Universitat de Barcelona

Fundamental Principles of Data Science Master's Thesis

Attention mechanisms in transformers:
A new formula with mathematical foundations and enhanced interpretability

Author:
Eddie Conti

Supervisor:
Prof. Arturo Vieiro Yanes
Prof. Oriol Pujol Vila

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

July 9, 2024

Acknowledgements

Thanks to professors Arturo Vieiro Yanes and Oriol Pujol Vila for the help during the thesis.

Contents

1 Introduction and overview

Large Language Models (LLMs) are AI systems capable of understanding and generating human language by processing vast amounts of text data. In recent years, specifically from 2017, the use of LLMs significantly increased thanks to the introduction of the Transformer architecture. This particular structure is characterized by several aspects as:

- Context-depending mechanism;
- Attention mechanism;
- Multitask-training using huge amounts of text;
- Parallel processing.

Even though the deep architecture is composed of several components, it can be summarized as a tool to find a proper embedding of the input and the output. The representation of text is the main aspect of LLMs. Formally, given a vocabulary $\mathcal{V}$ of size $|\mathcal{V}| = l$, an embedding is a function

$$E\colon \mathcal{V} \to \mathbb{R}^d$$

so that a word $w \in \mathcal{V}$ is represented by a row vector with d entries. therefore, a vocabulary $\mathcal{V}$ will be represented by a matrix $M \in \mathcal{M}_{\mathbb{R}}(l, d)$, where $\mathcal{M}_{\mathbb{R}}(l, d)$ is the space of real matrices of dimension $l \times d$. However, finding the proper function E is not straightforward. The easiest case would be to to represent each word $v_i \in \mathcal{V}$ with the unit vector $e_i = (0, \dots, 0, 1, 0, \dots, 0)$ where 1 is in the $i-$th position. Unfortunately, this embedding is not suitable for understanding and replicating human language. Indeed, if we perform $v_i \cdot v_j = 0$, namely every word are represented by orthogonal, and so independent, vectors. When representing the text with vectors we aim at capturing the similarity of the word based on the context: in this regards we expect that words as 'tree', 'leaf' are similar while 'train' and 'rock' are dissimilar. However, capturing similarity is not the only desirable goal. Language is often self-referential, in the sense that we have words that refer to or replace other words, even though their meanings are not related. To illustrate, consider the following example:

Luca is studying in Spain. He is very happy.

In the previous sentence, the word "He" is replacing "Luca", and we need to capture this aspect of language. Furthermore, to create an interactive tool, it is essential to capture this self-referential aspect at a high level. Everything is about embedding, or in other terms, finding a good way to represent your input so that your architecture can process and elaborate it.
The Transformer architecture meets these requirements and is able to produce embeddings suitable for a variety of tasks such as translation, text summarization, natural language generation, and more. Nevertheless, the number of parameters involved is extremely large and expensive. Moreover, the core of the Transformer, the Self-Attention mechanism, lacks a proper mathematical treatment. Given these limitations, this thesis delves into the aspects of transformers in mathematical terms.

In the first Chapter, we analyze the architecture, explaining each component in detail. This is a key aspect to understand at a deep level the architecture and the flow of information as well as the complexity of it. It is clear from the mathematical formulation in equations (3) that the computational cost of computing the attention weights is quadratic. This has led the scientific community to search for equivalent or similar formulations while reducing computational complexity.

In Chapter 2 we present already existing a posteriori approaches that reduce the complexity of the Transformer architecture. Specifically, we explore Sparse Attention, Linear Attention, and matrix factorization approaches to linearly scale the complexity. The main idea is to approximate or reduce the attention weight matrix. However, in this thesis we wanted to explore through experiments if there exists a structure in the attention weight matrix and exploit it to define an alternative formulation. Inspired by paper [12] this lead to the next and conclusive chapter.

In the third Chapter, we delve into the mathematical and geometrical aspects of the attention scores formula. In this section we explore the experimental session we carried out: firstly we used and adapted the code of the transformer architecture for translation from english to italian and then we visualize the attention weight matrices for several settings. Here, we derive a novel formula (cfr. (13)) that encapsulates the attention mechanism at a high level. Essentially, we guide queries and keys to interact in a specific manner, encoding the distinct roles of attention heads and directing values on where to seek context. This not only simplifies the architecture but also enhances the explainability of the underlying processes.
In mathematical terms, we can think of this formula as projecting the attention scores matrix, say H, onto the space of band matrices with fixed bandwidth. This convex subspace is clearly finite dimensional and therefore closed. As a consequence, the projection on this space is well-posed and unique. However, at the price of loosing the uniqueness of the projection (i.e. the best approximation for H), we defined a new space which is made by band matrices + error sparse matrices. We prove that this is a compact subspace which guarantees the existence of a matrix that best approximate H. This space allow us to find a better solutions and matrices with more elasticity since we introduce randomness.
We conclude the thesis validating the new formula, namely calculating how well the new formula for attention scores approximates the original one (see Table 16 and Table 17). We coded different functions to generate the desired matrices and we checked the approximation against the original.
Additionally, we explore the impact of different parameters such as w (context windows) and num-pos (number of relevant words in a sentence). These analyses provide deeper insights into how languages are processed and translated, revealing nuances in the roles of context and word relevance.
In conclusion, our new formula not only simplifies the attention mechanism but also offers enhanced interpretability and approximate properly the original attention weight matrix. This contribution lays the groundwork for further research on the role of context windows and the significance of word relevance, that could enrich our understanding of languages in translation.

All the experiments, code, images and papers can be found in the following GitHub repository: https://github.com/EddieConti/Master-Thesis-UB

1.1 Seq2seq

Commonly, LLMs are used for text generation and encoder-decoder is the standard modeling paradigm for sequence-to-sequence tasks. The encoder reads source sequence and produces its representation; while the decoder uses source representation from the encoder to generate the target sequence. Given an input sequence $x = (x_1, \dots, x_n) \in \mathcal{V}^n$ we aim at finding the target sentence $y^* = (y_1^*, \dots, y_{n'}^*)$, for certain $n, n' \in \mathbb{N}$, that is the most probable given the input. The output y^* is an element of a final set, for example, another vocabulary $\mathcal{V}'$ in case of translation task as the one we are going to consider. Formally, the target sequence that maximizes

the conditional probability $p(y|x)$:

$$y^* = \underset{y}{\operatorname{argmax}}\, p(y|x).$$

In the case of language models, any possible output $y = (y_1, ..., y_{n'})$ depends on the previous generated outputs. Then, the probability of a sequence of generated outputs can be decomposed as

$$p(y_1, \ldots, y_{n'}) = p(y_1) \cdot p(y_2|y_1) \ldots p(y_{n'}|y_1, \ldots, y_{n'-1}) = \prod_{t=1}^{n'} p(y_t|y_{<t})$$

and so taking into account the input and the parameters the aim is to determine the most probable output

$$y^* = \underset{y}{\operatorname{argmax}} \prod_{t=1}^{n'} p(y_t|y_{<t}, x).$$

Neural seq2seq models are trained to predict probability distributions of the next token given previous context (source and previous target tokens). At the timestep t a model predicts a probability distribution

$$p^{(t)} = p(\,\cdot\,|y_1, \ldots, y_{t-1}, x_1, \ldots, x_n).$$

The target at this step is $p^* = \text{one-hot}(y_t)$, i.e., we want a model to assign probability 1 to the correct token, y_t, and zero to the rest. The standard loss function is the cross-entropy loss:

$$Loss(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{l} p_i^* \log(p_i)$$

but since only one p_i^* (for the correct token y_t) is non-zero we get

$$Loss(p^*, p) = \log(p_{y_t}) = -\log(p(y_t|y_{<t}, x)).$$

Hence, at each step we maximize the probability a model assigns to the correct token. Now that we know how the model is trained, we focus on how to generate the sentence, i.e, how to find y^* that satisfies

$$y^* = \underset{y}{\operatorname{argmax}} \prod_{t=1}^{n'} p(y_t|y_{<t}, x).$$

The total amount of feasible solution is $|\mathcal{V}'|^{n'}$ which is typically too large, hence usually there are two approaches that can be used to find an approximate solution: Greedy Decoding and Beam Search. The former, consists of generating at each step a token with the highest probability. Even though it is a good baseline, mathematically

$$\underset{y}{\operatorname{argmax}} \prod_{t=1}^{n'} p(y_t|y_{<t}, x) \neq \prod_{t=1}^{n'} \underset{y_t}{\operatorname{argmax}}\, p(y_t|y_{<t}, x).$$

As a consequence, Beam Search, tries to tackle this issues by keeping track of several most probably hypotheses. At each step we continue each of the current hypotheses and pick top-N of them.

1.1.1 Capturing the context: Attention mechanism

So far, we have introduced the framework of LLMs, focusing on seq2seq models. The simplest architecture consists of two Recurrent Neural Networks (RNNs), e.g. two Long Short Time Memory (LSTMs) networks: an encoder RNN reads the source sentence, and its final state is used as the initial state of the decoder RNN. The hope is that the final encoder state "encodes" all information about the source, allowing the decoder to generate the target sentence based on this vector. However, the constraint of condensing all information into a single vector fails to capture the complexity of language and often proves insufficient for synthesizing the input. This motivates the introduction of the concept of attention (see paper [2]).
The attention mechanism is the foundation of the transformer architecture, which we will be discussed in the next section. The basic idea is to allow the model to weight the importance of each word within a sequence according to its context. In this way, the model can capture long-range word relationships without relying on rigid sequential structures. In simple terms, the attention mechanism looks at an input sequence and decides, at each step, which other parts of the sequence are important.

1.2 The transformer architecture

A transformer is an architecture presented for the first time in the 2017 paper *Attention Is All You Need* [1], for transforming one sequence into another one with the help of two parts: Encoder and Decoder. A basic sketch is represented in Figure 1.

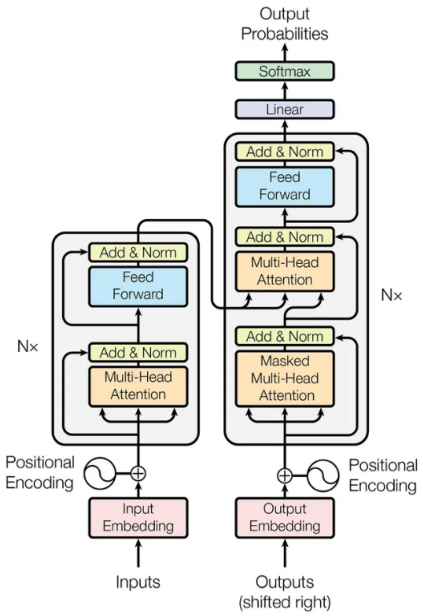


Figure 1: The transformer architecture. Image from [1].

Let us now explore the architecture in Figure 1. As we can see it is an Encoder-Decoder architecture: the former consists of encoding layers that process the input tokens iteratively one layer after another, while the latter consists of decoding layers that iteratively process the encoder's output as well as the decoder output's tokens so far.
The sequence is presented as a list of tokens x_i for $i = 1, \dots, n$, usually as one-hot encodings. We denote with X the matrix representing the input sequence of dimension $n \times l$[1]. Since the vocabulary size $|\mathcal{V}| = l$ is very large we reduce the dimensionality

[1]From an algorithmic point of view, we force the input to have always the same length. This is achieved by using padding, which consists of adding special tokens (usually called '[PAD]') at

by computing

$$X \cdot W^E, \quad W^E \in \mathbb{R}^{l \times d} \text{ and } d = 512,$$

where the matrix W^E is trainable. Now we have a matrix that is representative of our input sequence of size (n, d). Lastly, following reference [1], we multiply $X \cdot W^E$ by $\sqrt{512}$.
From now on we will omit that we actually work with tensors because the input is not a single sentence but rather a set of sentences of dimension $batch - size$.

1.2.1 Positional Encoding

As we already pointed out, all tokens are presented to the Transformer simultaneously. However, this results in the loss of the original order of tokens in the input sequence. The purpose of positional encoding is to encapsulate the relative positions of tokens within a target sequence. Before presenting the original function, let us delve into our intuition as to why positional encoding is necessary.
The general idea is to allow the model to differentiate between words with similar meanings but different positions in the sentence. For example, if we consider the following sentences:

> Luca did not won the football tournament and he was happy;
>
> Luca won the football tournament and he was not happy.

Without any additional information about the positions of the words, the model might treat the sentences similarly, although the meaning is different.
A first approach could be to add the index of the of the token to its representation: $x_i \mapsto x_i \cdot W^E + i$. However, as we will see in the next pages we are going to perform matrix operations and exponentiation, and $x_i \cdot W^E + i$ grows as the length of the sequence grows. This is dangerous as it can lead to exploding gradients.
Hence, one possibility would be to normalize index i so it ranges values between $[0, 1]$. Unfortunately, this strategy would still confuse the model if the input text length varies, since the result would contain different positional embeddings for the same positions in different input sequences. The solution presented in paper [1] involves using a sine and cosine function to create a d dimensional vector for each position in the sentence. The positional encoding is defined as a function $f\colon \mathbb{N} \to \mathbb{R}^d$ where, given $t \in \mathbb{N}$, the $i-$th entry is defined as

$$f(t)^{(i)} = \begin{cases} \sin(\omega_k \cdot t) & \text{if } i = 2k, \\ \cos(\omega_k \cdot t) & \text{if } i = 2k+1 \end{cases} \tag{1}$$

with

$$\omega_k = \frac{1}{N^{2k/d}}.$$

Alternatively, it can be expressed as a complex valued function as

$$f(t)^{(i)} = e^{it/\omega_k} \quad k = 0, \dots, d/2 - 1.$$

In paper [1] $N = 10000$, but in general is a free parameter that should be significantly larger than the biggest k that in our notation is $d/2 - 1$. The main idea behind this function is that it allows one to perform shifts as linear transformations and this is useful to express the relative position. For the sake of simplicity, let us fix $i = 2k$

$$\begin{aligned} f(t+s)^{(i)} &= \sin(\omega_k \cdot (t+s)) = \sin(\omega_k \cdot t + \omega_k \cdot s) \\ &= \sin(\omega_k \cdot t)\cos(\omega_k \cdot s) + \sin(\omega_k \cdot s)\cos(\omega_k \cdot t) \\ &= a \sin(\omega_k \cdot t) + b \cos(\omega_k \cdot t). \end{aligned}$$

the end of shorter sentences until the desired length is reached. In order to handle these padding tokens correctly during processing, a mask, that will be described later, is used to tell the model which positions are padding and should not be taken into account in the attention calculations.

Similarly, for $i = 2k + 1$

$$\begin{aligned} f(t+s)^{(i)} &= \cos(\omega_k \cdot (t+s)) = \cos(\omega_k \cdot t + \omega_k \cdot s) \\ &= \cos(\omega_k \cdot t)\cos(\omega_k \cdot s) - \sin(\omega_k \cdot t)\sin(\omega_k \cdot s) \\ &= a' \cos(\omega_k \cdot t) + b' \sin(\omega_k \cdot t), \end{aligned}$$

hence for a fixed offset s we can express the $(t+s)-$th position a linear combination of (known functions of) the $t-$th position. As stated by the authors they hypothesise that this property would allow the model to easily learn to attend relative positions. Furthermore, there is another relevant reason for this specific function: the output for a fixed t, i.e. for a fixed position, is a d dimensional vector

$$(\sin(\omega_1 \cdot t), \cos(\omega_2 \cdot t), \ldots, \sin(\omega_d \cdot t))$$

where $t \in \mathbb{N}$ influences the oscillation of the sine/cosine function. As a consequence when changing the parameter t we obtain a different vector that is representative of that position of the word. The positional embedding is a matrix $P \in \mathbb{R}^{N \times d}$ and it is then added to $X \cdot W^E$ in a compatible way

$$X \cdot W^E + P[:n, :] \in \mathbb{R}^{n \times d}, \tag{2}$$

where $P[:n, :]$ refers to the matrix formed by the first n rows of P. A dropout of 0.1 is then applied to the previous matrix.

1.2.2 Transformer Block

In this pages we are going firstly to introduce in mathematical formulas all the equations describing a transformer block and the final output of the encoder, then we are going to explain each equation in details and the meaning of the different parameters.

The crucial step of the architecture is the so called the transformer block consisting of several parts as shown in Figure 1. For simplicity, let us denote again by $X \in \mathbb{R}^{n \times d}$ the position-aware word embedding as given in (2). We can think of the transformer block as a function depending on some parameters θ

$$f_\theta \colon \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}.$$

In this scenario, the name *transformer* captures that function f_θ is just transforming the input as the domain and codomain are the same. The block can be described in mathematical terms as it follows: Given X, then $f_\theta(X) = Z$, $Z = (z_1, \ldots, z_n)$, $z_i \in \mathbb{R}^d$, where:

$$\begin{aligned} Q^{(h)}(x_i) = W_{h,q}^T(x_i),\ K^{(h)}(x_i) = W_{h,k}^T(x_i),\ V^{(h)}(x_i) = W_{h,v}^T(x_i) \quad & W_{h,q}, W_{h,k}, W_{h,v} \in \mathbb{R}^{d \times k}, \\ \alpha_{i,j}^{(h)} = \operatorname{softmax}_j\Big(\frac{Q^{(h)}(x_i) \cdot (K^{(h)}(x_j))^T}{\sqrt{k}}\Big), & \\ u_i' = \sum_{h=1}^{H} W_{c,h}^T \sum_{j=1}^{n} \alpha_{i,j}^{(h)} V^{(h)}(x_j), \qquad & W_{c,h} \in \mathbb{R}^{k \times d}, \\ u_i = \operatorname{LayerNorm}(x_i + u_i'; \gamma_1, \beta_1), \qquad & \gamma_1, \beta_1 \in \mathbb{R}^d, \\ z_i' = W_2^T \operatorname{ReLU}(W_1^T u_i + b_1) + b_2, \qquad & W_1 \in \mathbb{R}^{d \times m}, W_2 \in \mathbb{R}^{m \times d}, b_1, b_2 \in \mathbb{R} \\ z_i = \operatorname{LayerNorm}(u_i + z_i'; \gamma_2, \beta_2), \qquad & \gamma_1, \beta_1 \in \mathbb{R}^d. \end{aligned} \tag{3}$$

The LayerNorm function is defined as

$$\operatorname{LayerNorm}(z; \gamma, \beta) = \gamma \frac{(z - \mu_z)}{\sigma_z + \epsilon} + \beta, \tag{4}$$

where ϵ is a fixed number as 10^{-6} to prevent division by 0, and

$$\mu_z = \frac{1}{k}\sum_{i=1}^{k} z_i, \qquad \sigma_z = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(z_i - \mu_z)^2}.$$

Thus, given $x \in \mathbb{R}^{n\times d}$ we produce $z \in \mathbb{R}^{n\times d}$ such that $Z = f_\theta(X)$ where f describes the transformer block and the parameter θ consists of entries of weight matrices W along with the parameter of LayerNorm and RELU. In the general setting the final output is a stack of transformer blocks and can be computed as the composition

$$f_{\theta_L} \circ \ldots \circ f_{\theta_1}(X) \in \mathbb{R}^{n\times d}.$$

In the paper [1] the hyperparameters are $d = 512$, $k = 64$, $m = 2048$, $H = 8$ and $L = 6$. Where d is the dimension of the model, k is the dimension of each head (we are going to explain in a moment the meaning of head), m is used for the dimension in the RELU matrices, $H = d/k$ is the number of heads and L is the number of layers.

Let us now explore each step involved in the transformer block. The first equation is the formalization of the so called query, key and value vectors. First of all, three matrices $W_q, W_k, W_v \in \mathbb{R}^{d\times d}$ are randomly initialized: these matrices are then split to obtain sub-matrices of dimension (n, k) as it follows:

$$\begin{aligned} W_q &\to W_{1,q} = W_q[:, :k], \ldots, W_{H,q} = W_q[:, d-k:d], \\ W_k &\to W_{1,k} = W_k[:, :k], \ldots, W_{H,k} = W_k[:, d-k:d], \\ W_v &\to W_{1,v} = W_v[:, :k], \ldots, W_{H,v} = W_k[:, d-k:d]. \end{aligned} \tag{5}$$

Where $A[:, a:b]$ refers to the matrix formed by the columns from a to b of A. The case when $a = 0$ is denoted as $A[:, :b]$.
After reducing the dimensionality from (d, d) to (d, k), we compute query, key, value vectors as

$$Q^{(h)}(x_i) = W_{h,q}^T(x_i), \quad K^{(h)}(x_i) = W_{h,k}^T(x_i), \quad V^{(h)}(x_i) = W_{h,v}^T(x_i). \tag{6}$$

We denote d/k with H i.e. the number of heads. The geometric and intuitive meaning of these matrices can be found in chapter 3. The next expression describes the self-attention mechanism

$$Attention(Q^{(h)}, K^{(h)}, V^{(h)}) = Softmax\Big(\frac{Q^{(h)}(K^{(h)})^T}{\sqrt{k}}\Big)V^{(h)}, \tag{7}$$

and we usually refer to them as *attention values*, while the *attention weights* are

$$Softmax\Big(\frac{Q^{(h)}(K^{(h)})^T}{\sqrt{k}}\Big) \tag{8}$$

We compute the attention for each head i.e. each splitting of W_q, W_k, W_v so that

$$head_i = Attention(Q^{(i)}, K^{(i)}, V^{(i)}), \quad i = 1, \ldots, H.$$

It is essential to observe that each $W_{i,q}, W_{i,k}, W_{i,v}$ is of dimension (n, k) so that is "looking" at the whole sentence but in different positions, i.e. is looking at different attributes of the embedded input. The next equation is the output of the so called Multi-head attention. If Q, K, V refers to the concatenation of $Q^{(h)}, K^{(h)}, V^{(h)}$ then the third equation can be expressed as

$$\text{MultiHead}(Q, K, V) = Concat(head_1, \ldots, head_h)W_c \tag{9}$$

where W_c is a matrix of dimension $(H \times k, d) = (d, d)$ and so the output of Multi-Head attention lies in $\mathbb{R}^{n\times d}$. The role of the matrix W_c is to extract the most valuable information from each head by weighting their outputs and this is why in linear expression can be split into $W_{c,h}$ for $h = 1, \ldots, H$.

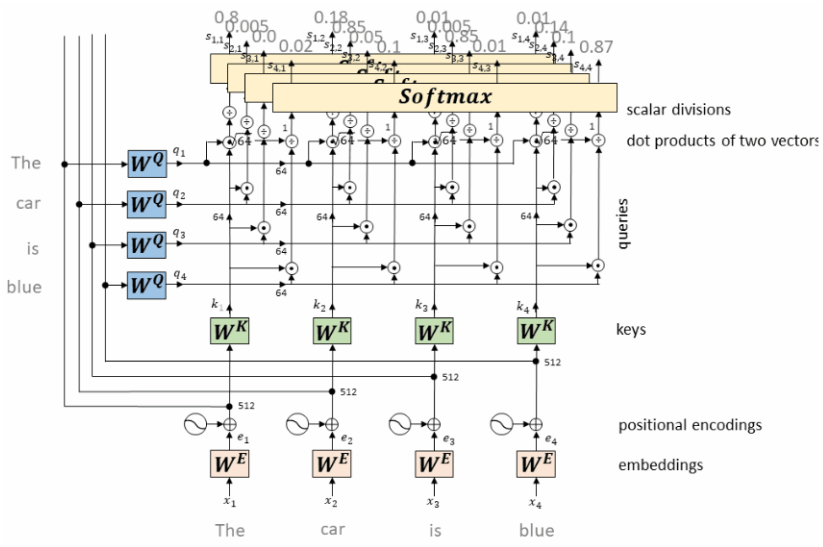


Figure 2: Embeddings, query and key vectors. This figure, as well as Figure 3 and 4 are taken from https://towardsdatascience.com/drawing-the-transformer-network-from-scratch-part-1-9269ed9a2c5e

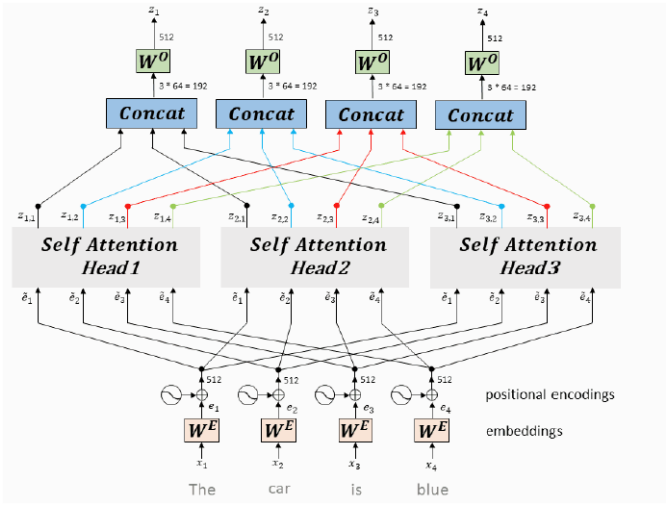


Figure 3: Multi-Head Attention. The matrix W^O is what we denoted with W_c

1.2.3 Motivation for scaling the dot products

Before proceeding with the transformer block, it is worthy to motivate the division by $\sqrt{k}$ in (8) in computing the attention weights $\alpha_{i,j}^{(h)}$. This factor is applied before the softmax because it leads to more stable gradients. To understand this, we recall that the softmax function is a vector function

$$\begin{aligned} &Softmax\colon \mathbb{R}^n \to \mathbb{R}^n \\ &z_i \mapsto s_i := \frac{e^{z_i}}{\sum_j e^{z_j}}. \end{aligned} \tag{10}$$

Now, if we compute the partial derivatives for $k \neq i$

$$\begin{aligned} \frac{\partial s_i}{\partial z_i} &= \frac{\partial}{\partial z_i}\frac{e^{z_i}}{\sum_j e^{z_j}} = \frac{e^{z_i}}{\sum_j e^{z_j}} + e^{z_i}\frac{\partial}{\partial z_i}\frac{1}{\sum_j e^{z_j}} = \frac{e^{z_i}}{\sum_j e^{z_j}} - \left(\frac{e^{z_i}}{\sum_j e^{z_j}}\right)^2 = s_i \cdot (1 - s_i), \\ \frac{\partial s_i}{\partial z_k} &= \frac{\partial}{\partial z_k}\frac{e^{z_i}}{\sum_j e^{z_j}} = e^{z_i}\partial z_k \frac{1}{\sum_j e^{z_j}} = \frac{-e^{z_i}e^{z_k}}{\left(\sum_j e^{z_j}\right)^2} = -s_i \cdot s_k. \end{aligned}$$

Therefore, the Jacobian of (10) can be expressed as

$$J_s = \begin{pmatrix} s_1 \cdot (1 - s_1) & -s_1 \cdot s_2 & \cdots & -s_1 \cdot s_n \\ -s_2 \cdot s_1 & -s_2 \cdot (1 - s_2) & \cdots & -s_2 \cdot s_n \\ \vdots & \vdots & \ddots & \vdots \\ -s_n \cdot s_1 & -s_n \cdot s_2 & \cdots & -s_n \cdot (1 - s_n). \end{pmatrix}$$

It is immediate to observe that the Jacobian becomes zero if occurs one among

$$s = (1, 0, \ldots, 0),\ s = (0, 1, 0, \ldots, 0), \ldots,\ s = (0, \ldots, 0, 1). \tag{11}$$

However, the Softmax function is not scale invariant, which means that with increasing scale, the function assigns a value close to 1 to the largest input value and 0 to all other values. To see this, if we consider the vector $z = (1, 2, 0.5, -1)$ the softmax will be $(0.2242, \mathbf{0.6095}, 0.1360, 0.0303)$ but if we consider the doubled vector $2z$ then the output is approximately $(0.11396, \mathbf{0.84203}, 0.04192, 0.00209)$.
Therefore, for large inputs the Softmax function is generating outputs which closely resemble the values in (11). To tackle the problem of the vanishing gradient is necessary to rescale the dot products as the larger the dimension d of the key vectors and query vectors, the larger the dot products will tend to be.

1.2.4 Layer Normalization and Feed Forward

At this point the output of the Multi-Head Attention U' is combined with X coming from residual connection. The idea is to still consider the original embedding and to move smoothly through the architecture. Vectors u'_i and x_i are combined using the Layer Normalization function in (4). The output is then

$$U = \gamma_1 \frac{X + U' - \mu_{x,u}}{\sigma_{x,u} + \epsilon} + \beta_1 \in \mathbb{R}^{n \times d}$$

Here we apply to each u_i a Feed Forward network consisting of two linear transformations with a ReLU activation in between. The dimensionality of input and output is d, and the inner-layer has dimensionality $m = 2048$ described by:

$$z'_i = \max(0; u_i W_1 + b_1) W_2 + b_2$$

where $W_1 \in \mathbb{R}^{d \times m}$, $W_2 \in \mathbb{R}^{m \times d}$ and $b_1, b_2 \in \mathbb{R}$. Finally, we again employ a residual connection and a normalization around the fully connected FFN analogously to as described above and produce the final embedding z_i for each x_i.
All of these operations can be sintetized using the function f_θ defined in (3) such that

$$f_\theta(X) = Z.$$

However, this is just a transformer block, but the encoder is made by L (number of layers) transformer blocks stacked, so that the final context aware embedding is

$$f_{\theta_L} \circ \ldots \circ f_{\theta_1}(X) \in \mathbb{R}^{n \times d}.$$

We summarize the architecture in the following picture.
The general motivation for such a deep architecture is that, in order to capture the relationships between words, a crucial aspect for translating and generating language, it is essential to analyze the input text from many different perspectives. Similar to what happens with images in convolutional neural networks, each encoder captures a depth-dependent connection among the input. Language is extremely complex and heterogeneous; therefore, we need a complex structure capable of apprehending all its facets.

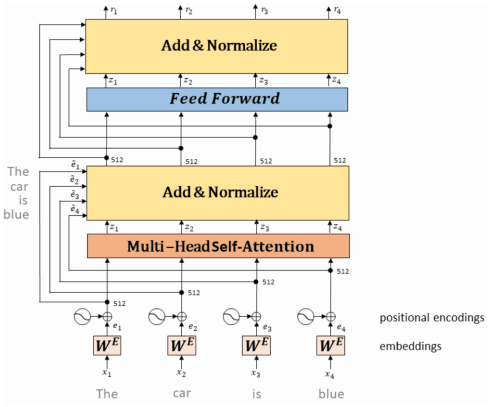


Figure 4: Graphical visualization of the transformer block we have described so far. The encoder is just a concatenation of these blocks.

1.2.5 The decoder

The decoder is responsible for sequentially generating output based on the information processed by the encoder. Structurally, it is similar to the encoder, although particular attention must be given to some aspects.
Firstly, the decoder takes positional information and embeddings of the output sequence as its input. During self-attention, the decoder employs a mask to ensure that each position can only be influenced by preceding positions in the sequence. This mask is crucial to ensure that the model generates sequential output correctly without "looking ahead" during generation. The masked multi-head attention is similar to what we described before; however, we apply a method to prevent computing attention scores for future words.

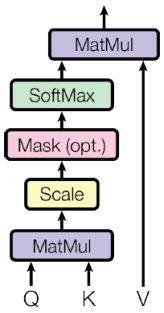


Figure 5: Details on how attention coefficients are computed. Image from paper [1]

As Figure 5 points out the mask is applied after the scaling operation. Let A be the matrix of scaled values $a_{i,j}$ and M the masking matrix defined, in a limit sense, as

$$\begin{pmatrix} 0 & -\infty & -\infty & \cdots & -\infty \\ 0 & 0 & -\infty & \cdots & -\infty \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -\infty \\ 0 & 0 & \cdots & \cdots & 0 \end{pmatrix}.$$

If we add matrix M to matrix S we obtain a matrix of masked scores whose elements are s_{ij} if $i \geq j$ and $-\infty$ otherwise. At this point if we apply the softmax to each row vector (the values for each token) if we have $-\infty$, i.e. a large negative number, then the output of the softmax will be (approximately) 0. As a consequence, we end up

with a matrix of the form

$$\begin{pmatrix} a'_{1,1} & 0 & 0 & 0 & 0 \\ a'_{2,1} & a'_{2,2} & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a'_{n-1,1} & a'_{n-1,2} & \cdots & a'_{n-1,n-1} & 0 \\ a'_{n,1} & a'_{n,2} & \cdots & \cdots & a'_{n,n} \end{pmatrix}.$$

The attention coefficients are computed as

$$\text{MaskedAttention}(Q, K, V) = \text{softmax}\left(M + \frac{QK^{\mathrm{T}}}{\sqrt{k}}\right)V. \tag{12}$$

It is worthy to note that even if in (12) we end up with a full matrix, the $j-$th row, i.e, the attention coefficients for input x_j, are computed as

$$(a'_{j,1}, \ldots, a'_{j,j}, 0, \ldots, 0) \cdot V.$$

In other words, they are linear combinations of $a'_{j,1}, \ldots, a'_{j,j}$ and this encodes exactly that the decoder is only influenced by preceding positions in the sequence. To sum up, the output of the first Multi-Head attention is a masked output vector with information on how the model should attend on the decoder's input.
In the second Multi-Head attention layer we use cross-attention so that the decoder interacts with the output of the encoder Z. If we denote again with X the input of the decoder, we apply the attention formula as

$$\text{softmax}\left(\frac{XZ^T}{\sqrt{k}}\right)Z$$

so that the query is from the decoder while key and values are from the encoder. This is consistent with what we said in the previous section. The query dialogue with the key from the encoder in order to capture the similarity between what will our output and our original input. Then, to produce the output we apply the values from encoder that are trained to produce words given the similarity. Here, the architecture proceeds similarly to the encoder's one: it processes the inputs with feed forward layers and normalization. Residual connections are applied to move smoothly through the architecture for the back propagation.
Finally, the output let us say $Y \in \mathbb{R}^{n' \times d}$ where n' is the length of the output sequence, is approximated by the closest word in the target vocabulary of size l' (for instance, in the case of text summarization or question-answer the vocabulary is the same, but if we are translating clearly differs from the input one) tasks by performing

$$Y \cdot P, \quad P \in \mathbb{R}^{d \times l'}.$$

Here, a softmax will produce probability scores between 0 and 1. We take the index of the highest probability score, and that equals to our predicted word. The decoder then takes the output, add's it to the list of decoder inputs, and continues decoding again until a <end> token is predicted. Therefore, the decoding steps produces a word that is the most probable according to the input and its context and the previous word generated: in this way we gained the ability to produce meaningful sentences that are related to the input sentence.

Now that we have detailed the architecture, in the next chapter, we will explore its limitations and present the efforts from the scientific community to address these challenges.

2 The amount of parameters in LLM

In this section, we present alternative formulations of the Transformer architecture. We will explore sparse attention, linear attention, and matrix factorization methods, each of which reduces the complexity of the Transformer architecture. These approaches serve as a foundation for our formulation, which we present in Chapter 3. with the difference that we aim at exploring at a deeper level the attention matrix and exploit its structure.

Transformers are the primary architecture employed for LLMs. Despite their notable capabilities in understanding and generating language, such as GPT, a crucial aspect of these models lies in their substantial number of parameters, particularly when applied to the English language. The desire to capture the richness of linguistic context has led to architectures of considerable size, comprised of millions or even billions of parameters. As explained in Chapter 1, in order to comprehend language, it is essential to use a huge variety of matrices (i.e., word representations) and long-distance word relationships.
The largest models can have a context window size of up to 128000 tokens as in the case of GPT-4o. Previous versions as GPT-3.5 has a context window ranging from $4,000$ to $16,000$ tokens, while the legacy GPT-3 has a context window of $2,000$ tokens. This is translated in a huge number of parameters involved in the model, see Table 1.
Additionally, the significant economic costs, as well as the environmental impact, in terms of computational resources and infrastructure must be taken into account.

Name	Release Date	Number of parameters
GPT-1	June 2018	117 million
BERT	October 2018	340 million
GPT-2	February 2019	1.5 billion
GPT-3	May 2020	175 billion
LaMDA	January 2022	137 billion
Minerva	June 2022	540 billion

Table 1: List of some LLMs with the amount of parameters

As it is clear from Table 1 training transformer-based architectures can be expensive, especially for long inputs.

Within this context, the challenge of balancing the expressive power of advanced language models with the need to optimize computational efficiency emerges.

2.1 Transformers for long sequences: changing the attention

The Transformer architecture suffers from limitations when it comes to processing long sequences. Most of the complexity of the architecture lies in the self-attention mechanism, where we compute the attention coefficients. Each token x_i is related to every other token x_j in order to capture the relationship between words. This means that the resulting complexity is $O(n^2)$, where n is the sequence length. Most Transformer models have a fixed sequence length. For example, the BERT model [6] is limited to 512 tokens. However, the ability to handle a variety of tasks is a desirable aspect. For tasks such as document summarization, DNA processing, or any task that requires processing long sequences, training becomes practically infeasible due to the enormous computational cost.

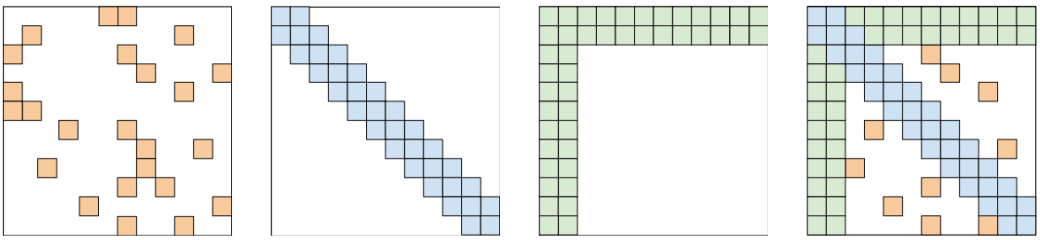
In the next pages we are going to cover different techniques that reduce this quadrat-

ically dependency on sequence length. These approaches try to reduce the quadratic dependency to a linear dependency in n.

2.1.1 Sparse attention

Sparse attention refers to an attention mechanism where the attention of each token is limited to a subset of other tokens. BigBird, a transformer model developed by Google Research (see [7]), implements a sparse attention mechanism that reduces the quadratic dependency to linear. As they proved empirically, the proposed sparse attention can handle sequences of length up to eight times largen that of what was previously possible using similar hardware. As a consequence, BigBird drastically improves performance on various Natural Language Processing (NLP) tasks such as question answering and summarization. More precisely, BigBird is a combination of global attention, local attention and random attention. In particular, BigBird consists of three main parts:

- a set of g global tokens attending to any part of the sequence;
- all tokens attending to a set of w local neighboring tokens;
- all tokens attending to a set of r random tokens.



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Figure 6: White color indicates absence of attention. (a) Random attention with $r = 2$, (b) sliding window attention with $w = 3$, (c) global attention with $g = 2$, (d) the combined BigBird model.

In mathematical terms, let $X = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$ the embedded input sequence as in (2). The generalized attention mechanism is described by a directed graph D whose vertex set is $\{1, \ldots, n\}$. Denote by $N(i)$ the out-neighbors set of node i in D, that is, the set of nodes that the attention mechanism will consider. The generalized attention is defined as

$$\mathrm{ATTN}_D(X)_i = x_i + \sum_{h=1}^{H} \sigma\Big(Q^{(h)}(x_i) K^{(h)}(X_{N(i)})^T\Big) \cdot V^{(h)}(X_{N(i)}), \tag{13}$$

where $Q^{(h)}, K^{(h)} : \mathbb{R}^d \to \mathbb{R}^k$ (k is the dimension of each head) are query and key functions, $V_h : \mathbb{R}^d \to \mathbb{R}^d$ is a value function, σ is a scoring function such as softmax and H denotes the number of heads (see section 1.2.2). Denote $X_{N(i)}$ the matrix formed by only stacking $\{x_j : j \in N(i)\}$ and not all the inputs. It is clear that (13) reduces to (7) if $N(i)$ is full, in the sense that we are considering the relationship among all the input words. The general idea behind BigBird architecture is that most contexts within NLP and computational biology have data which displays a great deal of locality of reference. In this sense, for a fixed token x_i, the tokens x_j for $i - w/2 \leq j \leq i + w/2$ are considered to be the most relevant for a context comprehension. Furthermore, adding random attention in the key function allows to capture

less obvious or less frequent information that may be crucial for understanding the context. Furthermore, introducing randomness in attention can make the architecture more robust to variations in input data (see [7]). Moreover, in contexts where relevant information may be distant or not immediately related, random attention could facilitate capturing broader and non-local relationships. To conclude, the Big Bird architecture involves the use of global attention, i.e, tokens that attend to all tokens in the sequence and to whom all tokens attend to. Concretely, we choose a subset $G \subseteq \{1, \ldots, n\}$ such that we explore the relationship between x_i and x_j for $i \in G$ and all j. Global attention allows to connect indirectly any pair (x_i, x_j). In fact, the article is based on the mathematical result that every complete graph can be approximated by random graphs. In this scenario, these three elements in the architecture aims at reproducing the original attention graph. As proven in Theorem 3 in [7], the complexity using sparse attention reduces to $O(n)$.

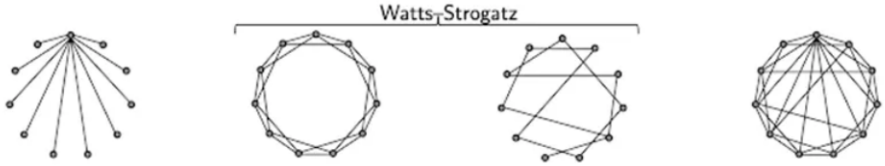


Figure 7: The effect of global, window and random attention in generating a sparse graph. Here, tokens represent nodes and the similarity scores calculated between tokens is the weight associated to the edges. It is important to note that the number of edges is equal to the number of inner products involved in the computation of the attention

Longformer, presented in [8], is an alternative that shares similarities with BigBird. This architecture scales linearly with the input sequence, making it efficient for longer sequences. It is composed by three elements that combined form the sparse attention: sliding window, dilated sliding window and global attention, see Figure 9.

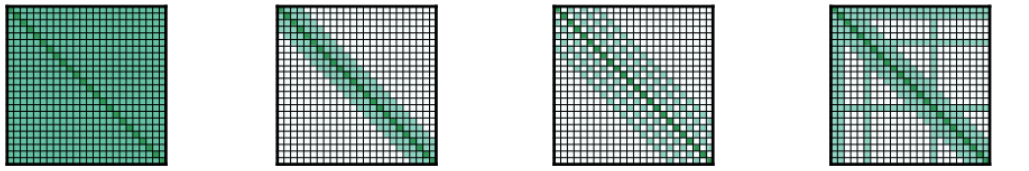


Figure 8: Visual explanation of the attention patterns in Longformer

First of all, given a fixed window size w, each token attends to $w/2$ tokens on each side. Therefore, the computation complexity of this pattern is $O(n \times w)$, which scales linearly with input sequence length n. By using multiple stacked layers of such windowed attention, this results in a large receptive field that is able to capture information across the entire input. To further increase the receptive field without increasing computation, the sliding window can be "dilated" by gaps of size dilation d. In a transformer with L layers, assuming w, d fixed, the receptive field is $L \times d \times w$, which can reach tens of thousands of tokens even for small values of d. In paper [8] they underline that in multi-headed attention, each attention head computes a different attention score. Therefore settings with different dilation configurations per head improves performance by allowing, inside a layer, some heads without dilation to focus on local context, while others with dilation focus on longer context. In this

way, we are "breaking down" the full attention in different experts that still aims at exploring the context at different scales.
Similarly to BigBird, they add global attention on few pre-selected input locations to allow enough flexibility to learn task-specific representations. Since the number of such tokens is small relative to and independent of n the complexity of the combined local and global attention is still $O(n)$. The attention coefficients are still computed as

$$Attention(Q, K, V) = Softmax\Big(\frac{QK^T}{\sqrt{k}}\Big)V,$$

but they use Q_s, K_s, V_s to compute attention scores of sliding window attention, while Q_g, K_g, V_g to compute attention scores for the global attention, following idea of Figure 9. In paper [8] the authors emphasise the good performance of the model for long sequences, similarly to BigBird.

2.1.2 Linear attention

In paper [10] the authors propose a method that is able to scale linearly with respect to the dimension of the output. They introduce the so called *linear transformer* that is a reformulation of self-attention by using a kernel-based formulation. To follow their notation, let us denote (cfr. section 1.2.2 for the definition of attention)

$$V' = Softmax\Big(\frac{QK^T}{\sqrt{k}}\Big)V. \tag{14}$$

We can write a generalized attention equation for any similarity function for the $i-$th row of V' as follow

$$V'_i = \frac{\sum_{j=1}^{n} \mathrm{sim}(Q_i, K_j)V_j}{\sum_{j=1}^{n} \mathrm{sim}(Q_i, K_j)} \tag{15}$$

where $\mathrm{sim}(Q_i, K_j)$ is a non-negative function that expresses how similar are two vectors. Equation (15) reduces to (14) in the case that

$$\mathrm{sim}(Q_i, K_j) = \exp\Big(\frac{Q_i^T K_j}{\sqrt{k}}\Big).$$

Before continuing it is essential to introduce the following definitions and results.

Definition 1. A Hilbert function space H is a reproducing kernel Hilbert space (RKHS) if the evaluation functionals $F_x \colon H \to \mathbb{R}$ such that $F_x(f) = f(x)$, $f \colon X \to X$, are continuous, i.e.

$$|F_x(f)| = |f(x)| \leq M||f|| \quad \forall f \in H, \quad \forall x \in X$$

for a given $M > 0$.

Invoking Riesz's representation theorem, every RKHS has a special function associated to it, namely the reproducing kernel:

Definition 2. A reproducing kernel is a function $K \colon X \times X \to \mathbb{R}$ such that

1. $K(x, \cdot) \in H$, $\forall x \in X$ and
2. $(f, K_x) = f(x)$, $\forall f \in H$ and $x \in X$.

The intuition for a reproducing kernel is a function that is able to reconstruct the elements of an Hilbert space. Given a measure μ, if $H = L^2(X, \mu)$, then

$$\begin{aligned} f(x) = (f, K_x) &= \int_X f(x')K_x(x')\, d\mu(x') \\ &= \int_X f(x')K(x, x')\, d\mu(x'), \end{aligned}$$

which shows that K is a function that is able to determine the punctual value of f if it accesses to all the information about f in terms of similarity to the interested point. From the uniqueness in Riesz's representation theorem it is immediate to conclude the following:

Theorem 2.1. *Every reproducing kernel K induces a unique RKHS, and every RKHS has a unique reproducing kernel.*

The interesting property of reproducing kernel is that they are a measure of similarity in a different (usually big) dimensional space. Let us now recall Mercer-Hilbert-Schmit theorem (cfr. [19]).

Theorem 2.2. *Let $X \subset \mathbb{R}^n$ be closed, μ a strictly positive Borel measure on X, K a continuous function on $X \times X$ satisfying the following conditions: for any finite set of points $\{x_i\}_{i=1}^N$ in X and real number $\{a_i\}_{i=1}^N$*

$$\sum_{i,j=1}^{N} a_i a_j K(x_i, x_j) \geq 0,$$

and

$$\int_X \int_X K(x,y)^2 \, d\mu(x) d\mu(y) < \infty$$

then

$$K(x,y) = \sum_{k=1}^{\infty} \lambda_k \phi_k(x) \phi_k(y)$$

where the series converges absolutely for each pair $(x,t) \in X \times X$ and uniformly on each compact subset of X. The coefficients λ_k forms a countable system of nonnegative eigenvalues, satisfying $\sum_{k=1}^{\infty} \lambda_k^2 < \infty$, of the operator

$$L_K f(x) = \int_X K(x,t) f(t) \, d\mu(t).$$

Now, in view of Theorem 2.2 it holds that a symmetric positive-definite reproducing kernel can be expressed as

$$K(x,y) = \phi(x)^T \phi(y) \tag{16}$$

where $\phi \colon X \to F$ is a feature map with F a Hilbert space. Conversely, every feature map defines a unique reproducing kernel according to (16).
We can assume sim, loosing generality, to be a reproducing kernel, so that $\text{sim}(a,b) = \phi(a)^T \phi(b)$. Therefore, substituting this in (15), we obtain

$$V_i' = \frac{\sum_{j=1}^{n} \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^{n} \phi(Q_i)^T \phi(K_j)}. \tag{17}$$

In this way we have an attention computation consisting only on dot products. Using associativity in (17) we get

$$V_i' = \frac{\phi(Q_i)^T \sum_{j=1}^{n} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{n} \phi(K_j)}.$$

Previous equation defines the *linear transformer* and it is crucial to underline that has a complexity of $O(n)$ because we can compute $\sum_{j=1}^{n} \phi(K_j) V_j^T$ and $\sum_{j=1}^{n} \phi(K_j)$ once and reuse them for every query and so n times. The same idea can be used for the masking such that the $i-$th position can only be influenced by a position j if and only if $j \leq i$, observing that we can express the attention as

$$V_i' = \frac{\sum_{j=1}^{i} \text{sim}(Q_i, K_j) V_j}{\sum_{j=1}^{i} \text{sim}(Q_i, K_j)} = \frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)}.$$

An alternative and simpler procedure is using the first-order approximation of the Taylor of the sim function. This idea is less general but shares similarities with the procedure mentioned above. Let us consider

$$e^{Q_i^T K_j} \approx 1 + Q_i^T K_j.$$

To ensure $1 + Q_i^T K_j \geq 0$ we can normalize Q_i and K_j using the l_2 norm so that $-1 \leq Q_i^T K_j \leq 1$. Equation (14) for the $i-$th entry becomes

$$V_i' = \frac{\sum_{j=1}^{n} \left(1 + \left(\frac{Q_i}{||Q_i||_2}\right)^T \sum_{j=1}^{n} \left(\frac{K_j}{||K_j||_2}\right)\right) V_j}{\sum_{j=1}^{n} \left(1 + \frac{Q_i}{||Q_i||_2}\right)^T \left(\frac{K_j}{||K_j||_2}\right)}$$

and simplified as:

$$V_i' = \frac{\sum_{j=1}^{n} V_j + \left(\frac{Q_i}{||Q_i||_2}\right)^T \left(\frac{K_j}{||K_j||_2}\right) V_j}{n + \left(\frac{Q_i}{||Q_i||_2}\right)^T \sum_{j=1}^{n} \left(\frac{K_j}{||K_j||_2}\right)}.$$

In paper [10] the authors show that linear transformer on image generation and automatic speech recognition can reach the performance levels of transformer, while being up to three orders of magnitude faster during inference.

2.1.3 Matrix factorization

In matrix factorization we assume that the matrix corresponding to the self-attention values is low rank, i.e., not all items are independent of each other. Therefore, the matrix QK^T which is, without word embedding, $n \times n$ is reduced to a matrix $n \times k'$ with $k' < n$ without significant loss in information. Since taking row-wise softmax of a square matrix of order n has complexity $o(n^2)$, reducing the matrix QK^T improves significantly the efficiency.
First of all, in paper [13] the authors perform a singular value decomposition (SVD) on the attention weights matrix

$$P = Softmax\left(\frac{Q^{(h)}(K^{(h)})^T}{\sqrt{k}}\right)$$

across different layers and different heads of the model. The results exhibit a clear long-tail spectrum distribution across each layer, head and task. This implies that most of the information of matrix P can be recovered from the first few largest singular values.

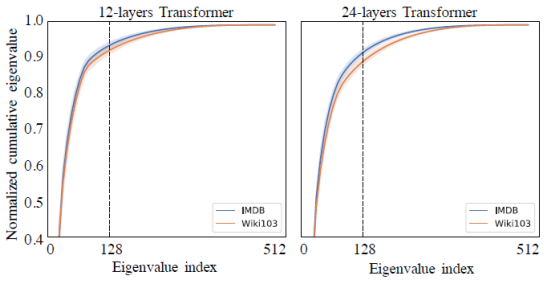


Figure 9: Spectrum analysis of the self-attention matrix for the two tasks performed

The experimental results are confirmed by the theoretical ones: the authors show that given the matrix P there exists a matrix $\tilde{P}$ with small approximation error, namely that

$$Pr(||\tilde{P}v^T - Pv^T|| < \epsilon||Pv^T||) > 1 - o(1) \tag{18}$$

for any v value vector. In other words, we are reading the approximation based on the action of that matrix on values vectors, i.e., the attention values. First of all, we can rewrite matrix P as

$$P = Softmax\underbrace{\left(\frac{Q^{(h)}(K^{(h)})^T}{\sqrt{k}}\right)}_{A} = \exp(A) \cdot D_A^{-1}$$

where D_A is a diagonal matrix. Now $\tilde{P}$ is defined as $\tilde{P} = \exp(A) \cdot D_A^{-1} R^T R$ where $R \in \mathbb{R}^{k' \times n}$ with random i.i.d. entries. The proof that $\tilde{P}$ approximates in the sense of (18) P relies on the Johnson-lindenstrauss (JL) lemma which states that

Lemma 2.1. *Given $0 < \varepsilon < 1$, a set X of m points in $\mathbb{R}^N$, and an integer $n > 8(\ln m)/\varepsilon^2$, there is a linear map $f : \mathbb{R}^N \to \mathbb{R}^n$ such that*

$$(1-\varepsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2$$

for all $u, v \in X$.

In simple terms, the lemma asserts that a set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved (provided $\epsilon > 0$ is not too small).
So far we proved that matrix P can be approximated in low-rank terms. One straightforward ideas can be to use the SVD to approximate the attention weights, but as pointed out by the authors this will results in a much heavier architecture. As a consequence, invoking JL lemma (cfr. paper [13]) and recalling that applying a linear map can be thought as a matrix multiplication, they redefine attention values for a given head as

$$head_i = \underbrace{Softmax\left(\frac{Q^{(i)}(E_i K^{(i)})^T}{\sqrt{k}}\right)}_{n \times k'} \underbrace{F_i V^{(i)}}_{k' \times d}. \tag{19}$$

In equation (19) we project the $(n \times k)$-dimensional key and values layers $K^{(i)}$, $V^{(i)}$ into $(k' \times k)$-dimensional projected key and value layers. Thus, if we can choose a very small projected dimension k', then we can significantly reduce the memory and space consumption.

3 Behind the scenes of Self-Attention

In this section we aim at describing in both mathematical and intuitive terms the Self-Attention mechanism. First of all, for a given head, the formula

$$Attention(Q, K, V) = Softmax\Big(\frac{QK^T}{\sqrt{k}}\Big)V$$

can be described in geometrical terms. Let us consider an embedded word $x_i \in \mathbb{R}^d$. According to formulas introduced in (3), the corresponding query and key vector are computed as

$$q_i = W_q^T(x_i), \quad k_i = W_k^T(x_i)$$

where $W_q, W_k \in \mathbb{R}^{d\times d}$. To compute then the attention coefficients, we perform the dot product $\langle q_i, k_j\rangle \in \mathbb{R}^+$ where k_j is the key vector for another embedded word x_j. Let us explain what happens from a geometric point of view: matrices W_k and W_q represents linear transformation that deforms the space.
As a consequence, query and key vector can be thought of representing the original vector x_i with a new pair of coordinates in $\mathbb{R}^d$. Subsequently, by performing the dot product between query and key vectors we are capturing how similar or dissimilar are they. From a linguistic point of view, we expect (and this is done by back propagation by adjusting the values in the matrices) that the embeddings of the various words cluster similar words and separate dissimilar words. The softmax is then applied to normalize the attention weights and then we multiply by V to obtain the attention values. Therefore, we are weighting all the values vector (i.e. another representation of the embedded word x_i) by coefficients that encode the similarity by words. In other terms, we are relating each input word with all the other input words in a way that we capture affinity between the words themselves.

To geometrically visualize the intuition, let us consider for the sake of simplicity two points $a, b \in \mathbb{R}^2$ and a third point c that is somehow in between a, b. We can think of c as a word that has multiple meaning and provide context.

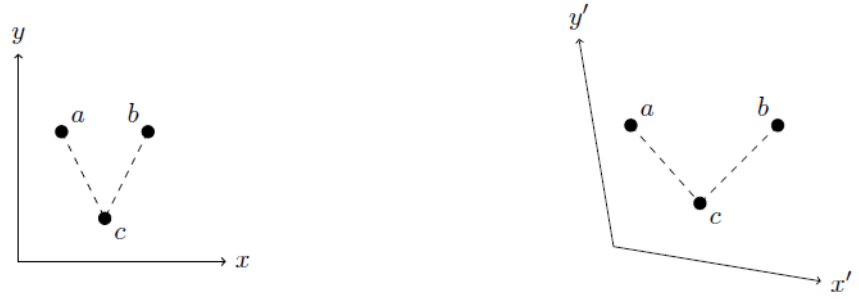


Figure 10: Visual explanation of how moving the space can be helpful to encode context.

When we provide context we are weighting each word according to all the other input words and so in our simple example, c will shift towards either a or b. It is clear that the right case sketched in above picture works better in moving c according to the context. Finally, W_v defined in (5) is taking profit of this context aware embedding and capture long term relationship.

In its generality, given $m \in \mathbb{N}$ pairs of key and value vectors (k_i, v_i), $k_i, v_i \in \mathbb{R}^d$ where d is the dimension of the embedding, for a given query vector $q_j \in \mathbb{R}^d$ the attention mechanism computes an output vector o_j as

$$o_j = \sum_{i=1}^{m} \alpha(q_j, k_i) g(v_i).$$

The function $g(\cdot)$ is a linear application from $\mathbb{R}^d$ to $\mathbb{R}^d$, while function $\alpha(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ computes the attention weights. For instance, in the case of transformers,

$$\alpha(q_j, k_i) = \frac{exp(\langle q_j, k_i \rangle)}{\sum_{l=1}^{m} exp(\langle q_j, k_l \rangle)}, \qquad g(v_i) = v_i.$$

In literature, the function α is generally referred to as *attention pooling* and it can be thought as a similarity measure. In general we can ask α to have various properties, but in general, for a fixed query vector q_j, we require the weights $\alpha(q_j, k_i)$ to form a convex combination, i.e.,

$$\sum_{i=1}^{m} \alpha(q_j, k_i) = 1, \quad \alpha(q_j, k_i) \geq 0 \qquad j = 1, \ldots, m.$$

With this notation, the geometrical intuition can be expressed in analytical terms assuming function α to be the normalized dot product,

$$o_j = \frac{1}{C}[\langle q_j, k_1 \rangle g(v_1) + \ldots + \langle q_j, k_m \rangle g(v_m)], \quad C = \sum_{i=1}^{m} \langle q_j, k_i \rangle.$$

Now if, for example, q_j attends mostly k_1, then

$$o_j \approx \frac{1}{C}[\langle q_j, k_1 \rangle g(v_1)] \approx g(v_1).$$

Therefore, we can think of attention weights as a "guide map" to values vectors $g(v_i)$.

In paper [12] the authors delve into the multi-head attention mechanism and evaluate the contribution made by individual attention heads in the encoder to the overall performance of the model. They found that for a translation task, only a small number of heads are important for translation and these heads play interpretable "roles". They showed that most of the heads can be pruned without significant loss in quality, but after the training process. Inspired by their work, our aim is to explore at a deep level the self-attention mechanism.

3.1 Experiments on attention layer

In this section we are going to present our results from the experiments that will lead to an alternative formulation for the attention weights. It is important to underline a couple of aspects before continuing: the aim of this thesis is to describe and analyze Self-Attention in mathematical terms. As a consequence, the experiments with the Transformer architecture are not optimal and we trained, for computational time reasons, just using one transformer block. For instance the results we got from the translation task are very far from the proper translation. Moreover, we focused on the original functions

$$\alpha(q_j, k_i) = \frac{\exp(\langle q_j, k_i \rangle)}{\sum_{l=1}^{m} \exp(\langle q_j, k_l \rangle)}, \qquad g(v_i) = v_i.$$

The conclusions we draw may be different with other functions.

In the experimental part we used the opus_books from HuggingFace from English to Italian[2], which included a large variety of texts, articles and web pages. In order to understand at a basic level what is the attention layer doing, we decided to train the transformer for 20 epochs with just 1 transformer block and either 1 or 8 heads, to better understand the flow of information and prevent the depth of the neural network from altering the scores. The goal is to produce a correct translation of the sentence involved, so the attention scores reflect this task.

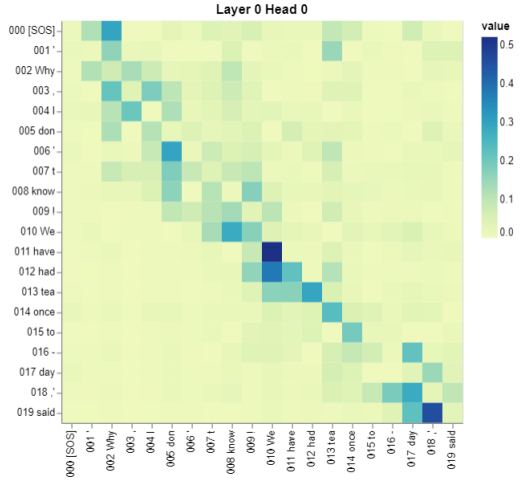


Figure 11: Attention visualization architecture with 1 layer and 1 head

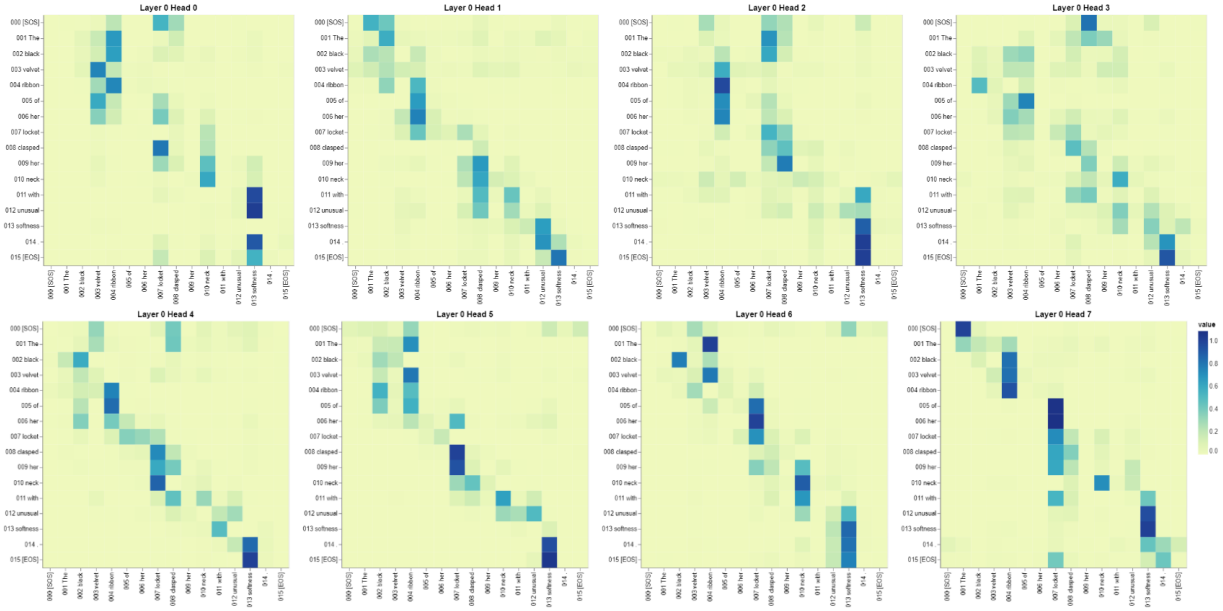


Figure 12: Attention visualization architecture with 1 layer and 8 heads

Before describing the above figures, let us remark the fact that, in paper [12] the matrices are better structured (and we will explain what do we mean by structure) as their experiments involved a more complex architecture. Nevertheless, our experiments, with a certain degree of error, still confirm their results.
After adapting the code of the transformer architecture we used a function to get and visualize the attention matrix for a specific sentence. We then focused on visually analysing the structure of the arrays for many different examples and architectures: with 1 head, with 8. In Figure 11 and 12 we report an example of our experimental session. These figures clearly show that attention weights matrices exhibit a certain structure while performing translation. Most of the heads show a diagonal structure in the matrix since translating the token referencing to the word itself should be the most important to produce the translated token. Furthermore, it is possible to see in head 2 and head 7 that there is stress on a specific token and this is identified by a darker vertical column in the matrix. This phenomenon captures another aspect of translating: in producing language there are words that are giving the general meaning or context of the whole sentence or words that are referencing to other words

[2]see https://huggingface.co/docs/datasets/index

such as pronouns. As a consequence, these layer with token attending a specific one are encoding this specific aspect. Furthermore it is worthy to underline that all the matrix involved are full rank with low l_2 norm.

3.1.1 On the structure of attention weights

The experiments carried shows a specific structure in the attention pooling function for the transformer

$$f(Q,K) = Softmax\Big(\frac{QK^T}{\sqrt{k}}\Big).$$

Three general structured heads can be identified:

- positional heads: tokens attend to a token's immediate neighbors;
- syntactic heads: tokens attending another token because of linguistic relationship;
- rare tokens heads: a group of token attending a specific token, usually an infrequent token.

What is interesting is that the attention scores are never spread among the matrix but the values are mostly concentrated around the diagonal, up to a proper shift of them. Taking into consideration this and the roles of heads we propose the following formulation for the attention scores:

$$f(P,\tilde{E}) = (I_nP + \tilde{E}), \qquad P,\tilde{E} \in \mathbb{R}^{n\times n}, \quad P \in \Sigma \subset \mathcal{M}_n(\mathbb{R}) \tag{20}$$

and therefore the new attention values

$$(I_nP + \tilde{E})V.$$

We write I_nP instead of P to stress on the action of P in shifting the attention. The matrix $\tilde{E}$ is an sparse error matrix with elements $|\epsilon_{i,j}| \leq \epsilon$ so with very low values, while P is a matrix taken from a set Σ with a specific structure that we will discuss afterwards. Before describing the matrices we are dealing with and their properties let us clarify the intuition behind formula (20): the role of the matrix P is to shift the positional heads, i.e. the diagonal, to the relevant structure that attention scores exhibit. In some sense, we are forcing query and keys to attend each other in a specific form such that encode the different roles of attention heads and so telling values where to look to get the context. After doing that we add an error matrix to introduce randomness and add elasticity to the formula.

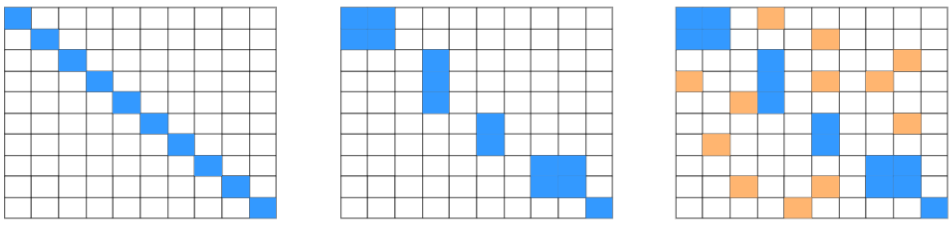


Figure 13: Effect of shifting the attention and adding sparse error. We start from the identity, we apply (central image) the matrix P that is shifting the attention and creating a specific structure, then (right image) we add some random noise, represented by orange blocks, to add robustness to our formula.

For example in Figure 13 we are forcing token $3, 4, 5$ to attend token 4 as we can

see in the central image since we detect a column of blue blocks. The matrix P in (20) has a specific structure. We can define 3 family matrices that we are interested in: $\Sigma_1 = I_n$ to capture the positional head as we are dealing with mostly diagonal matrices, $\Sigma_2(w)$ for a windows size w which is made by appropriate matrices to capture syntactic heads, namely block matrices that correlate tokens to some other specific tokens as in Figure 12 for head 6: here token $5, 6, 7$, namely words 'of', 'her', 'locket' are attending the word 'locket' as we can see a blue vertical column. In the syntactic structure of the sentence, the preposition 'of' and the possessive pronoun 'her' both relate to the noun 'locket' according to the grammatical rule of English sentence construction. In this structure, 'of' indicates the possession relationship between 'her' and 'locket', so both tokens 5 and 6 must point to token 7, 'locket', which represents the possessed object.
Lastly Σ_3 represents matrices able to codify rare tokens and so that all token are attending a specific one as, almost, for head 7 in Figure 12 in which we can see a vertical darker column for token 7, meaning that attention is pooled on this specific one. With this notation

$$\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \tag{21}$$

and the matrix P is taken among these possible sets. The windows size w is important to control the size of the blocks in Σ_2 and also it is consistent with the idea that syntactic relationships as verb-adjective are mainly close. We will dedicate particular attention on the role of w in the next pages.

3.1.2 Description of the sets

In this section we are going to describe the sets that appears in (21). The set Σ_1 consists of I_n as we are shifting the attention to the token position, and so the resulting matrix should be diagonal. The set Σ_3 is related to rare token heads of the form shown in Figure 14

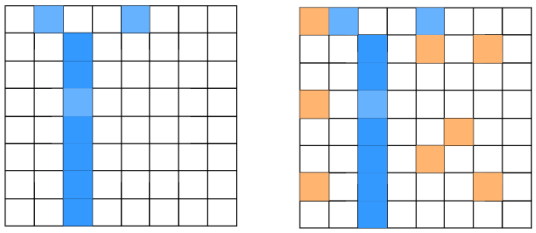


Figure 14: Exemplification of rare token heads.

These kind of matrices are characterized by two parameters: $\mathcal{T} = \{t_1, \dots, t_s\}$ the set of ordered tokens the attention focus on and their relative windows size $\mathcal{W} = \{w_1, \dots, w_s\}$ in the sense that token t_1 is attended by w_1 surrounding tokens and t_s by w_s surrounding tokens. It is sufficient to describe a single token with its window (t_1, w_1) as the general case would be just a concatenation of matrices. The base case, assuming t_1 in position i is represented by matrices of the form

$$\left[\begin{array}{c|c|c} I_{d_1} & 0 & 0 \\ \hline 0 & R & 0 \\ \hline 0 & 0 & I_{d_2} \end{array}\right]$$

where matrix R is of the form:

$$\begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$$

where we have w vertical 1s in one column i. This is capturing that this group of tokens are attending a specific token, pooling all attention on it. The base case it is just a concatenation of I_{d_1}, R, I_{d_2} such that $d_1 + w_1 + d_2 = n$ in the sense that we have the identity, then the rare token block and the identity.

Finally, the last set is Σ_2 in which we a structure close to a block diagonal matrix in which we have word, or group of words, attending other group of words capturing syntactic relationship. From the experiments it is possible to see (for example Figure 12 head 4) that group of words attending other group of words are usually close, at least in English vocabulary, therefore an easy way to describe these matrices would be to start with a multi diagonal matrix and apply a "dropout" to obtain the desired structure (see Figure 15).

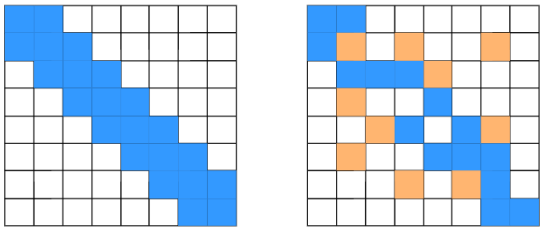


Figure 15: Exemplification of syntactic heads: we start from a band matrix, turn off some blocks and then add random orange blocks.

The dropout can be applied also to create "separate groups" capturing that we have no relevant information as punctuation. In mathematical terms, given a matrix $B \in \mathbb{R}^{n\times n}$ such that

$$b_{i,j} = 0 \quad \text{if } i > j + w, \quad \text{or} \quad j > i + w; \quad w \geq 0$$

where w is the bandwidth, then

$$P = \text{dropout}(B, p), \quad \text{for some } p \in (0, 1).$$

The three sets of matrices we defined can be thought as band matrices with an error matrix. Indeed, for rare token matrices of the form if we fix the windows size w, then the matrix is contained in a band matrix because the vertical column in matrix R does not, by construction, exceed the bandwidth.

We are now ready to express our ideas in mathematical and formal terms. First of all, let us define the following set for a fixed bandwidth size $w \in \mathbb{N}$:

$$\mathcal{B} = \{B \in \mathcal{M}_n(\mathbb{R}) \mid B \text{ is a } w\text{-band matrix}\}.$$

Let us also consider the entrywise $1-$norm on the set of real matrices

$$|M| = \sum_{i,j} |m_{i,j}|, \quad m_{i,j} \in M, \quad M \in \mathcal{M}_n(\mathbb{R}). \tag{22}$$

It is clear now that $\mathcal{M}_n(\mathbb{R})$ with this norm, becomes a normed vector space. Moreover, $\mathcal{B}$ is a finite dimensional subspace of $\mathcal{M}_n(\mathbb{R})$ because if $B_1, B_2 \in \mathcal{B}$ and $\lambda_1, \lambda_2 \in \mathbb{R}$ then since we have a fixed bandwidth clearly

$$\lambda_1 B_1 + \lambda_2 B_2 \in \mathcal{B}.$$

It is a known result that any finite dimensional subspace of a normed vector space is closed, hence we conclude that $\mathcal{B}$ is closed. It is also trivial that any linear subspace is also convex. In our case the space $\mathcal{B}$ has dimension

$$n + 2(n-1) + \ldots + 2(n-w) = 2\sum_{i=0}^{w}(n-i) - n = -(w+1)(w-2n) - n.$$

As a consequence, we can guarantee that given the matrix H for a certain head, there exists a unique $\tilde{B} \in \mathcal{B}$ such that

$$\tilde{B} = \underset{B \in \mathcal{B}}{\operatorname{argmin}} \, |H - B|$$

in view of the following theorem (see [17])

Theorem 3.1. *Let $\mathcal{H}$ be an Hilbert space with respect the norm $||\cdot||$. If C is a convex, closed, non-empty subset of $\mathcal{H}$ and q a point of $\mathcal{H}$. Then, there exists a unique $p \in C$ such that*

$$||p - q|| = dist(C, q) := \inf_{x \in C} ||x - q||.$$

In other words, given an attention weights matrix H, we can project onto the subspace $\mathcal{B}$ and obtain the best approximation of H in terms of band matrices. The experiments we carried out, show that most of the information is around the diagonal, which further motivates the choice of band matrices. However, the other entries with less weight (i.e. with a lower value, the yellow blocks in Figure 11 and 12) are non zero. The idea is to capture them with sparse random matrix as shown when we added the orange blocks in Figure 14 and Figure 15.
Therefore, fixed ϵ a small real value, we define the following set

$$\mathcal{X} = \{B + E_{rr} \mid B \in \mathcal{B}_{[0,1]}, \, E_{rr} \in \mathcal{M}_n(\mathbb{R})\}$$

where $\mathcal{B}_{[0,1]}$ is the set of matrices $\mathcal{B}$ with entries in $[0, 1]$ and E_{rr} is a sparse matrix such that

$$\max_{i,j} |e_{i,j}| \leq \epsilon, \quad E_{rr} = (e_{i,j}). \tag{23}$$

Now, it is straightforward that $\tilde{B} \in \mathcal{X}$ because matrix H has entries in $[0, 1]$ by definition of attention weights and we can take E_{rr} as the zero matrix. The set $\mathcal{X}$ is clearly bounded w.r.t norm (22) because given $X_1, X_2 \in \mathcal{X}$, exists $L > 0$, such that

$$|X_1 - X_2| < L.$$

Indeed let n_1 the number of elements in the band and n_2 the remaining elements so that $n_1 + n_2 = n^2$, then

$$|X_1 - X_2| < n_1 + \epsilon n_2.$$

Now, we are left to prove that the space $\mathcal{X}$ is closed. If we prove this, we can conclude that $\mathcal{X}$ is a compact set and the function $f \colon \mathcal{X} \to \mathbb{R}^+$ such that

$$f(X) = |H - X|$$

attains a minimum value on $\mathcal{X}$. Since $\tilde{B} \in \mathcal{X}$ this minimum value is less or equal than $|H - \tilde{B}|$ and so this is the right space to approximate our attention weights matrix.
To conclude that $\mathcal{X}$ is closed we have to prove that given $X_n \in \mathcal{X}$ such that

$$X_n \to X \implies X \in \mathcal{X} \quad \text{for } n \to \infty.$$

From the definition of the norm we are using, we have that

$$\lim_{n \to \infty} \sum_{i,j} |x_{i,j}^{(n)} - x_{i,j}| = 0, \quad x_{i,j}^{(n)} \in X_n, \quad x_{i,j} \in X$$

and so, in particular,

$$\lim_{n \to \infty} \max_{i,j} |x_{i,j}^{(n)} - x_{i,j}| = 0$$

which implies the pointwise convergence

$$\lim_{n\to\infty} x_{i,j}^{(n)} = x_{i,j} \quad \forall i,j.$$

Now, if we split $X_n = B_n + E_{rr}^{(n)}$ and focusing on the band part, it is immediate to conclude from the point-wise convergence that

$$\lim_{n\to\infty} B_n$$

is a band matrix with bandwidth less or equal that w. Now, let us focus on the limit on the error sparse matrix

$$\lim_{n\to\infty} E_{rr}^{(n)}.$$

According to our definition of norm and from the existence of $\lim_{n\to\infty} X_n$ we conclude that

$$\lim_{n\to\infty} e_{i,j}^{(n)}, \qquad E_{rr}^{(n)} = (e_{i,j}^{(n)})$$

exists. Now, let us fix i,j and assume the limit to be l. From (23) we conclude that $l < \epsilon$. Furthermore, we have only two scenarios from the existence of the limit: either $e_{i,j}^{(n)} = 0$ or $e_{i,j}^{(n)} \neq 0$ from a certain index N such that $n > N$. However, the second case may happen for a relatively small amount of times as the matrices involved are sparse (otherwise we would violate this condition). Therefore, the first case must be the predominant one and we can conclude that

$$\lim_{n\to\infty} E_{rr}^{(n)}$$

is "mostly" 0 and so we have an error sparse matrix.

3.1.3 Validation of attention scores approximation

We conclude our project validating formula (20) with respect to the original attention scores matrix. In other words, let us denote

$$A = Softmax\Big(\frac{QK^T}{\sqrt{k}}\Big)$$

arising after training the original transformer model, we aim at computing $|A - X|$ where X is a matrix as in (20). To do so, we coded three different functions to generate the matrices belonging to sets $\Sigma_1, \Sigma_2, \Sigma_3$ and then we computed the distance between A taken from the previous experiments with 8 heads and a set of matrices generated from these sets. We took the attention score heads for a sentence of length 16 (we can take one sentence because structure of attention scores is mostly the same, independently from the sentence taken as it can be seen in paper [12]), and compute the minimal distance generating 30 matrices from each Σ_i with $w = 3$ and $num\text{-}pos = 2$ (respectively the windows size/context and the number of words rare token attends). Also in this case the validation session was carried out for several matrices taken from the encoder. For simplicity we present an example of results, even tough generally they are the same, summarized in Tables 16 and 17.
The Table 16 shows that the mean error, which is the error per entries, is relatively small, concluding that we are properly capturing the structure of the weight matrix. It is important the role of w and $num - pos$. If for example we change them to be 10 and 1, respectively, we see in Table 17, considering that the attention weights matrix is relatively small and the randomness of our approach, that the mean error significantly increases.

Each language (Italian, English,...) and/or each type of text (e.g. by author) is expected to have optimal w and $num - pos$ parameters. Knowing these parameters

	Head_n	Min_distance	Mean Error
0	1	17.229423	0.067302
1	2	15.748942	0.061519
2	3	21.084405	0.082361
3	4	18.796007	0.073422
4	5	16.912899	0.066066
5	6	17.232267	0.067314
6	7	18.121464	0.070787
7	8	18.711096	0.073090

Figure 16: Validation results showing the approximation distance and the mean distance per element ($w = 3$ and num-pos= 2)

	Head_n	Min_distance	Mean Error
0	1	22.584185	0.088219
1	2	20.904304	0.081657
2	3	20.630814	0.080589
3	4	19.964075	0.077985
4	5	21.000884	0.082035
5	6	22.181472	0.086646
6	7	21.396999	0.083582
7	8	21.169499	0.082693

Figure 17: Validation results showing the approximation distance and the mean distance per element ($w = 10$ and $num - pos = 1$)

allows to represent the attention score matrix as a matrix with structure properties that would lead to relevant reduction of the total computational cost of the LLM model.

4 Conclusion and further investigations

The aim of this thesis was to understand at a profound level the role of attention mechanism in transformer. We were able to provide both a geometrical and analytical explanation for the formula of attention values and in particular of attention scores. The alternative and lighter formulation we found is interesting as it encodes the structure of the matrices that experiments revealed. In this scenario, then, attention is not a random process governed by back-propagation but is enriched with meaning and deeply reflects the role of translation. Far from the scope of this thesis but yet interesting would be to study the role of parameters w and num-pos in the code for the validation as we believe they contain valuable information on the idiom used. Indeed, by optimizing these hyperparameters for a given language we can understand how many words are relevant in a sentence and the general context size. Indeed, as pointed out in paper [18] the complexity of the language and its syntactic characteristics have to be taken into consideration during the construction of LM architectures.

References

[1] A. Vaswani et al., *Attention Is All You Need*, NIPS, 2017

[2] D. Bahdanau, K. Cho, Y. Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, arXiv, 2014

[3] M.T. Luong, H. Pham, C. D. Manning, *Effective Approaches to Attention-based Neural Machine Translation*, arXiv, 2015

[4] R.S. Navid, *Introduction to Transformers*, Institute of Computational Perception CP Lectures, 2020

[5] A. Kak, C Bouman, *Transformers: Learning with Purely Attention Based Networks*, Lectures on Deep Learning Purdue University, 2023

[6] J. Devlin, M. W. Chang, K. Lee, K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint, 2018

[7] M. Zaheer et al., *Big Bird: Transformers for Longer Sequences*, NIPS, 2020

[8] I. Beltagy, M. E. Peters, A. Cohan, *Longformer: The Long-Document Transformer*, arXiv, 2020

[9] I. Chalkidi et al., *An Exploration of Hierarchical Attention Transformers for Efficient Long Document Classification*, arXiv, 2022

[10] A. Katharopoulos et al., *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*, arXiv, 2020

[11] R. Li et al., *Linear Attention Mechanism: An Efficient Attention for Semantic Segmentation*, arXiv, 2020

[12] E. Voita et al., *Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned*, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019

[13] S. Wang et al., *Linformer: Self-Attention with Linear Complexity*, arXiv, 2020

[14] Y. Tay et al., *Long range arena: a benchmark for efficient transformers*, arXiv, 2020

[15] A. Zhang et al., *Dive into Deep Learning*, 2023

[16] S. Ji, Y. Xie, H. Gao, *A Mathematical View of Attention Models in Deep Learning*, Lectures at Texas A&M University, 2019

[17] D. Foschi, *Appunti di Analisi 3*, Lecture notes at Università degli studi di Ferrara, 2020

[18] D. Gerz et al., *On the Relation between Linguistic Typology and (Limitations of) Multilingual Language Modeling*, Apollo - University of Cambridge Repository, 2018

[19] Minh, H.Q., Niyogi, P., Yao, Y., *Mercer's Theorem, Feature Maps, and Smoothing*. Lecture Notes in Computer Science(), vol 4005. Springer, Berlin, 2006