



POSITION PAPER

The architecture of language: Understanding the mechanics behind LLMs

Andrea Filippo Ferraris^{1,2} , Davide Audrito³, Luigi Di Caro⁴ and Cristina Poncibò⁵ 

¹Department of Law, University of Bologna, Bologna, Italy; ²DIKE and the Law Faculty, Vrije Universiteit, Brussel, Belgium;

³Computer Science Department, University of Torino, and Legal Studies Department, University of Bologna, Bologna, Italy;

⁴Computer Science Department, University of Torino, Torino, Italy and ⁵Department of Law, University of Turin, Torino, Italy

Corresponding author: Cristina Poncibò; Email: cristina.poncibo@unito.it

(Received 1 December 2024; accepted 11 December 2024)

Abstract

Large language models (LLMs) have significantly advanced artificial intelligence (AI) and natural language processing (NLP) by excelling in tasks like text generation, machine translation, question answering and sentiment analysis, often rivaling human performance. This paper reviews LLMs' foundations, advancements and applications, beginning with the transformative transformer architecture, which improved on earlier models like recurrent neural networks and convolutional neural networks through self-attention mechanisms that capture long-range dependencies and contextual relationships. Key innovations such as masked language modeling and causal language modeling underpin leading models like Bidirectional encoder representations from transformers (BERT) and the Generative Pre-trained Transformer (GPT) series. The paper highlights scaling laws, model size increases and advanced training techniques that have driven LLMs' growth. It also explores methodologies to enhance their precision and adaptability, including parameter-efficient fine-tuning and prompt engineering. Challenges like high computational demands, biases and hallucinations are addressed, with solutions such as retrieval-augmented generation to improve factual accuracy. By discussing LLMs' strengths, limitations and transformative potential, this paper provides researchers, practitioners and students with a comprehensive understanding. It underscores the importance of ongoing research to improve efficiency, manage ethical concerns and shape the future of AI and language technologies.

Keywords: large language models (LLMs); artificial intelligence (AI); natural language processing (NLP)

1. Introduction

Large language models (LLMs) have revolutionized artificial intelligence (AI) and natural language processing (NLP) by achieving unprecedented proficiency in understanding and generating human language. Built upon the transformative transformer architecture, LLMs excel in tasks such as text generation, machine translation, question answering and sentiment analysis, often matching or surpassing human performance (Naveed et al., 2024).

This paper provides a brief overview of LLMs, touching upon their theoretical foundations, technical advancements and practical applications. We begin by introducing the transformer architecture, which addressed the limitations of earlier models like recurrent neural networks (RNNs) and convolutional neural networks (CNNs) through self-attention mechanisms that capture long-range

dependencies and contextual relationships. We explore how scaling laws, increased model sizes and advanced training techniques have propelled LLMs to new heights. Key innovations such as masked language modeling (MLM) and causal language modeling (CLM) underpin models like BERT (Devlin et al., 2019) and the GPT series. The paper also examines practical methodologies that enhance the adaptability and precision of LLMs, including fine-tuning strategies like parameter-efficient fine-tuning (PEFT) and techniques such as prompt engineering. We address challenges associated with LLMs, such as computational demands, biases and hallucinations – where models generate plausible but incorrect information – and present solutions like retrieval-augmented generation (RAG) to improve factual accuracy.

By outlining both the capabilities and limitations of LLMs, this paper aims to provide a foundational understanding for legal researchers, practitioners and students. We emphasize the transformative potential of these models in shaping the future of AI and language technologies, underscoring the importance of ongoing research to enhance efficiency and address ethical considerations.

2. Understanding the context: NLP and neural networks

NLP is a multidisciplinary field that combines linguistics, computer science and machine learning to enable machines to interpret and generate human language. Early NLP systems relied heavily on rule-based methods, which required extensive domain knowledge and were limited in scalability. These were soon replaced by statistical approaches and simple neural networks like the perceptron (Rosenblatt, 1958), which could learn basic patterns from data. These methods were soon replaced by statistical approaches and simple neural networks (McCulloch & Pitts, 1943) like the **perceptron**, which could learn basic patterns from data.

2.1. Neural networks: Foundations and challenges

Neural networks, inspired by the structure of the human brain, consist of layers of interconnected nodes or “neurons.” These neurons process input data by applying **weights**¹ (scaling factors) and **biases**² (offsets) before passing the result through an activation function. Early models like the perceptron were capable of handling simple classification tasks by adjusting these parameters to minimize errors, as quantified by a **loss function**³ (Terven, Cordova-Esparza, Ramirez-Pedraza, Chavez-Urbiola & Romero-Gonzalez, 2024) – a measure of the difference between predicted and actual outputs. However, these models were limited to linear decision boundaries and struggled with more complex, nonlinear problems.

The introduction of the **backpropagation algorithm** (Werbos, 1974) marked a significant advancement, allowing neural networks to adjust weights and biases more effectively using **gradient descent**. This method calculates gradients of the loss function to iteratively update the network’s parameters. Despite this breakthrough, deeper networks encountered the **vanishing gradient**

¹A weight in a neural network controls the importance of an input in influencing the output of a neuron. It adjusts the input’s value before summing it with others, effectively determining how much the input will impact the neuron’s decision. Higher weights give more significance to certain inputs, guiding the network’s behavior.

²Bias acts as an offset added to the weighted sum of inputs, allowing the activation threshold to shift. This adjustment helps the model better fit the data by enabling more flexible decision boundaries, ensuring the neuron can activate even when inputs are zero or minimal. It functions like the intercept in a linear equation, improving the network’s ability to generalize across diverse input patterns.

³The loss function refers to a measure of how far off a neural network’s predictions are from the actual values. When Paul Werbos introduced the backpropagation algorithm, it allowed networks to adjust their internal parameters, such as weights and biases, by calculating the error through this loss function. The goal during training is to minimize the loss, which indicates that the model’s predictions are becoming more accurate. The backpropagation algorithm updates the network’s parameters to reduce this loss iteratively, leading to improved model performance over time.

problem⁴ (Hochreiter, 1998), where gradients diminished as they propagated backward, slowing or halting the learning process in earlier layers.

2.2. Hardware and architectural advances

The resurgence of neural networks in the 21st century was driven by advancements in hardware, particularly graphics processing units (GPUs), which enabled efficient parallel computation. These improvements made it feasible to train deeper networks on large datasets, resulting in breakthroughs in tasks like computer vision and speech recognition. However, neural networks still faced limitations in handling sequential data and long-range dependencies, crucial for many NLP tasks.

2.3. From RNNs and CNNs to transformers

To address these challenges, more advanced architectures were developed. **RNNs** (Rumelhart, Hinton & Williams, 1986) introduced feedback loops to retain information across time steps, making them suitable for sequential data. Similarly, **CNNs** (LeCun et al., 1989; Lecun, Bottou, Bengio, & Haffner, 1998), designed for grid-like data such as images, provided local pattern detection. While these architectures offered improvements, they still struggled with scalability and efficiently capturing long-range dependencies in NLP tasks.

The introduction of transformers revolutionized NLP by addressing these challenges, offering superior handling of context and enabling parallel processing of large datasets. This innovation laid the groundwork for the development of LLMs, which can capture intricate language patterns and perform complex tasks with remarkable accuracy and fluency.

3. Transformers based architecture: a new paradigm

Introduced in the seminal 2017 paper *Attention is All You Need* (Vaswani et al., 2017), the transformer architecture fundamentally shifted the way models process and understand sequential data by eliminating the need for recurrent and CNNs traditionally used in language models. Instead, transformers rely on a mechanism called self-attention, which allows them to consider the entire input sequence simultaneously rather than processing it step-by-step.

4. Self-attention mechanism

Self-attention is the key innovation of the transformer architecture. Unlike recurrent networks, which process data in order, or convolutional networks, which focus on local patterns, transformers enable each word or token in the input to weigh the relevance of every other word in the sequence. It enables the model to weigh the relevance of each word (or token) in the input sequence with respect to each other word, capturing long-range dependencies and contextual relationships more effectively.

Mathematically, self-attention operates as follows:

1. Input representation:

Given an input sequence of tokens:

$$X = [x_1, x_2, \dots, x_n]$$

⁴The vanishing gradient problem arises during the training of neural networks, mainly when using backpropagation to update weights. It occurs when the gradients of the loss function with respect to the weights become extremely small as they are propagated backward through many layers of the network. This results in weights in the early layers of the network not being updated effectively, which slows down or stops the learning process. For further information see Hochreiter, Sepp. "The vanishing gradient problem during learning recurrent neural nets and problem solutions."

Each token is embedded into a continuous vector space to obtain embeddings:

$$E = [e_1, e_2, \dots, e_n]$$

2. Linear projections:

For each embedding e_i , we compute three vectors: a **query** q_i , a **key** k_i and a **value** v_i , using learned weight matrices W^q , W^k and W^v :

$$q_i = e_i \times W^q$$

$$k_i = e_i \times W^k$$

$$v_i = e_i \times W^v$$

3. Scaled dot-product attention:

The attention score between token i and token j is calculated using the scaled dot-product of their queries and keys:

$$\text{Attention Score}_{i,j} = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

- $q_i \cdot k_j^t$ denotes the dot product of q_i and the transpose of k_j .
- d_k is the dimensionality of the key vectors.
- The division by $\sqrt{d_k}$ scales the dot products to prevent large values that could result in small gradients during training.

The attention weights are obtained by applying the softmax function (Figure 1) to the attention scores:

$$\alpha_{(ij)} = \text{softmax}(\text{Attention Score}_{(ij)})$$

The output for each token i is a weighted sum of the value vectors v_j of all tokens:

$$z_i = \sum_{j=1}^n \alpha_{ij} \cdot v_j$$

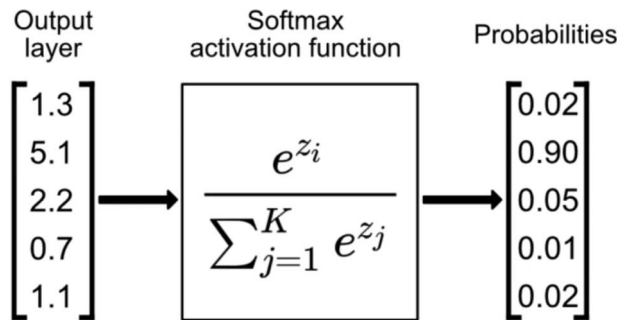


Figure 1. Illustrates the **softmax activation function** as used in large language models (LLMs). Each raw output is exponentiated and then normalized by dividing by the sum of exponentiated outputs, ensuring the resulting probabilities range from 0 to 1.

4. Context vector computation:

Collectively, in matrix form:

$$Q = E \times W^q$$

$$K = E \times W^k$$

$$V = E \times W^v$$

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) \cdot V$$

- **Q**, **K**, and **V** are matrices of queries, keys, and values for all tokens.
- **K^t** is the transpose of **K**.
- The multiplication **Q** × **K^t** computes the attention scores for all pairs of tokens simultaneously.

This mechanism allows the model to focus on relevant parts of the input while generating output. For example, in the sentence “The cat sat on the mat because it was soft,” the model can accurately capture that “it” refers to “the mat” by assigning higher attention weights between these tokens.

4.1. Positional encoding

One challenge in processing sequences simultaneously is maintaining the sense of order in the data, as transformers do not process inputs sequentially like RNNs. To address this, **positional encoding** (Chen et al., 2021) is introduced. As shown in Figure 2, positional encoding adds information about the position of each token in the sequence,⁵ ensuring the model can differentiate between words appearing at different positions and preserve the natural order of language (Kazemnejad, Padhi, Ramamurthy, Das & Reddy, 2023). These encodings are incorporated into the model’s input embeddings,⁶ allowing transformers to maintain both position and context without the need for recurrence.

The positional encoding **PE** is added to the input embeddings to inject positional information:

$$E_{\text{input}} = E + PE$$

The positional encoding is defined using sine and cosine functions of varying frequencies: For position **pos** and dimension **i**:

- For even dimensions (2i):

$$PE(\text{pos}, 2i) = \sin \left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}} \right)$$

⁵Tokenization refers to the process of splitting text into smaller units called *tokens* that the model can process. In the context of transformers, tokenization breaks down sentences into tokens, often words or subwords. Each token corresponds to a specific index in a vocabulary list, allowing the model to work with the text numerically. Tokenization is a crucial first step in transforming natural language into a format that a model can understand, making it an integral part of how transformers handle input sequences and extract meaning.

⁶An embedding is a numerical representation of tokens (words or subwords) that encodes their meanings in the form of vectors – arrays of numbers. These vectors place words in a multi-dimensional space where similar words are positioned closer together, helping models like transformers understand relationships and context between tokens.

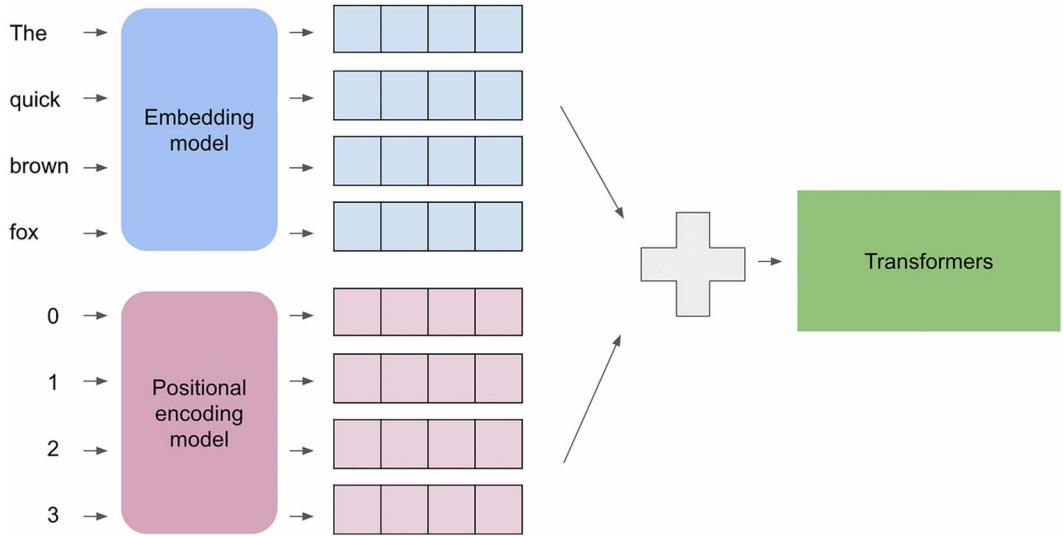


Figure 2. The image illustrates how positional encoding works in transformers. Word embeddings (blue boxes) are created from inputs like “The” and “quick,” while positional information (pink boxes) tracks word order. These are combined and then passed to the transformer model (green box), enabling it to understand word order in sequence processing.

- For odd dimensions ($2i + 1$):

$$PE(\text{pos}, 2i + 1) = \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

Where:

- **pos** is the position index of the token in the sequence.
- **i** is the dimension index.
- **d_model** is the dimensionality of the embeddings.

This formulation allows the model to learn positional relationships because the positional encodings provide unique vectors for each position, and the sinusoidal functions enable the model to generalize to sequences longer than those seen during training.

4.2. Multi-head attention

While self-attention allows the model to consider relationships between tokens, multi-head attention (represented in Figure 3) extends this capability by enabling the model to focus on different positions and representation subspaces (Cordonnier, Loukas & Jaggi, 2021).

1. Multiple attention heads:

Instead of computing attention once, the model uses **h** different attention heads, each with its own set of learned projections:

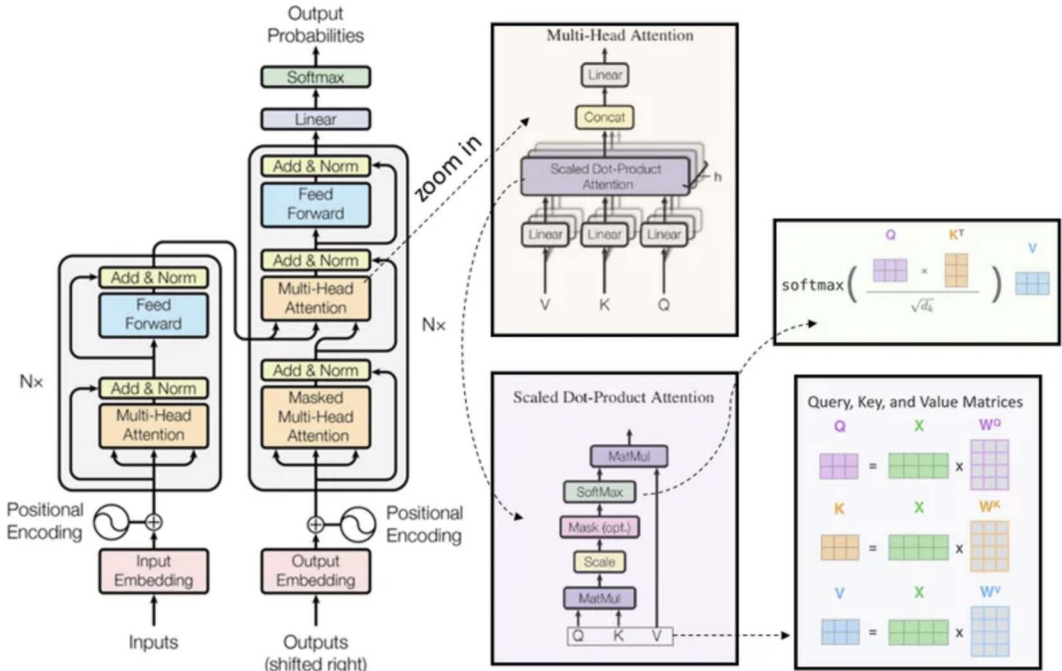


Figure 3. This image represents the core components of the transformer architecture, focusing on the multi-head attention mechanism. The left side shows the stacked layers of multi-head attention and feedforward layers, which are applied both to the input and output sequences. Positional encoding is added to account for word order in the sequence. On the right, a zoom-in reveals how scaled dot-product attention works by combining query, key, and value matrices, normalized through a softmax function, to calculate attention scores. This enables transformers to efficiently capture relationships between words regardless of their position. (Vaswani et al., 2017).

For head i :

$$Q_i = E_{input} \times W_i^q$$

$$K_i = E_{input} \times W_i^k$$

$$V_i = E_{input} \times W_i^v$$

$$head_i = \text{Attention}(Q_i, K_i, V_i)$$

2. Concatenation and output projection:

The outputs from all heads are concatenated and projected to form the final output:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1 \dots \text{head}_h) \times W^O$$

where W^O is the output projection matrix.

By having multiple heads, the model can capture diverse aspects of the input, such as syntax and semantics, and learn different types of relationships.

4.3. Feed-forward networks and residual connections

After the multi-head attention layer, each position undergoes a fully connected **position-wise feed-forward network** (FFN):

$$FFN(x) = \max(0, x W_1 + b_1) \times W_2 + b_2$$

Where:

- W_1 and W_2 are weight matrices.
- b_1 and b_2 are bias vectors.
- $\max(0, x)$ denotes the rectified linear unit activation function.

The FFN is applied independently to each position, allowing the model to transform the attended representations into a higher-level abstraction.

To facilitate training and improve gradient flow, the transformer architecture employs **residual connections** and **layer normalization**:

1. Residual connections:

The input to each sublayer is added to its output:

$$Residual(x) = x + Sublayer(x)$$

2. Layer normalization:

The residual output is normalized to stabilize the training:

$$Output = LayerNorm(Residual(x))$$

These techniques help prevent vanishing or exploding gradients and allow for deeper networks by ensuring that the signal remains strong as it moves through the layers.

5. Overall transformer architecture

The transformer architecture consists of two main components: the encoder and the decoder. This design is particularly effective for sequence-to-sequence tasks like machine translation, where an input sequence in one language is transformed into an output sequence in another.

5.1. Encoder-decoder structure

The encoder-decoder structure, illustrated in Figure 4, is a fundamental mechanism in sequence-to-sequence models designed for tasks such as translation, summarization, and text generation.

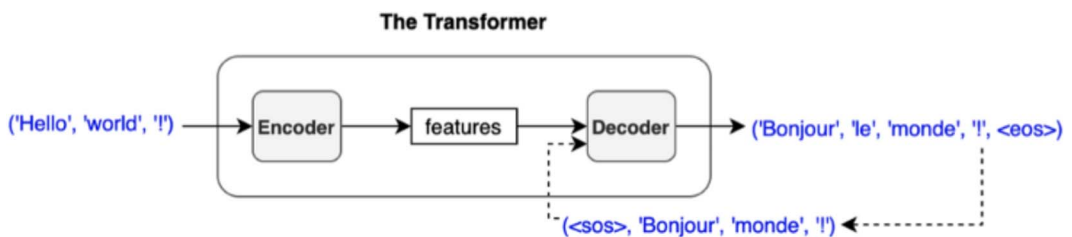


Figure 4. Diagram of an encoder-decoder transformer model demonstrating sequence-to-sequence translation.

The **encoder** processes the input sequence to generate a contextual representation. It begins by converting input tokens into continuous vectors using an embedding layer and adds positional encodings to retain the order of tokens. The encoder is composed of multiple identical layers, each containing a multi-head self-attention mechanism and a position-wise FFN, both followed by residual connections and layer normalization to enhance training stability and gradient flow.

The self-attention mechanism allows each token to attend to all other tokens in the sequence, capturing dependencies regardless of distance. The FFN further transforms these representations, introducing nonlinearity and enabling the model to learn complex patterns.

The **decoder** generates the output sequence by predicting one token at a time, using both the encoder's output and its own previously generated tokens. Like the encoder, it starts with an embedding layer and positional encodings. Each decoder layer includes a masked multi-head self-attention mechanism (to prevent access to future tokens), a multi-head attention mechanism over the encoder's output (allowing focus on relevant parts of the input) and a position-wise FFN, each followed by residual connections and layer normalization.

During the **encoding phase**, the encoder processes the entire input sequence simultaneously, producing encoded representations for each position. In the **decoding phase**, the decoder generates the output sequence step by step. At each step, it considers its own past outputs through masked self-attention and attends to the encoder's output via encoder–decoder attention, enabling it to incorporate information from the input sequence relevant to generating the next token.

5.1.1. Decoder-only transformers

In some applications, only the decoder part of the transformer is used. **Decoder-only transformers**, such as the GPT series, are specialized for tasks involving sequence generation based on prior context, like language modeling and text generation. These models consist solely of decoder layers with masked multi-head self-attention to ensure that predictions depend only on preceding tokens. They are trained to predict the next token in a sequence, making them ideal for tasks like autocomplete and text continuation. A visual representation of this structure, showing an attention word heatmap of a decoder-only architecture, is illustrated in [Figure 5](#).

5.1.2. Encoder-only transformers

Conversely, **encoder-only transformers** consist solely of the encoder stack and are designed for language understanding tasks. Models like BERT utilize this architecture. They employ bidirectional self-attention mechanisms, allowing tokens to attend to both past and future positions, thereby capturing context from the entire sequence. These models are trained using MLM, where some input tokens are masked, and the model learns to predict them based on surrounding context. This approach is effective for tasks such as sentiment analysis, named entity recognition, and question answering.

5.2. Open perspective

Despite their strengths, transformers are not without challenges. The self-attention mechanism, while powerful, requires significant computational resources, particularly in terms of memory. This is because self-attention involves comparing every element in the input sequence with every other element, which scales quadratically with the input length. For very large datasets or long input sequences, this can become prohibitively expensive. However, recent research has been focused on addressing these limitations by developing more efficient variants of transformers, such as sparse transformers and reformers, which aim to reduce the computational load without sacrificing performance. Additionally, quantized models⁷ (Egashira, Vero, Staab, He & Vechev, 2024) further enhance

⁷ A quantized model is a neural network where the precision of the model's parameters (weights and activations) is reduced, typically from floating-point ("fp," typically 32-bit or 16-bit) to lower precision, such as 8-bit or even smaller, like 4-bit. The

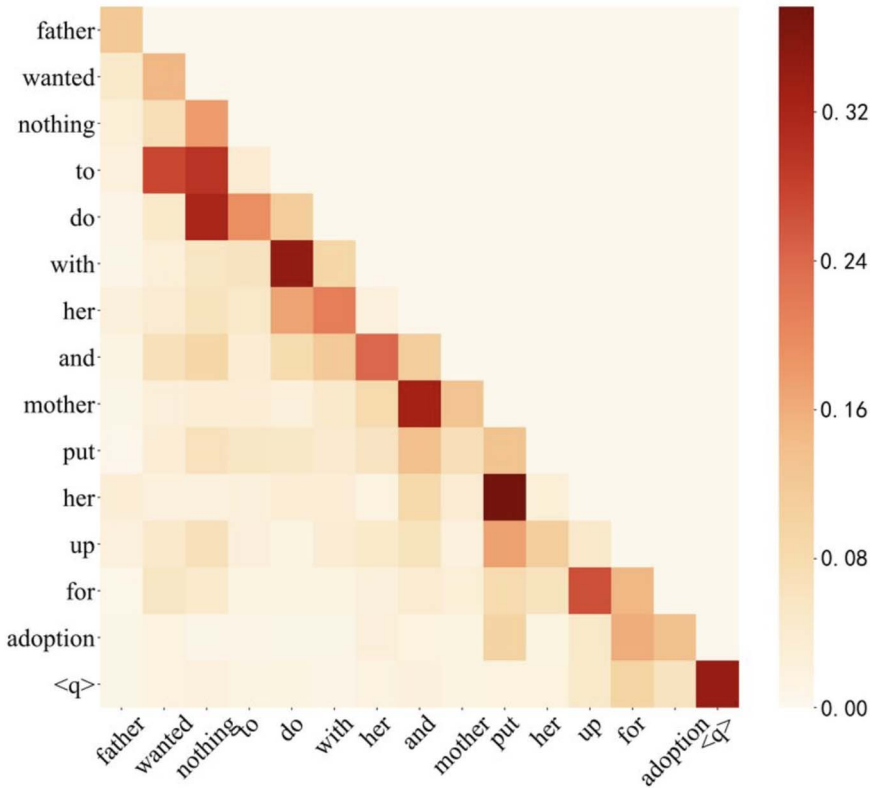


Figure 5. Attention heatmap from a decoder-only model, illustrating how each token in the sequence attends only to itself and previous tokens. The triangular structure results from masked self-attention, ensuring the model generates text autoregressively by relying solely on past context.

efficiency by reducing the precision of model weights (e.g., from 32-bit to 4-bit), allowing significant reductions in memory usage and enabling models to run on smaller hardware without significant performance.

6. LLMs: Scaling transformers to new heights

Building upon the transformative capabilities of the transformer architecture, **LLMs** represent a significant advancement in AI by scaling the core innovations of transformers to unprecedented levels (Naveed *et al.*, 2024). LLMs leverage self-attention mechanisms and extensive training to capture intricate patterns in text, enabling them to perform a wide array of language tasks with remarkable proficiency.

6.1. Scaling laws and model sizes

A critical aspect of LLMs is their scale – in terms of model size, training data quantity and computational resources – which significantly impacts their performance. Research by AI labs and research

goal of quantization is to reduce the memory footprint and computational requirements of the model, enabling faster inference and allowing deployment on resource-constrained hardware like mobile devices or edge systems, without significantly degrading performance. Quantization is especially important for large models, such as LLMs, to optimize them for efficiency while maintaining accuracy.

centers has established **scaling laws** that describe how increasing these factors lead to predictable improvements in model capabilities:

- **Model size (parameters):** LLMs like GPT-3 and GPT-4 contain hundreds of billions of parameters. Increasing the number of parameters allows the model to capture more complex patterns and nuances in language.
- **Data quantity:** Training on larger datasets exposes the model to a broader range of language uses, contexts and knowledge. This diversity enhances the model's ability to generalize across different tasks.
- **Compute resources:** Training large models on vast datasets requires substantial computational power. Advances in hardware (such as GPUs and TPUs) and distributed training techniques have enabled the training of LLMs at this scale.

Scaling laws suggest that as we proportionally scale up model size, data and compute resources, the model's performance continues to improve, often following a power-law relationship. This has motivated the development of ever-larger models to push the boundaries of language understanding and generation. However, it must be noted that as of today, this approach brings with it several challenges and considerations – both economical and ethical – such as the increasing need for expensive computational resources, environmental impact due to the large-scale consumption of electricity. This has led researchers to look into different directions, such as using smaller models (Lu Z. et al, 2024) in combination with use of highly curated and specialized training sets as an alternative to ever growing models (Liu et al., 2024).

6.1.1. Domain-specific small language models

While scaling has driven remarkable achievements in general-purpose language models, recent research has demonstrated the promise of smaller, specialized models trained on domain-specific data. These models, typically ranging from hundreds of millions to a few billion parameters, leverage targeted training data to achieve performance comparable to larger models within their specialized domains (Hsieh et al., 2023; Javaheripi et al., 2023). The efficiency of these models stems from their concentrated focus on domain-specific patterns, terminology and task requirements, effectively reducing the computational overhead associated with maintaining broad language understanding, thus having a significantly reduced environmental impact with parameter counts several orders of magnitude smaller (Schick & Schütze, 2020). This approach has proven particularly valuable in fields such as biomedicine (Gu et al., 2021) and legal document analysis⁸ where domain expertise and precision are necessary. The success of these specialized models suggests that strategic data curation and domain-focused architecture optimization may offer a complementary path to the scaling paradigm (Zhang, Zeng, Wang & Lu, 2024).

6.2. Training techniques

LLMs are typically trained in two stages: **pretraining** (Schneider, Meske & Kuss, 2024; Wang, Li, Wu, Hovy & Sun, 2023; Zhou et al., 2023) and **fine-tuning** (Parthasarathy, Zafar, Khan & Shahid, 2024). During pretraining, the model learns general language representations from vast amounts of text data without explicit supervision (Ding, Qin & Yang et al., 2023). Two primary objectives guide this phase:

1. **Causal language modeling (CLM):** Utilized by models like the GPT series, CLM trains the model to predict the next word in a sequence given all previous words. This unidirectional approach is suitable for generation tasks, where the model maximizes the likelihood of the next word based on the preceding context.

⁸See an example of a practical application: <https://www.personal.ai/pi-ai/legal-ai-the-small-language-model-advantage>.

2. **Masked language modeling (MLM)**: Employed by models such as BERT, MLM involves predicting missing words in a sequence where some tokens are randomly masked. This bidirectional approach allows the model to learn from both left and right contexts, minimizing the prediction error for the masked tokens (Merchant, Rahimtoroghi, Pavlick & Tenney, 2020).

After pre-training, LLMs undergo **fine-tuning** on task-specific datasets to adapt them to particular applications. Fine-tuning can be **supervised**, using labeled data for tasks like question answering, sentiment analysis, or named entity recognition. In cases where labeled data is scarce, **unsupervised fine-tuning** leverages unsupervised objectives to adapt the model to new domains.

Unsupervised learning plays a crucial role in the initial training phase, enabling the model to learn general language patterns from unlabeled data. Supervised learning becomes important during fine-tuning, where the model is taught to perform specific tasks based on labeled datasets.

6.2.1. *Parameter-efficient fine-tuning (PEFT)*

PEFT (Han, Gao, Liu, Zhang & Zhang, 2024; Xu, Xie, Qin, Tao & Wang, 2023) methods have emerged to address the computational challenges associated with fine-tuning massive models with billions of weights (Fu et al., 2023). Instead of updating all weights, techniques like **Low-Rank Adaptation (LoRA)** allow only a small subset of weights to be fine-tuned. **LoRA** (Hu et al., 2021) introduces low-rank matrices to specific layers, adapting the model without altering its full architecture, which significantly reduces memory and computational demands.

Quantized LoRA (QLoRA) combines this approach with quantization, storing model weights in lower-precision formats like 4-bit, further reducing resource needs while maintaining accuracy. **Quantization-aware LoRA (QA-LoRA)** (Xu et al., 2023) goes a step further by applying quantization selectively to critical weight matrices, balancing efficiency and performance in constrained environments. These techniques enable the fine-tuning of large models on smaller hardware, reducing computational overhead without sacrificing precision.

6.3. *Parameter tuning*

While fine-tuning optimizes model architecture for specific applications, parameter tuning offers flexible adjustments to model outputs based on input prompts. This helps tailor responses for characteristics like creativity, precision or length, enhancing task-specific performance without altering the model's structure (Liao, Li, Shang & Ma, 2022).

6.3.1. *Key parameters in LLM tuning*

- **Temperature**: This controls the randomness of the model's output by adjusting the probability distribution of predicted words. Lower temperatures make responses more deterministic, while higher temperatures increase variability, fostering creativity in responses like poetry.
- **Seed**: The seed ensures reproducibility by fixing the random number generator's starting point, making it possible to produce the same outputs for identical inputs across multiple trials – crucial for testing and debugging.
- **Top-k sampling**: This technique restricts the next word prediction to the k most probable words, reducing the risk of the model choosing unlikely or incoherent words. Smaller values of k make the output more focused, enhancing accuracy.
- **Top-p (nucleus sampling)**: A more dynamic approach than Top-k, nucleus sampling selects words whose combined probability exceeds a certain threshold (p), ensuring a balance between diversity and coherence.
- **Max Tokens**: This parameter limits the number of tokens the model generates in response, useful for managing the length of outputs, such as in summarization tasks where brevity is needed.

- **Frequency and Presence Penalty:** Both parameters manage word repetition. Frequency penalties reduce redundancy by discouraging the model from repeating words, while presence penalties further limit the reuse of words that have already appeared in the text.
- **Stop sequences:** These are specific tokens that signal the model to halt text generation, particularly useful for structured tasks or dialogues that need concise responses.
- **Logit bias:** Logit bias allows for direct control over the probability distribution, steering the model toward or away from certain words – vital for ensuring the use of domain-specific terminology or avoiding irrelevant language.

By adjusting these parameters, users can ensure LLMs meet specific needs, whether optimizing for creativity, precision or domain-specific vocabulary. This layer of control complements fine-tuning and provides a powerful toolset for task adaptation, enabling more effective utilization of LLMs across diverse applications.

6.4. Prompt engineering

Prompt engineering is a technique used to optimize the inputs provided to LLMs, ensuring they generate more accurate, relevant and useful outputs (Chen, Zhang, Langrené & Zhu, 2023). A “prompt” in this context refers to the text or instructions given to the model, guiding its response. Unlike methods that alter the model’s architecture or underlying weights, prompt engineering focuses solely on refining the input to influence the output without changing the model itself.

6.4.1. Key concepts in prompt engineering

- **Clarity and specificity:** Well-crafted prompts are clear and specific, reducing ambiguity and leading to more accurate responses.
- **Contextual information:** Providing the right amount of background or context can significantly improve the relevance and coherence of the model’s outputs.
- **Task demonstration (Few-shot learning):** By including examples of the desired task (few-shot learning), the model can generalize better and provide higher-quality responses.

6.4.2. Techniques in prompt engineering

1. **Zero-shot prompting:** The model is expected to generate a response without any examples, relying purely on pre-trained knowledge.
2. **One-shot prompting:** A single example of the task is provided in the prompt, helping the model better understand the format and expected output while still minimizing the number of examples.
3. **Few-shot prompting:** A few examples of the task are included in the prompt, which helps the model understand the format and expected output.
4. **Chain-of-thought (CoT) prompting:** CoT guides the model through a step-by-step reasoning process, which is particularly effective for tasks that require logical progression or complex reasoning. CoT methodologies often have subcategories, such as **Tabular CoT**, which is tailored for handling tasks that involve structured data or tables by applying step-by-step reasoning within tabular formats.
5. **Instruction tuning:** Clear and direct instructions help the model perform specific tasks, such as summarizing or generating lists.
6. **Self-consistency:** This technique involves generating multiple responses for the same prompt and selecting the most consistent one, improving reliability, especially in reasoning tasks.

6.5. Evaluating LLMs performance

Evaluating LLMs is essential to ensuring their accuracy, reliability and fairness across various applications, from healthcare to law. Performance evaluation can be divided into two main categories: human assessments and automated methods.

Human evaluation involves domain experts or users reviewing model outputs for factors like fluency, coherence, relevance and factual accuracy (Feng *et al.*, 2024). This approach is especially critical in fields such as legal (Chalkidis, Fergadiotis, Malakasiotis, Aletras & Androutsopoulos, 2020), financial (Wu *et al.*, 2023) and medical applications (Wang & Zhang, 2024), where nuanced and context-specific knowledge is required. However, human evaluation is labor-intensive and difficult to scale for large volumes of model iterations or outputs.

Automated evaluation methods provide scalable and objective metrics. They measure aspects such as fluency, accuracy and relevance of the text output. Common methods include the following:

- **Bilingual Evaluation Understudy (BLEU)** (Papineni, Roukos, Ward & Zhu, 2002), **Recall-Oriented Understudy for Gisting Evaluation (ROUGE)** (Lin, 2004) and **Metric for Evaluation of Translation with Explicit Ordering (METEOR)** (Lavie & Denkowski, 2009) scores: These assess the quality of text generation (e.g., translation, summarization) by comparing model outputs to reference texts based on content overlap and lexical similarity.
- **Perplexity** (Colla, Delsanto, Agosto, Vitiello & Radicioni, 2022) and **F1 scores** (Zhang, Wang & Zhao, 2015): Perplexity measures how well a language model predicts sequences of text, focusing on fluency. F1 scores, combining precision and recall, are used for classification tasks to evaluate how well the model categorizes or identifies information.
- **Adversarial robustness testing** (Zimmermann, Brendel, Tramer & Carlini, 2022): This method tests how LLMs perform under challenging or adversarial inputs, ensuring that models can handle unexpected or tricky queries without producing incorrect or biased responses.
- **Fairness and bias testing** (Rodolfa, Saleiro & Ghani, 2020): These frameworks measure the ethical performance of models by identifying and mitigating any gender, racial or cultural biases in generated content, ensuring the model outputs are fair and nondiscriminatory.

These evaluation techniques help optimize LLMs for performance while ensuring they meet ethical and reliability standards across various applications.

7. Context windows, hallucinations and other challenges in LLMs

LLMs excel in tasks involving language comprehension and generation, but they are not without limitations. Two of the most prominent challenges are the management of context windows and the issue of hallucinations, among other inherent difficulties in LLMs.

LLMs operate within **fixed context windows** (Dsouza, Glaze, Shin & Sala, 2024), typically from a few thousands to a few hundred thousand tokens. This limitation constrains the amount of text the model can consider at once. In scenarios requiring long-form analysis, like legal reviews or complex conversations, earlier parts of the input might be discarded, leading to a loss of continuity and potentially impacting the quality of the response. While newer models such as GPT-4 have extended context windows, the inherent limitation remains, posing challenges for tasks that demand deep contextual understanding.

Hallucinations (Azamfirei, Kudchadkar & Fackler, 2023) occur when LLMs generate text that is plausible but incorrect or entirely fabricated (Ye, Liu, Zhang, Hua & Jia, 2023). Because LLMs generate predictions based on statistical patterns learned from training data, they might confidently present false information. This is particularly dangerous in critical fields such as healthcare, finance and law, where factual accuracy is essential. Models can invent statistics, references or claims, complicating

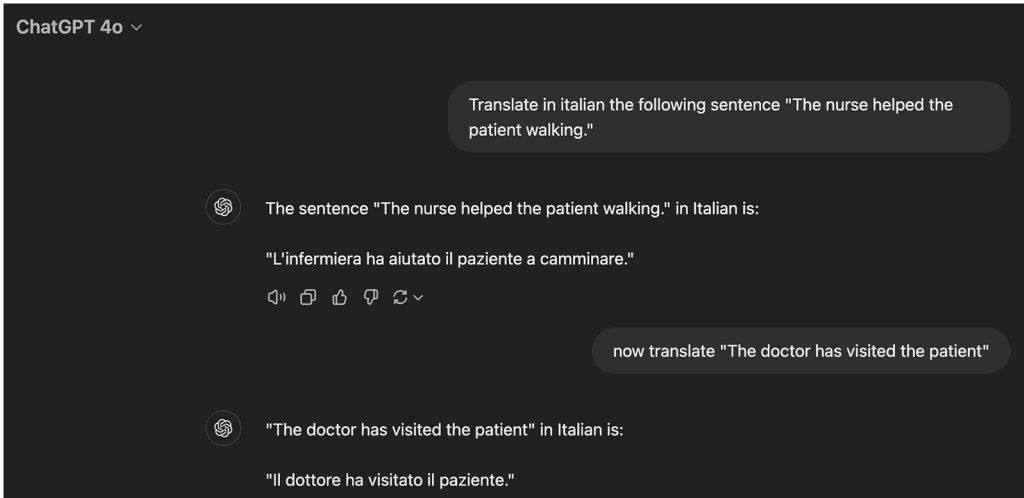


Figure 6. This image demonstrates gender bias in LLMs. The model assumes a female nurse and a male doctor in its translations, reflecting common stereotypes embedded in training data.

the task of trustworthiness. Mitigation strategies include refining training datasets, incorporating real-time knowledge bases and enhancing human oversight during model fine-tuning.

LLMs also struggle with **bias amplification**, as they reflect the biases present in their training data, which can perpetuate harmful stereotypes (illustrated in Figure 6, where an example of gender bias learned by the model is depicted). Additionally, LLMs remain **opaque** in their decision-making processes, making it difficult to interpret how outputs are generated. Finally, **energy consumption** is a growing concern (Samsi et al., 2023), as training large models demands substantial computational resources, raising ethical and environmental considerations.

8. Improving LLM accuracy: Retrieval-augmented generation (RAG)

LLMs have shown remarkable capabilities in natural language understanding and generation. However, as previously presented, they still face limitations, particularly around the accuracy and timeliness of the information they generate. These models are trained on vast datasets but may lack up-to-date or domain-specific knowledge, leading to hallucinations, outdated responses, or factually incorrect outputs. This is where RAG steps in to enhance the accuracy and factual reliability of LLMs.

8.1. Retrieval-augmented generation (RAG)

RAG is an advanced approach that integrates information retrieval systems with LLMs to enhance their accuracy and relevance (Lewis et al., 2020; Li, Su, Cai, Wang & Liu, 2022). RAG models combine the generative capabilities of LLMs with external knowledge sources, such as databases or document collections, enabling the model to pull real-time information rather than relying solely on pretrained data. This mechanism addresses the limitations of LLMs, such as hallucinations and outdated knowledge, by grounding generated responses in retrieved, factual data. A flowchart illustrating the architecture of a RAG model, detailing the interaction between the retrieval and generation components, is shown in Figure 7.

RAG operates in two phases: retrieval and generation. In the retrieval phase, the model searches a vast external knowledge base to gather relevant information based on the input query. In the generation phase, the retrieved data is then used to condition the response, enabling the LLM to

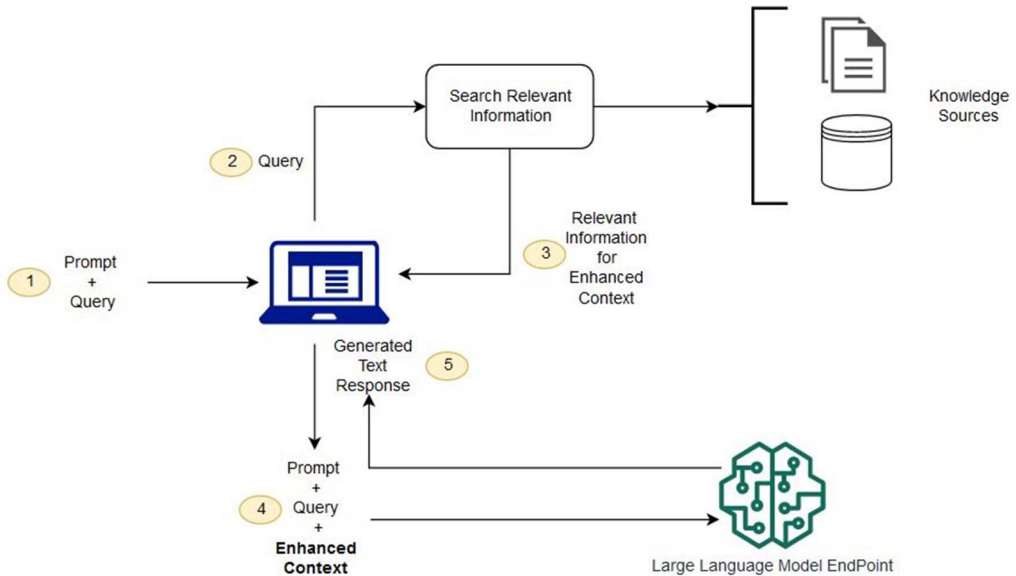


Figure 7. The diagram showcases the workflow of retrieval-augmented generation (RAG). It starts with a prompt and query, which are used to search external knowledge sources for relevant information. The retrieved information enhances the context, which is then passed along with the original query to the large language model (LLM). This improved context helps the LLM generate a more accurate and relevant text response.

provide more accurate and contextually grounded outputs. By integrating retrieval with generation, RAG mitigates the issue of hallucinations, significantly reducing instances of fabricated or inaccurate content.

The ability to retrieve up-to-date information makes RAG particularly effective for dynamic fields such as news reporting, medical diagnostics and legal document analysis, where real-time accuracy is paramount.

8.2. Evaluation of RAG efficacy and metrics

Evaluating the efficacy of RAG models requires both traditional and novel metrics tailored to the retrieval-enhanced framework. Key metrics include the following:

- **Retrieval accuracy:** Ensures that the external knowledge source effectively supplements the LLM, reducing hallucinations and improving factuality. Tools such as Retrieval Augmented Generation Assessment (RAGAs) (Es, James, Espinosa-Anke & Schockaert, 2023) (RAG Automatic Scoring) are emerging to automate this process by evaluating both the retrieval quality and the final generated output.
- **Factuality and groundedness:** A critical metric for RAG models is ensuring that generated responses are factually grounded in the retrieved documents. Evaluation frameworks like Luna (Saidov, Bakalova, Taktasheva, Mikhailov & Artemova, 2024) assess how well the generated text aligns with retrieved facts, helping to reduce inaccuracies and inconsistencies.

9. Conclusions

LLMs have revolutionized NLP by harnessing transformer architectures to achieve unprecedented proficiency in language understanding and generation. They have transformed industries

such as healthcare, law and customer service by enabling applications that require high fluency and precision. Despite these advancements, LLMs face ongoing challenges, including computational resource demands, context window limitations and issues related to bias and factual accuracy.

Innovations like PEFT, quantization techniques and RAG are actively addressing these challenges, enhancing the efficiency, scalability and reliability of LLMs. As these models continue to grow in scale and capability, they hold the promise of extending beyond language tasks to impact fields like computer vision and enable multimodal AI applications.

With a continued focus on improving efficiency and addressing ethical considerations, LLMs are poised to play a pivotal role in shaping the future of technology and AI, driving forward the capabilities of AI systems across a wide array of domains.

Funding statement. This research was supported by the Fondazione CRT, under the 2023 Call, aimed at advancing interdisciplinary studies in the intersection of law, LLMs and policy. The funding body played no role in the design, execution or publication of this work.

Competing interests. The authors declare that they have no competing interests, financial or otherwise, that could have influenced the content or conclusions of this research.

References

- Azamfirei, R., Kudchadkar, S. R., & Fackler, J. (2023). Large language models and the perils of their hallucinations. *J Critical Care*, 27(1), 120.
- Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., & Androutsopoulos, I. (2020). LEGAL-BERT: The Muppets straight out of law school. arXiv. <https://arxiv.org/abs/2010.02559>.
- Chen, B., Zhang, Z., Langrené, N., & Zhu, S. (2023). Unleashing the potential of prompt engineering in large language models: A comprehensive review. arXiv preprint arXiv:2310.14735.
- Chen, P.-C., Tsai, H., Bhojanapalli, S., Chung, H. W., Chang, Y.-W., & Ferng, C.-S. (2021). A simple and effective positional encoding for transformers. arXiv. <https://arxiv.org/abs/2104.08698>.
- Colla, D., Delsanto, M., Agosto, M., Vitiello, B., & Radicioni, D. P. (2022). Semantic coherence markers: The contribution of perplexity metrics. *Artificial Intelligence in Medicine*, 134, 102393.
- Cordonnier, J.-B., Loukas, A., & Jaggi, M. (2021). Multi-head attention: Collaborate instead of concatenate. arXiv. <https://arxiv.org/abs/2006.16362>.
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv. <https://arxiv.org/abs/1810.0480>.
- Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.-M., Chen, W., Yi, J., Zhao, W., Wang, X., Liu, Z., Zheng, H.-T., Chen, J., Liu, Y., Tang, J., Li, J., & Sun, M. (2023). Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3), 220–235. doi:10.1038/s42256-023-00626-4
- Dsouza, A., Glaze, C., Shin, C., & Sala, F. (2024). Evaluating language model context windows: A “Working Memory” test and inference-time correction. arXiv preprint arXiv:2407.03651.
- Egashira, K., Vero, M., Staab, R., He, J., & Vechev, M. (2024). Exploiting LLM quantization. arXiv. <https://arxiv.org/abs/2405.18137>.
- Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023). RAGAS: Automated evaluation of retrieval augmented generation. arXiv preprint arXiv:2309.15217.
- Feng, K., Ding, K., Ma, K., Wang, Z., Zhang, Q., & Chen, H. (2024). Sample-efficient human evaluation of large language models via maximum discrepancy competition. arXiv. <https://arxiv.org/abs/2404.08008>.
- Fu, Z., Yang, H., So, A. M. C., Lam, W., Bing, L., & Collier, N. (2023, June). On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 37, No. 11, 12799–12807).
- Gu, Y., Tinn, R., Cheng, H., Lucas, M., Usuyama, N., Liu, X., ... Poon, H. (2021). Domain-specific language model pre-training for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1), 1–23.
- Han, Z., Gao, C., Liu, J., Zhang, J., & Zhang, S. Q. (2024). Parameter-efficient fine-tuning for large models: A comprehensive survey. arXiv. <https://arxiv.org/abs/2403.14608>
- Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2), 107–116. <https://doi.org/10.1142/S0218488598000094>

- Hsieh, C. Y., Li, C. L., Yeh, C. K., Nakhost, H., Fujii, Y., Ratner, A., ... Pfister, T. (2023). Distilling step-by-step! Outperforming larger language models with less training data and smaller model sizes. arXiv preprint arXiv:2305.02301.
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... Chen, W. (2021). Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.
- Jawaheripi, M., Bubeck, S., Abidin, M., Aneja, J., Bubeck, S., Mendes, C. C. T., ... Gopi, S. (2023). Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1, 3.
- Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., & Reddy, S. (2023). The impact of positional encoding on length generalization in transformers. arXiv. <https://arxiv.org/abs/2305.19466>.
- Lavie, A., & Denkowski, M. J. (2009). The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3), 105–115.
- LeCun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W., & Jackel L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4), 541–551. doi: <https://doi.org/10.1162/neco.1989.1.4.541>
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. <https://doi.org/10.1109/5.726791>
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.
- Li, H., Su, Y., Cai, D., Wang, Y., & Liu, L. (2022). A survey on retrieval-augmented text generation. arXiv preprint arXiv:2202.01110
- Liao, L., Li, H., Shang, W., & Ma, L. (2022). An empirical study of the impact of hyperparameter tuning and model optimization on the performance properties of deep neural networks. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(3), 1–40.
- Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out* (pp. 74–81). Association for Computational Linguistics. <https://aclanthology.org/W04-1013>
- Liu, Y., Cao, J., Liu, C., Ding, K., & Jin, L. (2024). Datasets for large language models: A comprehensive survey. arXiv. <https://arxiv.org/abs/2402.18041>
- Lu, Z., Li, X., Cai, D., Yi, R., Liu, F., Zhang, X., Lane, N. D., & Xu, M. (2024). Small language models: Survey, measurements, and insights. arXiv. <https://arxiv.org/abs/2409.15790>
- Lu, Z., Li, X., Cai, D., Yi, R., Liu, F., Zhang, X., ... Xu, M. (2024). Small language models: Survey, measurements, and insights. arXiv. <https://arxiv.org/abs/2409.15790>.
- McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), 115–133. doi: [10.1007/BF02478259](https://doi.org/10.1007/BF02478259)
- Merchant, A., Rahimtoroghi, E., Pavlick, E., & Tenney, I. (2020). What happens to BERT embeddings during fine-tuning? arXiv preprint arXiv:2004.14448.
- Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., ... Mian, A. (2024). A comprehensive overview of large language models. arXiv. <https://arxiv.org/abs/2307.06435>.
- Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311–318).
- Parthasarathy, V. B., Zafar, A., Khan, A., & Shahid, A. (2024). The ultimate guide to fine-tuning LLMs from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. arXiv. <https://arxiv.org/abs/2408.13296>.
- Rodolfa, K. T., Saleiro, P., & Ghani, R. (2020). Bias and fairness (in Machine Learning). In *Big Data and Social Science: A Practical Guide to Methods and Tools* (pp. 281–312). Chapman and Hall/CRC.
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386–408. doi: [10.1037/h0042519](https://doi.org/10.1037/h0042519)
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation.
- Saidov, M., Bakalova, A., Taktasheva, E., Mikhailov, V., & Artemova, E. (2024). LUNA: A framework for language understanding and naturalness assessment. arXiv preprint arXiv:2401.04522.
- Samsi S., Zhao D., McDonald J., Li B., Michaleas A., Jones M., Bergeron W., Kepner J., Tiwari D., & Gadepally V. (2023). From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–9. [10.1109/HPEC58863.2023.10363447](https://doi.org/10.1109/HPEC58863.2023.10363447).
- Schick, T., & Schütze, H. (2020). It's not just size that matters: Small language models are also few-shot learners. arXiv preprint arXiv:2009.07118.
- Schneider, J., Meske, C., & Kuss, P. (2024). Foundation models. *Business and Information Systems Engineering*, 66(2), 221–231. doi: [10.1007/s12599-024-00851-0](https://doi.org/10.1007/s12599-024-00851-0)
- Tervén, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., Chavez-Urbiola, E. A., & Romero-Gonzalez, J. A. (2024). Loss functions and metrics in deep learning. arXiv. <https://arxiv.org/abs/2307.02694>.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. arXiv. <https://arxiv.org/abs/1706.03762>.

- Wang, D., & Zhang, S.** (2024). Large language models in medical and healthcare fields: Applications, advances, and challenges. *Artificial Intelligence Review*, 57(299). doi:10.1007/s10462-024-10921-0
- Wang, H., Li, J., Wu, H., Hovy, E., & Sun, Y.** (2023). Pre-trained language models and their applications. *Engineering*, 25, 51–65. doi:10.1016/j.eng.2022.04.024
- Werbos P.** (1974). Beyond Regression New Tools for Prediction and Analysis in the Behavioral Sciences, Ph.D. Thesis, Harvard University, Cambridge.
- Wu, S., Irsoy, O., Lu, S., Dabrovolski, V., Dredze, M., Gehrmann, S., & Mann, G.** (2023). BloombergGPT: A large language model for finance. <https://arxiv.org/abs/2303.17564> (Retrieved 10 October 2024).
- Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., & Wang, F. L.** (2023). Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. arXiv. [10.48550/arXiv.2312.12148](https://arxiv.org/abs/10.48550/arXiv.2312.12148)
- Ye, H., Liu, T., Zhang, A., Hua, W., & Jia, W.** (2023). Cognitive mirage: A review of hallucinations in large language models. arXiv preprint arXiv:2309.06794.
- Zhang, D., Wang, J., & Zhao, X.** (2015, September). Estimating the uncertainty of average F1 scores. In *Proceedings of the 2015 International conference on the theory of information retrieval* (pp. 317–320).
- Zhang, P., Zeng, G., Wang, T., & Lu, W.** (2024). Tinyllama: An open-source small language model. arXiv preprint arXiv:2401.02385.
- Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., ... Sun, L.** (2023). A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT. arXiv. <https://arxiv.org/abs/2302.09419>.
- Zimmermann, R. S., Brendel, W., Tramer, F., & Carlini, N.** (2022). Increasing confidence in adversarial robustness evaluations. *Advances in Neural Information Processing Systems*, 35, 13174–13189.

Andrea Filippo Ferraris, PhD Fellow LAST-JD, ALMA AI, University of Bologna and PhD Fellow in Law at DIKE and Law faculty, Vrije Universiteit Brussel, Brussels. Email: andrea.ferraris3@unibo.it and andrea.filippo.ferraris@vub.be

Davide Audrito, PhD Fellow at LAST-JD, Computer Science Department, University of Torino, and Legal Studies Department, University of Bologna. Email: d.audrito@unito.it and davide.audrito2@unibo.it

Luigi Di Caro, Associate Professor of Computer Science, Computer Science Department, University of Torino. Email: luigi.dicaro@unito.it

Cristina Poncibò, Full Professor of Comparative Law, Department of Law, University of Turin. Email: cristina.poncibo@unito.it