2022. Low-rank softmax can have unargmaxable classes in theory but rarely in practice. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6738–6758, Dublin, Ireland. Association for Computational Linguistics.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2018. Cognate-aware morphological segmentation for multilingual neural translation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 386–393, Belgium, Brussels. Association for Computational Linguistics.

E.J. Gumbel. 1935. Les valeurs extrêmes des distributions statistiques. *Annales de l'institut Henri Poincaré*, 5(2):115–158.

Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. 2023. Visual attention network. *Computational Visual Media*, 9(4):733–752.

Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. Distributional vectors encode referential attributes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 12–21, Lisbon, Portugal. Association for Computational Linguistics.

Wes Gurnee and Max Tegmark. 2024. Language models represent space and time. In *The Twelfth International Conference on Learning Representations*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy. PMLR.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. 2017. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223.

Katharina Hämmerl, Alina Fastowski, Jindřich Libovický, and Alexander Fraser. 2023. Exploring anisotropy and outliers in multilingual language models for cross-lingual semantic sentence similarity. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7023–7037, Toronto, Canada. Association for Computational Linguistics.

R. W. Hamming. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160.

ZS Harris. 1954. Distributional structure.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. {DEBERTA}: {DECODING}-{enhanced} {bert} {with} {disentangled} {attention}. In *International Conference on Learning Representations*.

Dan Hendrycks and Kevin Gimpel. 2023. Gaussian error linear units (gelus).

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, San Diego, California. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

A. E. Hoerl and R. W. Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 29:3451–3460.

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.

Matthew A. Jaro. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference* *on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. 2022. Understanding dimensional collapse in contrastive self-supervised learning. In *International Conference on Learning Representations*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

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

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st edition. Prentice Hall PTR, USA.

Sekitoshi Kanai, Yasuhiro Fujiwara, Yuki Yamanaka, and Shuichi Adachi. 2018. Sigsoftmax: Reanalysis of the softmax bottleneck. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

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

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

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

Arne Köhn. 2015. What's in an embedding? analyzing word embeddings through multilingual evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073, Lisbon, Portugal. Association for Computational Linguistics.

Hadas Kotek, Rikker Dockum, and David Sun. 2023. Gender bias and stereotypes in large language models. In *Proceedings of the ACM collective intelligence conference*, pages 12–24.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

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

N. Kishore Kumar and J. Schneider. 2017. Literature survey on low rank approximation of matrices. *Linear and Multilinear Algebra*, 65(11):2212–2244.

Sachin Kumar and Yulia Tsvetkov. 2019. Von mises-fisher loss for training sequence to sequence models with continuous outputs. In *Proc. of ICLR*.

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.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

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`.

VI Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10, page 707.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online. Association for Computational Linguistics.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning.

Davis Liang, Hila Gonen, Yuning Mao, Rui Hou, Naman Goyal, Marjan Ghazvininejad, Luke Zettlemoyer, and Madian Khabsa. 2023. Xlm-v: Overcoming the vocabulary bottleneck in multilingual masked language models.

Davis Liang, Hila Gonen, Yuning Mao, Rui Hou, Naman Goyal, Marjan Ghazvininejad, Luke Zettlemoyer, and Madian Khabsa. 2023. XLM-V: Overcoming the Vocabulary Bottleneck in Multilingual Masked Language Models. *arXiv e-prints*, page arXiv:2301.10472.

Anne-Laure Ligozat, Julien Lefevre, Aurélie Bugeau, and Jacques Combaz. 2022. Unraveling the hidden environmental impacts of ai solutions for environment life cycle assessment of ai solutions. *Sustainability*, 14(9).

Ying-Chen Lin. 2021. Breaking the softmax bottleneck for sequential recommender systems with dropout and decoupling.

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.

Robert E. Longacre. 1970. *Hierarchy in Language*, pages 173–196. De Gruyter Mouton, Berlin, Boston.

Max M. Louwerse and Nick Benesh. 2012. Representing spatial structure through maps and language: Lord of the rings encodes the spatial structure of middle earth. *Cognitive Science*, 36(8):1556–1569.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. CharBERT: 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.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Clara Meister, Wojciech Stokowiec, Tiago Pimentel, Lei Yu, Laura Rimell, and Adhiguna Kuncoro. 2023. A natural bias for language generation models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 243–255, Toronto, Canada. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

William Merrill, Ashish Sabharwal, and Noah A. Smith. 2022. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856.

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

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

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. Interspeech 2010*, pages 1045–1048.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1992–1997, Austin, Texas. Association for Computational Linguistics.

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

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

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pages 246–252. PMLR. Reissued by PMLR on 30 March 2021.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*.

Putra Fissabil Muhammad, Retno Kusumaningrum, and Adi Wibowo. 2021. Sentiment analysis using word2vec and long short-term memory (lstm) for indonesian hotel reviews. *Procedia Computer Science*, 179:728–735. 5th International Conference on Computer Science and Computational Intelligence 2020.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

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

Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. 2023. Efficient transformers with dynamic token pooling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6403–6417, Toronto, Canada. Association for Computational Linguistics.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. 2024. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Forty-first International Conference on Machine Learning*.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1–38.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. 2024. Transformers are multi-state rnns.

Laurel Orr, Megan Leszczynski, Simran Arora, Sen Wu, Neel Guha, Xiao Ling, and Christopher Re. 2020. Bootleg: Chasing the tail with self-supervised named entity disambiguation.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures. In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*, Cardiff, United Kingdom. Leibniz-Institut für Deutsche Sprache.

John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. 2008. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Rrubaa Panchendrarajan and Aravindh Amaresan. 2018. Bidirectional LSTM-CRF for named entity recognition. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong. Association for Computational Linguistics.

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.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

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

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems*, volume 36, pages 36963–36990. Curran Associates, Inc.

Jackson Petty, Sjoerd van Steenkiste, Ishita Dasgupta, Fei Sha, Dan Garrette, and Tal Linzen. 2023. The impact of depth and width on transformer language model generalization.

Ofir Press, Noah Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Giovanni Puccetti, Anna Rogers, Aleksandr Drozd, and Felice Dell'Orletta. 2022. Outlier dimensions that disrupt transformers are driven by frequency. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1286–1304, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

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 Rajaee 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 Rajaee and Mohammad Taher Pilehvar. 2022. An isotropy analysis in the multilingual BERT embedding space. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1309–1316, Dublin, Ireland. Association for Computational Linguistics.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Matthias C. Rillig, Marlene Ågerstrand, Mohan Bi, Kenneth A. Gould, and Uli Sauerland. 2023. Risks and benefits of large language models for the environment. *Environmental Science & Technology*, 57(9):3464–3466. PMID: 36821477.

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.

David E. Rumelhart and James L. McClelland. 1987. *Learning Internal Representations by Error Propagation*, pages 318–362.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How good is your tokenizer? on the monolingual performance of multilingual language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the*

11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 3118–3135, Online. Association for Computational Linguistics.

Jonne Saleva and Constantine Lignos. 2021. The effectiveness of morphology-aware segmentation in low-resource neural machine translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 164–174, Online. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*.

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.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.

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

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need.

Luohe Shi, Hongyi Zhang, Yao Yao, Zuchao Li, and Hai Zhao. 2024. Keep the cost down: A review on methods to optimize llm' s kv-cache consumption.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Oleh Shliazhko, Alena Fenogenova, Maria Tikhonova, Anastasia Kozlova, Vladislav Mikhailov, and Tatiana Shavrina. 2024a. mGPT: Few-Shot Learners Go Multilingual. *Transactions of the Association for Computational Linguistics*, 12:58–79.

Oleh Shliazhko, Alena Fenogenova, Maria Tikhonova, Anastasia Kozlova, Vladislav Mikhailov, and Tatiana Shavrina. 2024b. mgpt: Few-shot learners go multilingual. *Transactions of the Association for Computational Linguistics*, 12:58–79.

Karen Sparck Jones. 1988. *A statistical interpretation of term specificity and its application in retrieval*, page 132–142. Taylor Graham Publishing, GBR.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval.

Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. 2022a. A contrastive framework for neural text generation.

Yixuan Su, Fangyu Liu, Zaiqiao Meng, Tian Lan, Lei Shu, Ehsan Shareghi, and Nigel Collier. 2022b. TaCL: Improving BERT pre-training with token-aware contrastive learning. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2497–2507, Seattle, United States. Association for Computational Linguistics.

Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. 2020. Contrastive distillation on intermediate representations for language model compression. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 498–508, Online. Association for Computational Linguistics.

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

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 Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Pier Giuseppe Sessa, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology.

Shiro Terashima, Kazuya Takeda, and Fumitada Itakura. 2003. A linear space representation of language probability through svd of n-gram matrix. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 86(8):61–70.

Evgeniia Tokarchuk and Vlad Niculae. 2022. On target representation in continuous-output neural machine translation. In *Proceedings of the 7th Workshop on Representation Learning for NLP*, pages 227–235, Dublin, Ireland. Association for Computational Linguistics.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. 2021. Training data-efficient image transformers and distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019a. Representation learning with contrastive predictive coding.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019b. Representation learning with contrastive predictive coding.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Lingxiao Wang, Jing Huang, Kevin Huang, Ziniu Hu, Guangtao Wang, and Quanquan Gu. 2020a. Improving neural language generation with spectrum control. In *International Conference on Learning Representations*.

Pei-Hsin Wang, Sheng-Iou Hsieh, Shieh-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juan. 2020b. Contextual temperature for language modeling.

Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. 2020c. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Warren Weaver. 1952. Translation. In *Proceedings of the Conference on Mechanical Translation*, Massachusetts Institute of Technology.

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. 2020. Visual transformers: Token-based image representation and processing for computer vision.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. 2021. Cvt: Introducing convolutions to vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22–31.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th international conference on world wide web*, pages 1391–1399.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Lizhong Xiao, Guangzhong Wang, and Yang Zuo. 2018. Research on patent text classification based on word2vec and lstm. In *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, volume 01, pages 71–74.

Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. 2021. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Advances in Neural Information Processing Systems*.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14138–14148.

Lingling Xu, Haoran Xie, Zongxi Li, Fu Lee Wang, Weiming Wang, and Qing Li. 2023. Contrastive learning models for sentence representations. *ACM Trans. Intell. Syst. Technol.*, 14(4).

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

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022b. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5065–5075, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2018. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*.

LILI YU, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 2023. MEGABYTE: Predicting million-byte sequences with multiscale transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Sangwon Yu, Jongyoon Song, Heeseung Kim, Seongmin Lee, Woo-Jong Ryu, and Sungroh Yoon. 2022. Rare tokens degenerate all tokens: Improving neural text generation via adaptive gradient gating for rare token embeddings. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29–45, Dublin, Ireland. Association for Computational Linguistics.

Jerrold H Zar. 2005. Spearman rank correlation. *Encyclopedia of Biostatistics*, 7.

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.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 34661–34710. Curran Associates, Inc.

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

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

George Kingsley Zipf. 1935. *The psycho-biology of language : an introduction to dynamic philology*. The psycho-biology of language: an introduction to dynamic philology. Houghton Mifflin, Oxford, England.

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.