



**Università
di Brescia**

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

**DOTTORATO DI RICERCA IN INGEGNERIA
DELL'INFORMAZIONE**

ING-INF/05

CICLO XXXVIII

UNDERSTANDING AND APPLYING LLMs:
FROM FIGURATIVE LANGUAGE TO
INDUSTRIAL APPLICATIONS

Dottorando:
Nicola Arici

Supervisore:
Prof. Alfonso E. Gerevini
Co-Supervisore:
Prof. Ivan Serina

*A chi oggi posso chiamare
compagno, amico, famiglia, casa*

Abstract

Understanding what knowledge Transformer-based Large Language Models (LLMs) acquire during pretraining and how such models can be reliably integrated into production systems are both central challenges for Natural Language Processing. This thesis develops along these complementary research directions: the first studies how semantic abilities emerge during pretraining, with a focus on figurative language; the second investigates the integration of Transformer-based language models into real-world workflows within the learning management system developed by Mega Italia Media S.p.A, which co-funded my PhD scholarship.

For the first direction, we study the acquisition of figurative language knowledge through the Minimum Description Length (MDL) probing framework, evaluating multiple checkpoints saved throughout training, and extending the analysis with a layer-wise study and controlled experiments about the impact of training data. Experimental results indicate that figurative language is not learned as a single uniform capability: different figures of speech exhibit different acquisition dynamics. Moreover, most of the relevant knowledge is acquired early during pretraining and then stabilizes, producing a plateau where additional training yields limited improvements. Layer-wise analyses further suggest that the representations which retain more of the figurative knowledge are typically found in the central part or in the second half of Generative Pre-Trained Transformers (GPT). Finally, controlled comparisons suggest that larger and more diverse corpora are beneficial, while a literary-focused corpus does not provide measurable improvements over a general-purpose corpus for figurative language recognition.

For the applied direction, this thesis investigates two industrial use cases developed within Mega Italia Media. The first addresses the automatic assessment of open-ended learner responses, motivated by the need to reduce the manual grading effort while preserving human evaluation in compliance-oriented health and safety training courses. The second focuses on automatic ticket assignment, motivated by the need to optimize the initial handling of support requests by reducing the workload of support coordinators and improving the consistency of routing decisions, with the goal of shortening response times and increasing the overall efficiency of the support workflow. Together, the two systems are part of a coordinated effort to improve the learning experience of learners using the Mega Italia Media platform, while simultaneously optimizing internal processes and operational costs for company staff.

Abstract

Comprendere quali conoscenze i Large Language Models (LLM) basati su architetture Transformer acquisiscano durante la fase di addestramento, e come tali modelli possano essere integrati in modo affidabile in sistemi di produzione, rappresenta una duplice sfida centrale per il Natural Language Processing (NLP). La presente tesi si sviluppa lungo queste direttrici di ricerca complementari: la prima analizza le modalità con cui emergono le competenze semantiche nel corso dell’addestramento, con particolare attenzione al linguaggio figurato; la seconda indaga l’integrazione di modelli linguistici basati su Transformer in flussi di lavoro reali, all’interno del Learning Management System (LMS) sviluppato da Mega Italia Media S.p.A., società che ha cofinanziato la mia borsa di dottorato.

Per quanto riguarda la prima direttrice, viene studiata l’acquisizione della conoscenza del linguaggio figurato attraverso il framework di probing basato sul Minimum Description Length (MDL), valutando diversi checkpoint salvati nel corso dell’addestramento ed estendendo l’analisi mediante uno studio per livelli e tramite esperimenti controllati sull’impatto dei dati di training. I risultati sperimentali indicano che il linguaggio figurato non viene appreso come una capacità unica e omogenea: differenti figure retoriche mostrano dinamiche di acquisizione eterogenee. Inoltre, la maggior parte delle conoscenze rilevanti viene acquisita nelle fasi iniziali dell’addestramento e successivamente tende a stabilizzarsi, dando luogo a un plateau oltre il quale un ulteriore addestramento produce benefici marginali. L’analisi per livelli suggerisce inoltre che le rappresentazioni che preservano una maggiore quantità di informazione relativa al linguaggio figurato si collocano tipicamente nella parte centrale o nella seconda metà del modello GPT. Infine, confronti controllati indicano che corpora di dimensioni maggiori e caratterizzati da una maggiore eterogeneità risultano vantaggiosi, mentre un corpus a prevalente contenuto letterario non produce miglioramenti misurabili rispetto a un corpus generalista per il riconoscimento del linguaggio figurato.

Per quanto concerne la direttrice applicativa, la tesi analizza due casi d’uso industriali sviluppati all’interno di Mega Italia Media. Il primo riguarda la valutazione automatica di risposte aperte fornite dagli utenti in contesti formativi, ed è motivato dall’esigenza di ridurre il carico di correzione manuale mantenendo al contempo un presidio umano, in conformità con i requisiti normativi dei corsi di formazione in materia di salute e sicurezza sul lavoro. Il secondo caso d’uso è incentrato sull’assegnazione automatica dei ticket di assistenza, con l’obiettivo di ottimizzare la gestione iniziale delle richieste

di supporto, riducendo il carico operativo dei coordinatori e migliorando la coerenza delle decisioni di instradamento, al fine di abbreviare i tempi di risposta e incrementare l'efficienza complessiva del flusso di supporto. Nel loro insieme, i due sistemi si inseriscono in un'iniziativa coordinata volta a migliorare l'esperienza formativa degli utenti della piattaforma Mega Italia Media, ottimizzando al contempo i processi interni e i costi operativi per il personale aziendale.

Contents

1	Introduction	2
1.1	Topic of the Thesis	2
1.2	Structure of the Thesis	4
2	State of the Art	5
2.1	Word Embedding	6
2.2	The Transformer Architecture	9
2.3	Bidirectional Encoder Representations from Transformers	16
2.4	Generative Pre-Trained Transformers	19
2.5	Probing Techniques	23
3	Understanding LLMs	30
3.1	Methodology	31
3.2	Language Models and Tasks	34
3.3	Experimental Results	37
3.3.1	Learning Trajectories of Figurative Language	39
3.3.2	Layer-Wise Distribution of Figurative Language Knowledge	46
3.3.3	Impact of Training Data	53
4	Implementing LLMs	58
4.1	Automatic Scoring for Workplace Safety Courses	59
4.1.1	Dataset and Task Formulation	60
4.1.2	Model Architecture	63
4.1.3	Experimental Results	65
4.1.4	Real World Implementation	67
4.2	LLM-based Ticket Assignment	70
4.2.1	Task and Data	71
4.2.2	Methodology	74
4.2.3	Experimental Results	76
5	Conclusion	80

Chapter 1

Introduction

This thesis illustrates the study and work carried out at the Dipartimento di Ingegneria dell'Informazione of Università degli Studi di Brescia during my Ph.D. in Information Engineering (ING-INF/05), cycle XXXVIII. My work was conducted under the supervision of Prof. Alfonso E. Gerevini and Prof. Ivan Serina. The doctoral scholarship supporting this research was co-funded by Mega Italia Media S.p.A., an Italian company that has operated since 1988 in the domain of corporate training, with a strong specialization in workplace health and safety. The company develops and commercializes a wide catalog of eLearning courses and provides digital services and tools to support training delivery at scale.

1.1 Topic of the Thesis

Suppose that we read a single sentence and we have to decide whether its meaning should be interpreted literally or figuratively. For instance, the sentence “*I would give my life for a coffee.*” is a natural example of figurative language (hyperbole), while a superficially similar sentence such as “*I have lived in this country all my life.*” expresses a literal meaning. This kind of binary decision, although simple to state, requires a model to capture semantic properties that depend on plausibility, context, and abstract meaning.

Large Language Models (LLMs) achieve state-of-the-art performance across many Natural Language Processing (NLP) tasks, but the internal mechanisms supporting this performance remain opaque. This lack of interpretability raises concerns about robustness, reliability, and safety, and motivates the broader scientific question of whether these models truly understand language or primarily exploit statistical regularities at scale. In this thesis, we address this problem by leveraging the Minimum Description Length (MDL) (Voita and Titov, 2020) probing technique to evaluate how much knowledge LLMs encode about semantic tasks, with a particular focus on the detection of figurative language in the form of figures of speech.

The focus on semantics is motivated by the observation that semantic interpretation often requires models to integrate long-distance dependencies, resolve ambiguity, and construct abstract meaningful representations, and is therefore more challenging than

recovering morphological or syntactic patterns. For this reason, probing semantic phenomena can reveal useful insights into what neural language models encode, how this knowledge emerges during training, and where it is represented across model layers. The semantic phenomena investigated in this thesis include four figures of speech: hyperbole, metaphor, oxymoron, and pleonasm.

To study how linguistic knowledge is acquired, we adopt a trajectory-based perspective. Given a linguistic property, we trace the so-called learning trajectory by probing multiple subsequent checkpoints saved during pre-training, applying a consistent methodology at each checkpoint. This approach makes it possible to determine whether a property is learned consistently, reaches a plateau or exhibits signs of regression after a certain number of training iterations, or begins to emerge only after sufficient training. Beyond trajectory analysis, we also perform layer-wise probing, associating each layer of the Transformer stack with a probe, in order to analyze how knowledge is distributed across the network.

The experimental study is grounded in a controlled setting that supports systematic comparisons across checkpoints and model scales. In particular, we rely on the Pythia model suite, which provides models of different sizes trained on the same data in the same order, together with a large set of intermediate checkpoints released during training. This enables a detailed investigation of figurative language acquisition across training and scaling.

In addition to understanding what LLMs encode, this thesis also addresses the problem of applying language models in concrete operational settings. This originates from industrial collaborations with Mega Italia Media S.p.A., an Italian company operating in the domain of corporate training with a strong specialization in workplace health and safety, and developing a large catalog of eLearning courses and digital tools to support training delivery at scale. A central element is DynDevice LMS, Mega Italia Media's Learning Management System (LMS), conceived to support the creation, delivery, and documentation of corporate training across heterogeneous modalities, including asynchronous eLearning, classroom sessions, and synchronous web conferences. DynDevice was initially released in 2011 and has since evolved through continuous updates into a modular framework that supports course authoring, training record management, and integrations with complementary organizational services. The platform is positioned to satisfy compliance-oriented requirements typical of regulated training contexts and explicitly targets end-to-end traceability of learning activities, including enrollment workflows, reporting, and document archiving.

In this context, we present two systems. The first addresses the automatic assessment of open-ended learner answers within mandatory course examinations, with the objective

of reducing manual grading effort while maintaining reliable evaluation standards and complete auditability of outcomes. The second addresses the automatic classification of customer assistance tickets, with the objective of improving response times and routing accuracy in technical support pipelines.

1.2 Structure of the Thesis

In this section, we provide a brief overview of the structure of the thesis and summarize the content of each chapter.

In Chapter 2, we present a structured overview of the methodological foundations underlying modern neural language models and the analytical tools used to study their internal representations. The chapter introduces distributed representations of language, the Transformer architecture, encoder-based models such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), decoder-based autoregressive models such as Generative Pre-Trained Transformers (GPT) (Radford et al., 2018), and the motivation for probing techniques. Finally, the chapter introduces the Minimum Description Length (MDL) (Voita and Titov, 2020) principle as a framework for evaluating probes by jointly accounting for predictive performance and model complexity.

Chapter 3 focuses on understanding LLMs through a probing-based methodology. We introduce the complete probing framework adopted in this thesis, define the semantic tasks used for analysis, and study how figurative language knowledge emerges during training by tracing learning trajectories across pre-training checkpoints. The chapter also includes a layer-wise analysis to investigate how figurative language knowledge is distributed across Transformer layers, and an analysis of how different pre-training corpora influence the acquisition of figurative language.

In Chapter 4, we describe two industrial systems developed in collaboration with Mega Italia Media. The first system addresses automatic scoring of open-ended answers in regulated workplace safety training contexts, and is designed as a human-in-the-loop tool that supports evaluators while preserving traceability and auditability. The second system addresses automatic ticket assignment in customer support workflows, exploring prompting-based strategies for routing tickets to the appropriate internal macro-area.

Finally, Chapter 5 concludes the thesis by summarizing the main findings of the probing study and the industrial case studies, and discussing limitations and directions for future work.

Chapter 2

State of the Art

This chapter presents a structured overview of the methodological foundations underlying modern neural language models and the analytical tools used to study their internal representations. Rather than providing an exhaustive historical survey, the chapter introduces the key conceptual and architectural developments that motivate the empirical analyses presented in the remainder of this thesis.

The discussion begins with distributed representations of language, starting from static word embeddings. Early embedding models demonstrated that lexical semantic regularities could be captured through dense vector representations learned from large corpora, thereby establishing distributional semantics as a central paradigm in natural language processing. However, such representations are inherently context-independent, which limits their ability to model polysemy, compositional meaning, and long-range dependencies.

A major shift in representation learning was introduced with the Transformer architecture. By replacing recurrence with self-attention mechanisms, Transformers enabled scalable modeling of global contextual interactions and facilitated the training of deep architectures on large-scale datasets. This architectural innovation laid the foundation for the development of modern pre-trained language models.

Within this framework, encoder-based models such as BERT ([Devlin et al., 2019](#)) and decoder-based autoregressive models such as GPT ([Radford et al., 2018](#)) represent two complementary design paradigms. BERT introduced bidirectional contextualization through masked language modeling, producing representations particularly effective for token-level and sentence-level understanding tasks. GPT models, in contrast, emphasize autoregressive generation and have demonstrated that large-scale pre-training alone can yield strong zero-shot and few-shot generalization capabilities.

As these models increased in scale and complexity, understanding which linguistic properties they encode and how such information is distributed across layers became a central research problem. This need motivated the development of probing techniques, which cast linguistic analysis as a supervised prediction problem over internal representations. Probing methods have been widely adopted to investigate syntactic, semantic, and pragmatic properties, but they have also raised methodological concerns related to

probe expressiveness, memorization effects, and the interpretability of results.

To address these limitations, several new approaches have been proposed. Among them, the MDL principle (Voita and Titov, 2020) provides a principled framework for evaluating probes by jointly accounting for predictive performance and model complexity. MDL-based probing enables more robust comparisons across models, layers, and training stages, and supports the analysis of learning dynamics rather than static end-point evaluations.

2.1 Word Embedding

The first problem faced when applying deep learning techniques to NLP tasks is how to represent words in a format suitable for neural networks, such as numbers, vectors, or matrices.

The first NLP systems relied on **one-hot encoding**, where each word in the vocabulary was represented by a sparse vector with a single active element. Given a set of words of size K , the one-hot encoding for the i -th word is a binary vector of size K with 1 in the i -th position and 0 in all the others. Figure 2.1 reports an example for the sentence “*The cat chased the mouse, while the tiger watched silently*”, with $K = 10$. As shown, this encoding guarantees that each word has a unique representation.

However, the one-hot encoding has several limitations. Each word is represented by a sparse vector whose length equals the vocabulary size, containing a single active element and zeros elsewhere. This results in extremely high-dimensional and memory-inefficient representations, particularly for large corpora. Moreover, one-hot vectors are orthogonal, meaning that they capture no semantic or syntactic relationships between words; for example, the vectors for *cat* and *tiger* are as dissimilar as those for *cat* and *mouse*. Consequently, models must learn all the linguistic similarities entirely from the data instead of relying on the representation itself. The approach also lacks flexibility, as new or unseen words cannot be encoded without redefining the entire vocabulary. Finally, because of their sparsity, one-hot vectors lead to computational inefficiency when used as input to neural networks.

These drawbacks motivated the development of dense word embeddings, which provide low-dimensional continuous representations that preserve semantic similarity and improve learning efficiency. The first major breakthrough came with **Word2Vec** Mikolov et al. (2013). Unlike one-hot encoding, which is high-dimensional and does not take into account semantics, Word2Vec produces dense, continuous word vectors that capture both syntactic and semantic relationships. These representations are learned from raw text data by predicting word co-occurrence patterns, enabling the model to

The cat chased the mouse, while the tiger watched silently

The	0	0	0	0	0	0	0	0	0
cat	0	1	0	0	0	0	0	0	0
chased	0	0	1	0	0	0	0	0	0
the	0	0	0	1	0	0	0	0	0
mouse	0	0	0	0	1	0	0	0	0
while	0	0	0	0	0	1	0	0	0
the	0	0	0	0	0	0	1	0	0
tiger	0	0	0	0	0	0	0	1	0
watched	0	0	0	0	0	0	0	0	1
silently	0	0	0	0	0	0	0	0	0

Figure 2.1: One-hot encoding for vocabulary of $K = 10$ words. Each word is represented by a K -dimensional binary vector, with a single active element (value 1) corresponding to the word's index and all other elements set to 0.

infer meaningful relationships between words based on their distributional context.

Word2Vec is grounded in the principle that words appearing in similar contexts tend to have similar meanings. It implements this idea through two lightweight architectures: the **Continuous Bag of Words (CBOW)** and the **Skip-gram** models. As shown in Figure 2.2, in CBOW the model predicts a target word given its surrounding context words, effectively learning to compress contextual information into a single vector representation. Conversely, the Skip-gram model reverses this task, predicting the context words given a central word.

The network consists of three main layers: an input layer, a single projection (embedding) layer, and an output layer. The input layer receives the context words, typically within a symmetric window around the target word. Each word selects its corresponding word embedding from a shared embedding matrix $W \in \mathbb{R}^{V \times D}$, where V is the vocabulary size and D the embedding dimension. The embedding vector produced by the projection layer is then fed directly into the output layer, which is a softmax classifier that estimates the probability distribution over all possible target words.

The original implementation was trained on large corpora, such as the Google News

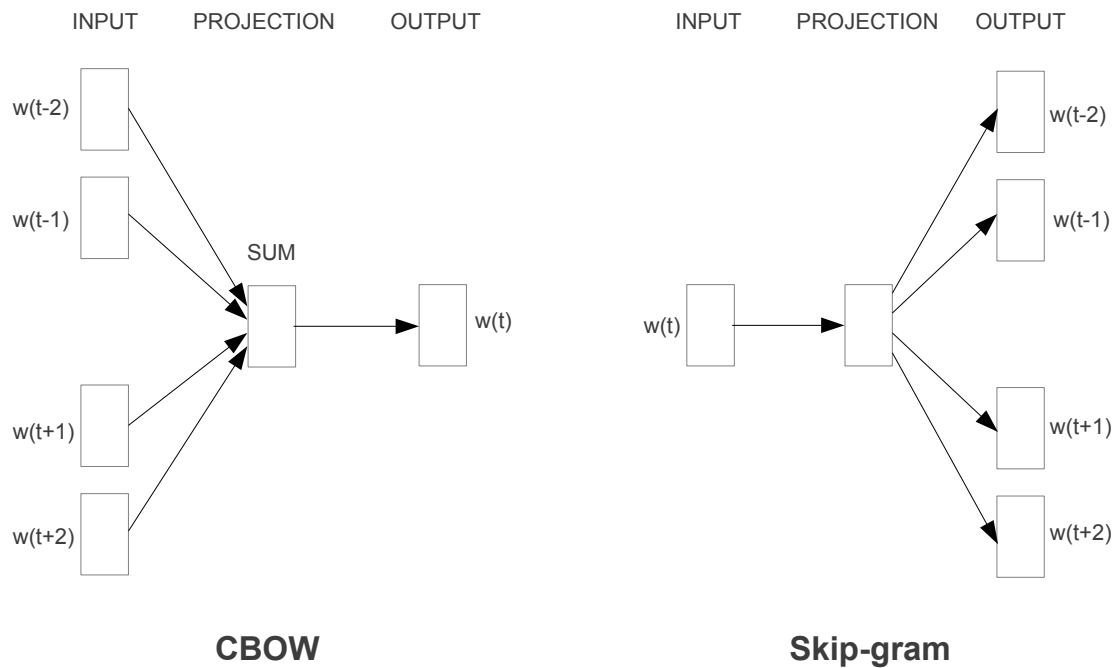


Figure 2.2: Architectures of the CBOW and Skip-gram models introduced in Word2Vec. On the left, the CBOW model predicts the current target word $w(t)$ from the embeddings of surrounding context words within a fixed window. Conversely, on the right, the Skip-gram model inverts this process by using the target word $w(t)$ to predict its neighboring context words. Adapted from Mikolov et al. (2013)

dataset containing approximately six billion tokens, with a vocabulary size of one million words. Typical hyperparameter configurations included an embedding dimension D between 300 and 1000, a context window size of 4–8 words on each side. Training was performed using stochastic gradient descent with a linearly decaying learning rate starting at 0.025, and three to five epochs over the corpus were generally sufficient to achieve convergence.

One of the most influential findings of Word2Vec was that word vectors exhibit linear regularities in the semantic space. Vector arithmetic operations, such as

$$\text{vector}(\text{king}) - \text{vector}(\text{man}) + \text{vector}(\text{woman}) \approx \text{vector}(\text{queen})$$

revealed that the learned embeddings encode abstract relationships, including analogies (country and capital, gender pairs) and grammatical transformations (adjective to adverb, singular to plural).

Although Word2Vec represented a breakthrough in distributed word representations, it was not without limitations. Its embeddings are **context independent**, meaning

that each word is assigned a single static vector regardless of its usage. As a result, the model cannot distinguish among different senses of polysemous words such as *bank* (the riverbank vs. the financial institution) or capture context-dependent nuances. Furthermore, Word2Vec relies on local context windows, learning relationships primarily from co-occurrences within a few neighboring words; this limits its ability to encode long-range dependencies.

These limitations inspired the next major evolution in representation learning: contextual embeddings. Contextual models, beginning with Embeddings from Language Models (ELMo) (Peters et al., 2018), introduced context-sensitive representations by deriving word vectors from internal states of deep bidirectional Recurrent Neural Network (RNN) trained on language modeling tasks. Each word's embedding varies according to its position and neighboring words in a sentence, effectively modeling meaning in context.

Building on this foundation, Transformer-based architectures such as BERT (Devlin et al., 2019) and GPT (Radford et al., 2018) further extended contextualization through self-attention mechanisms that consider all words in a sequence simultaneously. These models not only resolve polysemy but also capture complex syntactic and semantic relationships over long distances, marking a decisive shift from static to dynamic language representations.

2.2 The Transformer Architecture

The Transformer is a deep learning architecture developed by Google in 2017 and proposed in "Attention is All You Need" by Vaswani et al. (2017). As the title of the work states, the fundamental building block is Attention, in particular the newly presented *Self-Attention*. At a high level, if we treat the Transformer as a black-box model, given an input sentence in a language (e.g. the Italian sentence "Sono uno studente di dottorato"), the architecture produces an output sequence in another language (e.g. in English "I am a PhD student"). If we examine the model in more detail, we observe that its architecture consists of two main components: the **Encoder stack** and the **Decoder stack**. Each stack is built by repeating a single type of layer multiple times (six repetitions in the original paper, with subsequent architectures employing even more), the Encoder block and the Decoder block. In the architecture, the number of encoders and decoders is the same. In Figure 2.3 the original architecture presented in Vaswani et al. (2017) is reported.

Embedding

The first step in every Neural Language Model (NLM) is the transformation of words into numerical representations. As seen in Section 2.1, each word is first divided into tokens

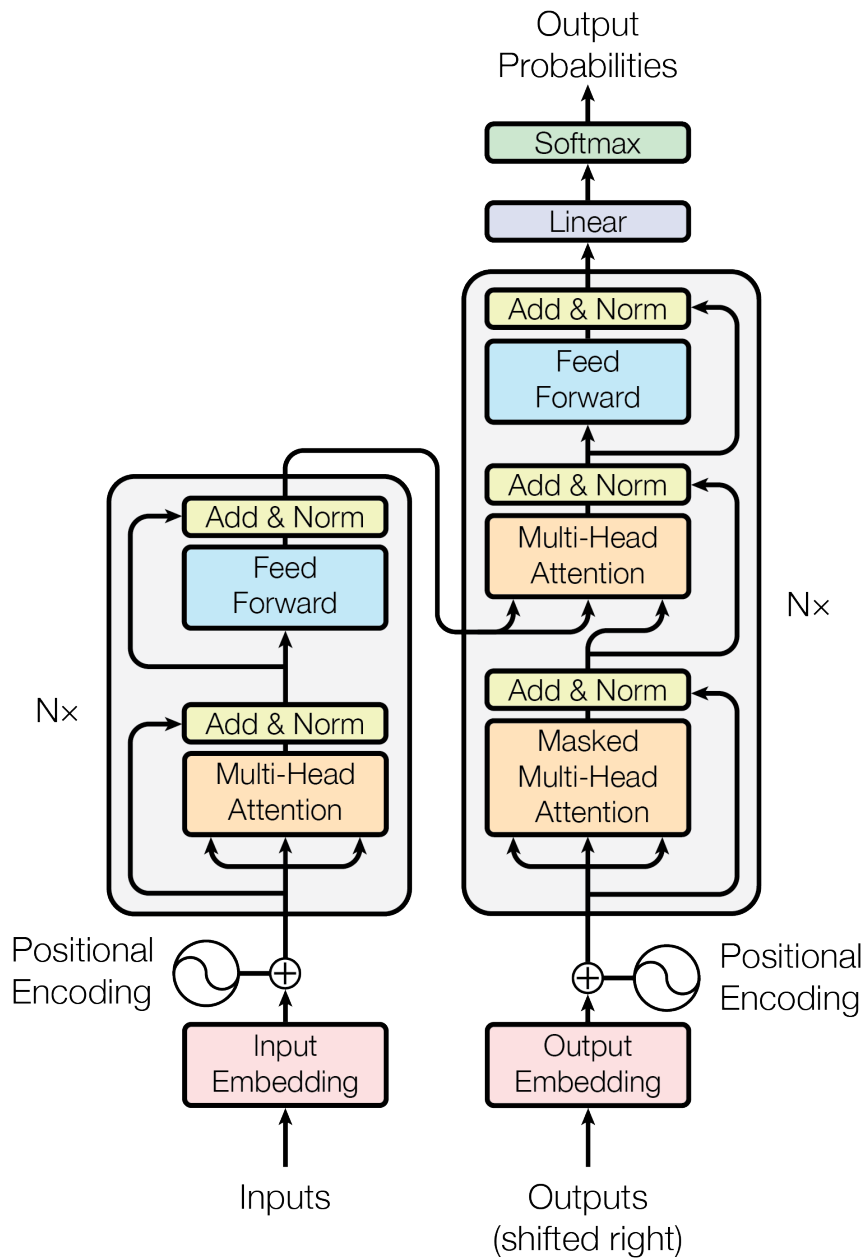


Figure 2.3: Overview of the original Transformer architecture. The **encoder stack** (left) consists of N identical layers, each containing a multi-head self-attention mechanism followed by a position-wise feed-forward network, with residual connections and layer normalization applied after each sub-layer. The **decoder stack** (right) follows a similar structure but includes an additional masked multi-head attention layer to ensure autoregressive generation. Input embeddings are processed at the bottom of each stack, while the decoder output is passed through a linear layer and softmax function to generate the final probability distribution. Adapted from Vaswani et al. (2017).

and then each token is mapped to a corresponding embedding vector. This mapping is defined by the embedding matrix $W \in \mathbb{R}^{v \times d}$, where v denotes the vocabulary size (i.e. the total number of tokens in the vocabulary) and d is the embedding dimension (set to 512

in the original Transformer model). The embeddings of a given input sequence of length n are arranged into a matrix $X \in \mathbb{R}^{n \times d}$, with each row corresponding to the embedding of a token.

Unlike earlier NLP architectures such as RNNs or Convolutional Neural Networks (CNNs), the Transformer does not inherently encode sequential order. To address this limitation, the author introduce **Positional Encoding**, a specific vector added to each token embedding before it enters the encoder stack. In the original formulation, the positional encoding is fixed and set to:

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{(2i/d)}}\right) \quad PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{(2i/d)}}\right) \quad (2.1)$$

where $pos \in [0, n-1]$ is the token position in the sequence and $i \in [0, \lfloor \frac{d}{2} \rfloor]$ indexes the embedding dimensions. The resulting vectors, obtained by summing the embeddings with their corresponding positional encoding, form the input to the first encoder layer.

Encoder

As shown in Figure 2.3, each encoder block in the Transformer consists of two main components: a *Multi-Head Attention* mechanism and a *Feed Forward Network*. The first layer, also known as *self-attention*, is the core of the architecture, enabling the model to capture the interaction of words within a sentence or a document regardless of their distance.

Consider an input sequence of n token embeddings x_i with $i \in [0, n-1]$. For each token, the model computes three distinct vector representations: the *query* (q_i), the *key* (k_i) and the *value* (v_i). These are obtained through learned linear transformations using the weight matrices W_q , W_k , and W_v :

$$q_i = x_i W_q, \quad k_i = x_i W_k, \quad v_i = x_i W_v, \quad (2.2)$$

where $W_q, W_k, W_v \in \mathbb{R}^{d \times b}$, with b an hyperparameter set to 64 in the original implementation.

The self-attention mechanism then computes, for a candidate token x_c , a set of *attention scores* α_{ic} that measures the relevance of each token x_i with respect to x_c . These scores are obtained by the dot product between the query vector of the candidate token q_c and the key vectors k_i of all tokens:

$$\alpha_{ic} = \sigma\left(\frac{q_c k_i^T}{\sqrt{b}}\right), \quad (2.3)$$

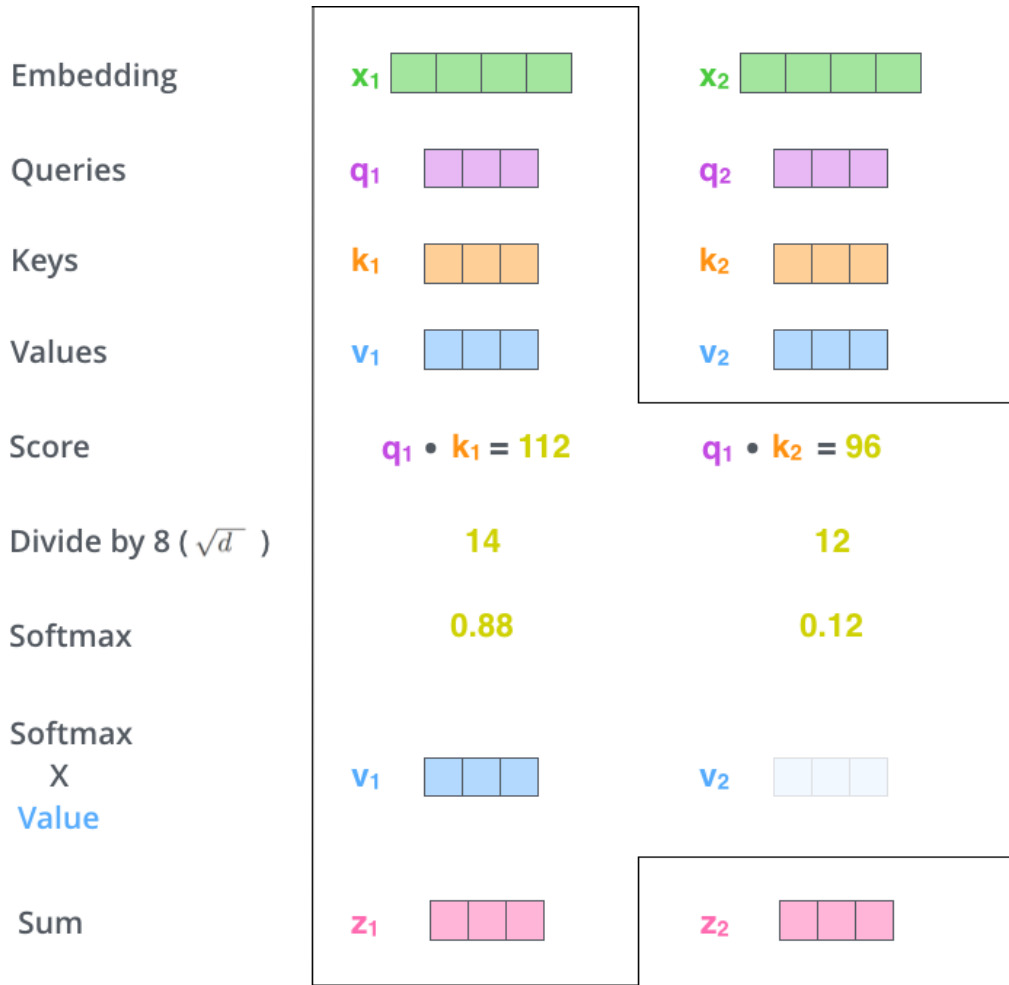


Figure 2.4: Illustration of a single self-attention step for a two-token sequence, focusing on the computation of the output representation z_1 for the first embedding x_1 . The query vector q_1 is compared with the key vectors (k_1, k_2) through dot products to produce attention scores, which are then scaled by $\sqrt{d}$ and normalized via the softmax function. The resulting attention weights are applied to the value vectors (v_1, v_2) to obtain the weighted sum that defines the output z_1 . Adapted from <https://jalammar.github.io/illustrated-transformer/>.

where σ denotes the softmax function, which normalizes the scores into a probability distribution. The scaling factor $\sqrt{b}$ prevents the dot products from growing too large as the dimensionality increases.

Finally, the final candidate representation z_c is computed as the weighted sum, through the attention scores α_{ic} , of the value vectors v_i :

$$z_c = \sum_{i=0}^{n-1} \alpha_i v_i \quad (2.4)$$

This process is illustrated in Figure 2.4, which shows the computation of z_1 for a two-token sequence.

To make computation more efficient, the self-attention mechanism is implemented using matrix operations. Let $X \in \mathbb{R}^{n \times d}$ be the matrix of stacked input embeddings. Then, using the same matrices W_k , W_q , and W_v , we can redefine Equation (2.2) as:

$$Q = XW_q \quad K = XW_k \quad V = XW_v \quad (2.5)$$

where $Q, K, V \in \mathbb{R}^{n \times b}$. Furthermore, we can combine Equation (2.3) and Equation (2.4) as:

$$Z = \sigma\left(\frac{QK^T}{\sqrt{d}}\right)V \quad (2.6)$$

to obtain a more compact definition for the final representation $Z \in \mathbb{R}^{n \times b}$ computed by the self-attention layer. This matrix formulation allows all tokens to attend to each other simultaneously in a single step, leveraging parallel computation, and making Transformer highly efficient.

As shown in Figure 2.3, the original Transformer architecture expands the self-attention mechanism by introducing *Multi-Head Attention*. Specifically, instead of learning a single set of projection matrices, the model learns h independent sets of parameters:

$$W_i^Q \in \mathbb{R}^{d \times b_k}, \quad W_i^K \in \mathbb{R}^{d \times b_k}, \quad W_i^V \in \mathbb{R}^{d \times b_v}, \quad (2.7)$$

for each head $i \in \{1, \dots, h\}$. Each head applies scaled dot-product attention independently:

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \quad (2.8)$$

The outputs of all heads are then concatenated and projected back to the model dimension through a final matrix $W^O \in \mathbb{R}^{h \cdot b_v \times d}$:

$$Z = \text{concat}(head_1, \dots, head_h)W^O. \quad (2.9)$$

In the base Transformer model, the number of heads was set to $h = 8$, with $d_k = d_v = 64$ so that the total dimension $d_{\text{model}} = 512$ is evenly divided between heads (Vaswani et al., 2017). The only requirement for h is that it divides d_{model} , ensuring that each head works with a consistent lower-dimensional representation. An example with a 2-token sentence and 8 heads is reported in Figure 2.5.

This final representation Z is given to a simple fully connected Feed Forward Network (FFN), which is applied to each position separately and identically. This consists of two

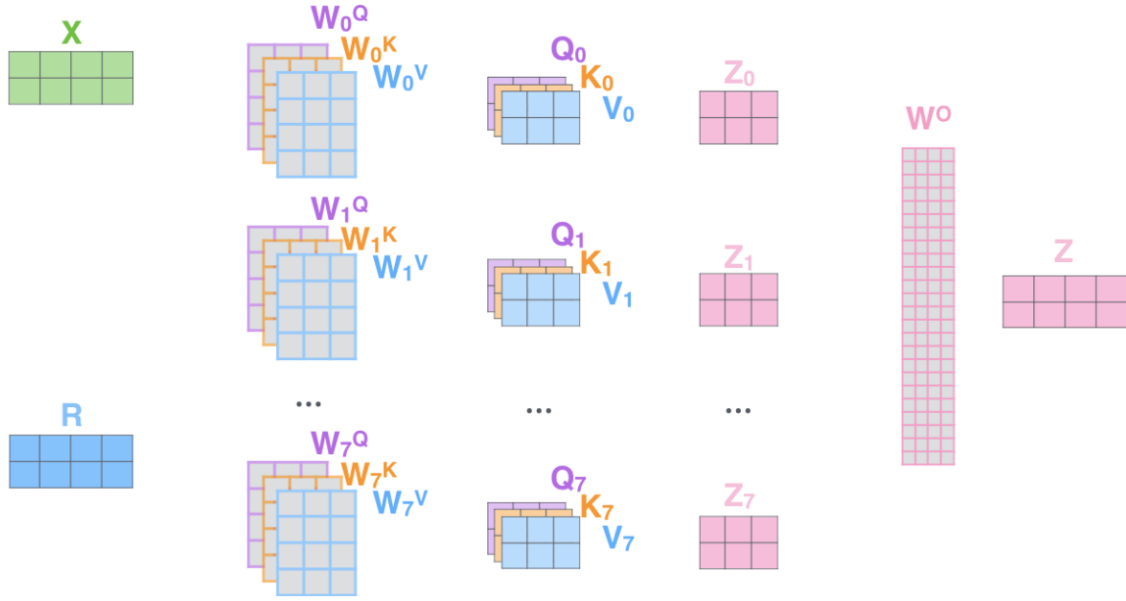


Figure 2.5: Multi-Head Attention mechanism with $h = 8$ heads for a $n = 2$ tokens sentence. The input matrix X is multiplied for each set of head weights matrices W_i^Q, W_i^K, W_i^V to obtain the query, key and value matrices Q_i, K_i, V_i . Adapted from <https://jalammar.github.io/illustrated-transformer/>.

linear transformations with a ReLU activation in between.

$$FFN(x) = \max(0, xW_1 + b_q)W_2 + b_2 \quad (2.10)$$

In the original implementation, the FFN hidden dimension is set to 2048.

The final piece of the encoder is *the residual connection*. After each layer, an operation of Add and Norm is performed to stabilize the output and preserve the information. Given $SubLayer(X)$, the output of one of the two encoder layers, the Add & Norm module performs $LayerNorm(X + Sublayer(X))$, where $LayerNorm$ is the normalization presented in Ba et al. (2016).

The encoder output matrix is multiplied by two weight matrices $W^{k_{encdec}}$ and $W^{v_{encdec}}$ to produce two new representations, namely K_{encdec} and V_{encdec} . These two representations are additional input to each decoder block.

Decoder

The decoder stack in the original Transformer architecture proposed by Vaswani et al. (2017) mirrors the encoder in structure, but introduces an additional crucial component that links the two halves of the model. Each decoder block comprises three layers: *Masked Multi-Head Self-Attention*, an *Encoder–Decoder Multi-Head Attention* layer, and a *Feed-Forward Network*. These inputs are the outputs of the previous decoding steps

(i.e. the generated sentence so far), and for the first iteration, the input is a special *start-of-sentence* token (the original implementation does not define a standard token, the most used ones are $\langle \text{sos} \rangle$, $\langle \text{bos} \rangle$ or just $\langle \text{s} \rangle$). Like the encoder stack, before entering into the decoders, the positional encoding is added to the input.

The first layer is a Masked Multi-Head Self-Attention mechanism. It allows the decoder to attend to previously generated tokens while preventing access to future positions through a causal mask, ensuring that the prediction at position i can depend only on positions $j \leq i$. Formally, the self-attention for the decoder can be expressed as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V \quad (2.11)$$

where M is a masking matrix with $-\infty$ in the positions that correspond to future tokens.

The second layer performs Multi-Head Attention. Here, the query matrix Q originates from the output of the first (masked) attention layer, while the key and value matrices (K_{encdec} and V_{encdec}) are derived from the encoder output. This step enables the decoder to attend to relevant parts of the source sequence when generating each target token.

The final layer is a position-wise FFN, identical to that used in the encoder. Residual connections and layer normalization are applied around each of the three layers in the same way as described above for the encoder.

After the final decoder layer, a linear projection followed by a softmax operation transforms the decoder output vectors into probability distributions over the vocabulary. The linear layer maps the decoder output to a logit vector of dimension V , where V is the vocabulary size. The softmax function converts these logits into probabilities, from which the token with the highest probability is selected as the next output. The process continues until the model generates the special end-of-sentence token ($\langle \text{eos} \rangle$).

The original Transformer model was trained on translation tasks on the WMT 2014 English-German and English-French datasets. The English-German corpus contained approximately 4.5 million sentence pairs with a shared source-target vocabulary of 37000 tokens, while the French-English corpus consisted of 36 million sentence pairs with 32000 tokens.

Transformer established a new performance standard for neural machine translation. In the WMT 2014 English–German task, the Transformer model achieved a BLEU score of 28.4, exceeding the best previously reported systems, including recurrent and convolutional ensembles, by more than 2 BLEU points. On the English–French benchmark, it attained a BLEU score of 41.8, the highest result for any single model at the time.

These gains were achieved with a fraction of the computational cost. The base model trained for approximately 12 hours on eight NVIDIA P100 GPUs, while comparable RNN

or CNN architectures required multiple weeks of training and extensive model ensembling. By discarding recurrence and relying solely on multi-head attention, the Transformer eliminated sequential dependencies in computation, enabling full parallelization and substantially faster convergence.

The combination of improved accuracy, reduced training cost, and architectural elegance marked a decisive shift in sequence modeling research and established the foundation for the modern generation of large-scale attention-based language models.

2.3 Bidirectional Encoder Representations from Transformers

From the Transformer architecture, two principal model families have emerged: the encoder-only model and the decoder-only model. The best representative for the first group is BERT by [Devlin et al. \(2019\)](#). This model leverages only the encoder stack of the Transformer architecture (the left block in Figure 2.3), and introduces several key ideas that outperform the previous NLM in eleven NLP benchmarks, nine of which belong to the GLUE benchmark ([Wang et al., 2018](#)) and two to question answering tasks.

This was obtained through a new technique called **fine-tuning**, heavily inspired by transfer learning. In the first place, BERT is pretrained on a generic unlabeled corpus with generic linguistic tasks. Then, a task-specific classification head is attached to the model, in most cases a small neural network, and the training continues over a small labeled task-specific dataset for fewer epochs. All parameters, both the new task-specific head and the pre-trained BERT weights, are updated jointly during supervised training. Since BERT, this became a standard practice in the Natural Language Processing field.

The major breakthroughs are in the pretraining procedure that is based on two main objectives: the **Masked Language Modeling (MLM)** task and the **Next Sentence Prediction (NSP)** task. The first randomly masks a subset of input tokens and trains the model to predict the original tokens based on their bidirectional context, enabling it to capture rich contextual dependencies. On the other hand, the second task is designed to help the model learn inter-sentence relationships by predicting whether two sentences occur consecutively in the corpus.

To perform these tasks, the authors introduce a set of special tokens: (i) [CLS], introduced as the first token of each input sequence, (ii) [SEP], used as a separator between sentences and at the end of the sentence, and (iii) [MASK] used in the MLM task to hide the real token from the model. The special tokens are treated as words and thus their representation are trained during the pre-training procedure.

Furthermore, BERT pre-training does not require labeled data. For the MLM task the

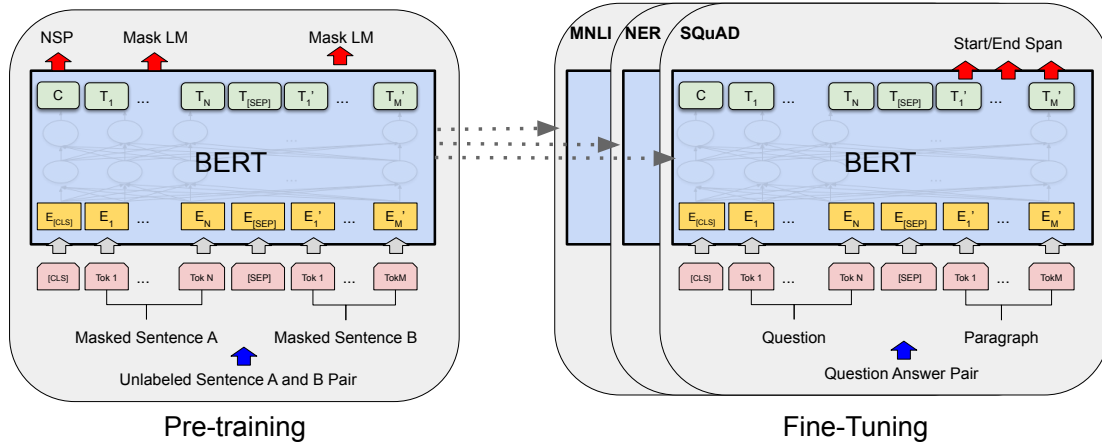


Figure 2.6: Illustration of the dual-phase BERT framework showing the pre-training and fine-tuning processes. During pre-training (left), BERT learns bidirectional language representations through two unsupervised tasks: Masked Language Modeling and Next Sentence Prediction, applied to unlabeled sentence pairs. In the fine-tuning phase (right), the same pre-trained architecture is adapted to downstream tasks by updating all model parameters with task-specific input and output layers. Adapted from [Devlin et al. \(2019\)](#)

masked tokens becomes the label that the model has to predict; for the NSP task the label is automatically derived from whether the algorithm takes two consecutive sentences or not. Thus, it was possible to train BERT on larger corpora, in particular over *BookCorpus* and *English Wikipedia*, with approximately 3B words combined.

As shown in the left part of Figure 2.6, the two tasks are performed simultaneously on the same input sequence. Given two tokenized sentences, first, a $[\text{CLS}]$ token is added as the first token of the first sentence, and a $[\text{SEP}]$ token is added at the end of each sentence, before their concatenation. Then, each token of the sequence is masked with a 15% probability; of those selected tokens, 80% are masked with the $[\text{MASK}]$ token, 10% are replaced with a random token, and 10% are left unchanged. Once the Transformer encoder has processed the sequence, the $[\text{CLS}]$ token representation is fed into a simple neural classifier, composed of a projection layer and a softmax, and the sentence consecutiveness is predicted.

Formally, for each input sentence pair (S_1, S_2) , made up of the sequence of tokens $t = (t_1, t_2, \dots, t_n)$, let M be the random subset of masked tokens M for the MLM task, and $y \in \{0, 1\}$, where $y = 1$ if S_2 follows S_1 , and $y = 0$ otherwise, denote the NSP label.

The MLM loss is defined as:

$$\mathcal{L}_{\text{MLM}} = - \sum_{i \in M} \log P_{\theta}(w_i | \tilde{w}) \quad (2.12)$$

where $\tilde{w}$ denotes the corrupted input sequence and $P_{\theta}(w_i | \tilde{w})$ is the predicted probability

of the true token w_i at position i . Meanwhile, the NSP loss is computed as:

$$\mathcal{L}_{\text{NSP}} = -\left[y \log P_{\theta}(y = 1 \mid S_1, S_2) + (1 - y) \log P_{\theta}(y = 0 \mid S_1, S_2)\right] \quad (2.13)$$

where $P_{\theta}(y \mid S_1, S_2)$ represents the softmax probabilities of the NSP classification task over the [CLS] token representation. Finally, the complete BERT pre-training objective combines Equation (2.12) and Equation (2.13) as:

$$\mathcal{L}_{\text{BERT}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{NSP}} \quad (2.14)$$

The original architecture was trained for 1M steps using batches of 256 sentence pairs of 512 tokens each for approximately 4 days on 16 TPUs. The authors released two models: BERT_{BASE} (12 encoders, 768 hidden dimension, and 12 attention heads) with 110M parameters, and BERT_{LARGE} (24 layers, 1024 hidden dimension, and 16 heads) with 340M parameters.

The real strength of BERT lies in its versatility in solving a multitude of NLP tasks, ranging from token classification, such as Name Entity Recognition (NER), sentence classification, for example sentiment analysis, or more complex tasks such as question answering.

As illustrated on the right side of Figure 2.6, fine-tuning enables BERT to be adapted efficiently to a wide range of downstream tasks. Once a pre-trained BERT encoder is combined with a lightweight classification head, the entire architecture can be optimized end-to-end using only a relatively small amount of task-specific supervision. In contrast to previous neural architectures that required substantial annotated corpora and careful feature engineering, BERT typically achieves strong performance with minimal data and modest training budgets. Empirically, the original authors show that fine-tuning all model parameters for approximately ten epochs on as few as 1K labeled sentences is often sufficient to surpass the state of the art established by earlier NLP models. This demonstrates not only the robustness of the representations learned during the pre-training but also the practical efficiency of the BERT transfer-learning paradigm for downstream adaptation.

However, in recent years, the focus has shifted to the development of large generative models, which have proven to have superior capabilities to decoder-based models, even in solving classification tasks.

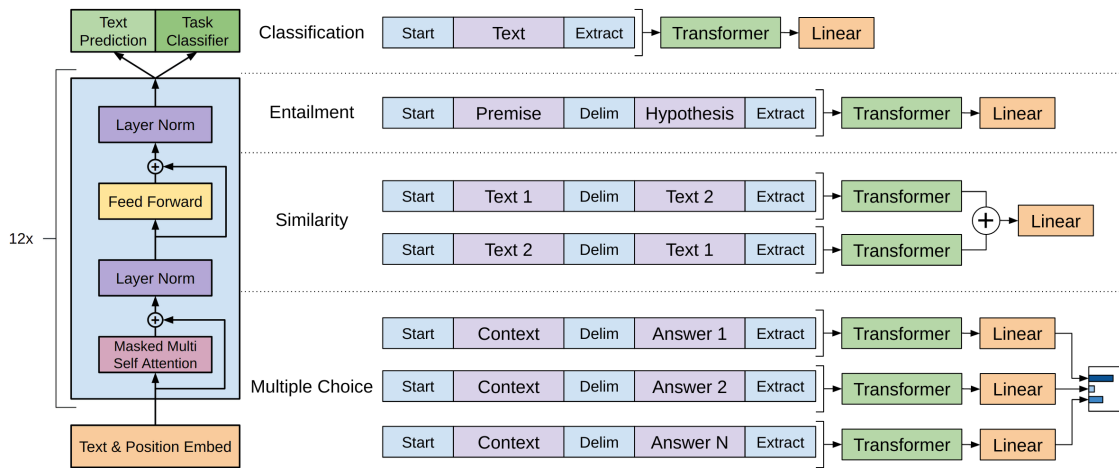


Figure 2.7: Overview of the GPT architecture and its task-specific input transformations. The left panel illustrates the 12-layer decoder block used during pre-training, composed of masked multi-head self-attention, position-wise feed-forward networks, residual connections, and layer normalization. The model is first trained with a left-to-right language modeling objective to acquire general-purpose linguistic representations. The right panel shows how the same pre-trained model is adapted to downstream tasks through minimal textual input modification. Only a lightweight linear layer is added on top of the final hidden representation for each task (the green block on top of the stack), while all Transformer parameters are fine-tuned end-to-end. Adapted from [Radford et al. \(2018\)](#)

2.4 Generative Pre-Trained Transformers

The first implementation of Generative Pre-Trained Transformers introduced by OpenAI researchers in [Radford et al. \(2018\)](#) adapts the original Transformer design to autoregressive language modeling. Unlike the encoder–decoder structure of the original Transformer architecture ([Vaswani et al., 2017](#)), GPT employs only the decoder stack, enforcing a strictly left-to-right flow of information. As shown in Figure 2.7, GPT consists of a multi-layer Transformer decoder composed of masked multi-head self-attention, position-wise feed-forward networks, residual connections, and layer normalization. In contrast to the original Transformer decoder, since there is no encoder, the cross-attention layer is removed. The resulting architecture contains more than 110M parameters.

The training paradigm is divided into a **generative pre-training** phase and a **discriminative fine-tuning** phase. During pre-training, the model is optimized with a standard left-to-right language modeling objective on long-form text. This process allows the model to acquire broad lexical, syntactic, and semantic knowledge from unlabeled data.

In detail, the pre-training procedure is conducted for 100 epochs on the BookCorpus dataset ([Zhu et al., 2015](#)), a collection of thousands of unpublished novels that provides long, contiguous text segments. This characteristic is crucial because the left-to-right

Transformer architecture benefits from extended narrative context. Such continuity allows the model to capture phenomena like coreference resolution, topic progression, and long-distance dependencies that would be absent in shuffled or sentence-level corpora. During training, the model receives 64 sequences of 512 tokens per batch and learns to predict the next token in an autoregressive way. Formally, given an unsupervised corpus of tokens $U = u_1, \dots, u_n$, the model learns to maximize the following likelihood language modeling objective:

$$L_1(U) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta) \quad (2.15)$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ .

Fine-tuning then adapts the pre-trained model to a target task by introducing minimal task-specific parameters, typically a linear output layer and minimal textual input modification; some examples are reported in the right part of Figure 2.7. For example, in the multiple-choice setting, the original context is concatenated separately with each candidate answer using a delimiter token, forming multiple input sequences that are processed independently by the model, whose resulting scores are then compared to select the most likely answer. This strategy preserves the architecture across tasks and enables effective transfer. With a small fine-tuning, GPT outperformed the previous model in a multitude of tasks, ranging across language inference, question answering, common reasoning, semantic similarity, and classification.

Following these results, the next year the same research group presented the second generation of Generative Pre-Trained Transformers (GPT-2) in (Radford et al., 2019). It introduced several substantial improvements over the first GPT-1, particularly in terms of scale, data diversity, and the model's ability to perform a wide range of tasks in a zero-shot setting. GPT-2 increased the model capacity by more than an order of magnitude, scaling up to 48 decoders and $1.5B$ parameters.

A critical advance was the introduction of the **WebText** dataset, a large-scale collection of text (approximately 40GB of data) extracted from outbound Reddit links that reached a quality threshold. Compared to BooksCorpus used for GPT-1 (only 4GB of data), WebText offered substantially broader topical and stylistic coverage, encouraging the model to acquire robust general-purpose linguistic knowledge.

GPT-2 also demonstrated, for the first time, a strong **unsupervised multitask** behavior. Without any explicit fine-tuning, the model learned to perform tasks such as reading comprehension, summarization, translation, commonsense reasoning, and question answering. Its performance improved logarithmically with scale and approached or exceeded specialized supervised baselines on several benchmarks, including language modeling and inference tasks. This zero-shot capability emerged naturally from the

generative objective, suggesting that language models internalize not only grammar and semantics but also procedural task patterns from naturally occurring text.

Across standard language modeling and understanding benchmarks, GPT-2 achieves consistent and often substantial improvements over prior models. On language inference-style evaluations such as the Children’s Book Test, accuracy on common-noun prediction increases by roughly 7–8% points, while named-entity prediction improves by about 6–7% points, closing much of the gap with human performance.

The third generation of Generative Pre-Trained Transformers (GPT-3) introduces substantial advancements that extend the capabilities demonstrated by GPT-2, primarily through substantial scaling in model size, data diversity, and the emergence of robust in-context learning. The most significant improvement is the expansion of the model capacity to 175B parameters. This scaling leads to consistent gains across language modeling metrics and downstream tasks, showing that larger autoregressive Transformers can leverage scale as an implicit source of task generalization.

Furthermore, GPT-3 replaces supervised fine-tuning with in-context learning, implemented through three prompting strategies:

- **Zero-shot:** only natural language description of the task;
- **One-shot:** a single demonstration in the prompt;
- **Few-shot:** several demonstrations provided as examples.

With this approach, GPT-3 outperforms earlier models by large margins, even without any task-specific fine-tuning. For example, few-shot accuracy on TriviaQA (Joshi et al., 2017) reaches 71.2%, surpassing fine-tuned baselines operating in the same closed-book setting. On LAMBADA, few-shot performance increases from GPT-2’s 63% to 86.4%. Similar improvements occur in reading comprehension and in translation tasks.

These results were achieved also thanks to a significantly expanded and more carefully curated training data. GPT-3 was trained on filtered CommonCrawl, with over 45TB of raw data (approximately 450GB of clean and filtered text), resulting in a broad, multilingual and stylistically diverse corpus. This diversity enables the model to perform competitively in multilingual settings.

The introduction of GPT-3 marked a turning point in the development of LLMs. Its demonstration of robust in-context learning, task-agnostic prompting, and substantial performance gains achieved solely through scale established a new paradigm for model design. GPT-3 showed that a single autoregressive Transformer, trained on sufficiently large and diverse data, could generalize across tasks without architectural modification or supervised fine-tuning.

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



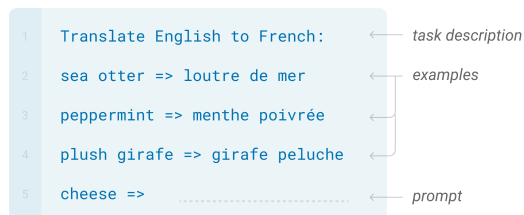
One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.



Figure 2.8: Comparison of GPT-3 in-context learning (left) and traditional fine-tuning (right). The left panel illustrates zero-shot, one-shot, and few-shot settings, where the model relies exclusively on natural-language task descriptions and a small number of exemplars without performing any gradient updates. The right panel depicts standard fine-tuning, in which model parameters are repeatedly updated using supervised examples. Adapted from [Brown et al. \(2020a\)](#).

Subsequent generations such as GPT-4 and GPT-5 build upon these foundations with improvements in reasoning ability, robustness, and alignment, while introducing a multimodal approach, integrating images and sound.

The influence of GPT-3 rapidly extended beyond its own implementation. Its core principles, and the shift from open to closed source, inspired the creation of several large-scale open models, including GPT-NeoX ([Black et al., 2022](#)), and the LLaMA family ([Touvron et al., 2023](#)), all of which adopt the same decoder-only Transformer architecture and rely on massive datasets to achieve broad generalization.

2.5 Probing Techniques

Probing techniques in NLP were developed to interpret and analyze the internal representations of neural language models. As these models encode a wide range of linguistic information in distributed and opaque hidden states, probing classifiers offer a way to assess which properties are captured by such representations. This is achieved by training auxiliary classifiers to predict specific linguistic attributes from internal activations (Belinkov, 2022; Belinkov and Glass, 2019), providing a practical framework for investigating the representational content of neural architectures. The appeal of probing originated from its ability to provide fine-grained measurements of representational content that were not accessible through standard evaluation metrics. By training auxiliary classifiers on frozen model representations, researchers are able to study syntactic, semantic, and morphological phenomena at different layers and across different architectures, contributing to a more detailed understanding of how modern NLP systems generalize linguistically.

Early work focused on static word embeddings, where simple classifiers were trained to predict morphological, syntactic, and semantic attributes (Gupta et al., 2015; Köhn, 2015). These studies demonstrated that embeddings captured fine-grained linguistic properties beyond what downstream performance alone could reveal.

In the following years, probing became a stable methodological paradigm, extensively applied to fixed sentence embeddings, such as Word2Vec (Adi et al., 2017; Conneau et al., 2018). This line of work established robust empirical findings, including the hierarchical organization of linguistic information: lower layers tend to encode local features, whereas higher layers capture increasingly abstract properties. Similar results were achieved in studies focused on Transformer-based architectures, such as BERT (Miaschi et al., 2020; Vig and Belinkov, 2019).

Despite its impact, probing also exhibited notable methodological limitations. As represented in Figure 2.9, Two main limitations were discovered: (i) probes can achieve strong accuracy leveraging representations provided by randomly initialized untrained language models, and (ii) even if the probe is trained to detect a random set of labels.

First, Zhang and Bowman (2018) show that probing results must be interpreted with caution. When the probing classifier is trained on the full auxiliary dataset, even representations produced by randomly initialized and untrained LSTMs yield surprisingly strong performance on Part-of-Speech (POS) tagging. This suggests that a probe may sometimes rely on superficial regularities in the data, rather than exclusively revealing linguistic information encoded by the model. To investigate this issue more directly, the authors introduce a word-identity prediction task, in which the probe must recover the

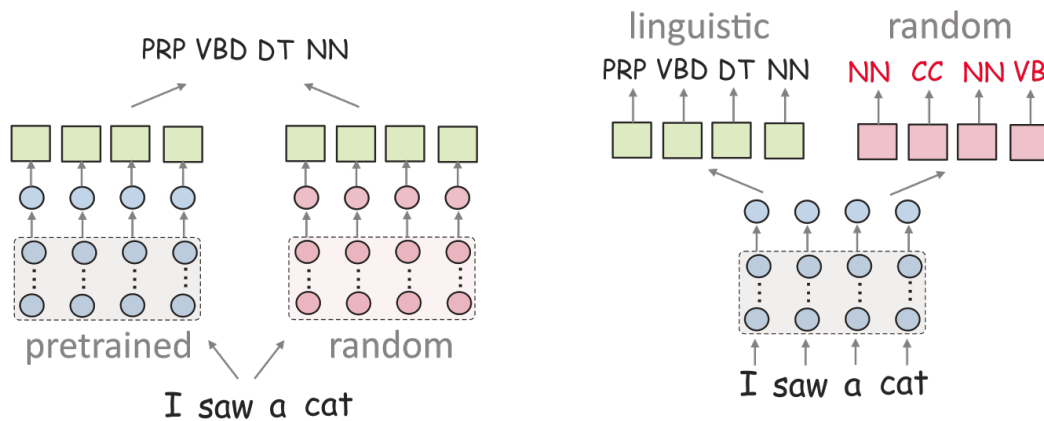


Figure 2.9: Two key limitations of probing using a Part-of-Speech tagging example. Circles denote the language model (blue for a pretrained model, pink for a randomly initialized one), while squares denote the probe (green when trained on correct linguistic labels, red when trained on randomized labels). On the left, probes can also perform well on untrained, random representations, suggesting that probe performance may reflect the classifier’s capacity rather than meaningful information encoded by the pretrained model (Zhang and Bowman, 2018). On the right, probes can achieve high accuracy even when trained on random labels, indicating that they may simply memorize superficial patterns rather than uncover genuine linguistic structure (Hewitt and Liang, 2019). Adapted from https://lena-voita.github.io/posts/mdl_probes.html

exact lexical identity of neighboring words from the hidden states of an LSTM encoder. Interestingly, probes built on top of randomly initialized LSTMs outperform those based on fully trained encoders on this task. As a consequence, high probe accuracy in this task reflects the probe’s capacity to memorize correlations in the representation space rather than evidence that the underlying encoder-model has learned syntactic abstractions.

To address this, the authors propose reducing the size of the probing dataset as a diagnostic method. When the probing training set is small, memorization becomes difficult, and probes reveal clearer distinctions between trained and untrained representations. Furthermore, to fully validate the results of the investigation, they propose to compare the probes applied to trained models with the probes applied to random or untrained baselines, ensuring that the differences in performance reflect the true linguistic signal rather than the expressivity of the probe.

The following year, Hewitt and Liang (2019) show, through controlled experiments on ELMo representations, that highly expressive Multi Layer Perceptron (MLP) probes can achieve strong performance on POS tagging while also performing comparably on synthetic control tasks where word types are assigned random labels. Since these control tasks contain no linguistic structure, such results indicate that probe accuracy can be driven by memorization rather than by linguistic information encoded in the representations. Indeed, the small gap between linguistic and control accuracy (e.g., 97.3% versus 92.8%) suggests that probe expressivity dominates the evaluation, making

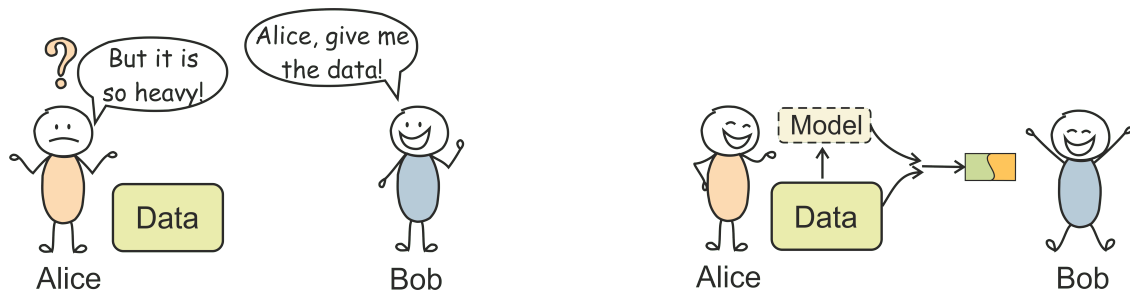


Figure 2.10: Illustration of the Minimum Description Length probing framework. Alice must transmit the labels to Bob, but sending the raw data directly is inefficient. Instead, she first learns a model that captures regularities between the representations and the labels, transmits the model parameters, and then uses the model to compress the labels. The total number of bits required to communicate both the model and the compressed labels corresponds to the MDL of the labels given the representations. Adapted from https://lena-voita.github.io/posts/mdl_probes.html

it difficult to disentangle genuine representational content from the probe’s capacity to memorize type-level associations.

The authors argue that probing accuracy alone is insufficient for interpretation because high-capacity probes can learn linguistic tasks independently of the representations they evaluate. To solve this problem, they introduce control tasks that assign random but deterministic outputs to word types while preserving the structure of the linguistic task. Because control tasks contain no meaningful information, the performance of the probe on them directly measures their capacity for memorization. They define *selectivity* as the difference between linguistic-task accuracy and control-task accuracy, providing a principled metric for determining how much a probe relies on the underlying representation. Finally, they show that probe interpretability improves when probe complexity is tightly controlled through small hidden layers and other constraints.

In response to these challenges, several methodological advances have been proposed to mitigate the confounding effects of probe expressivity. Among these, complexity-sensitive metrics based on the Minimum Description Length principle have emerged as a particularly robust solution. Minimum Description Length reframes probing as a compression problem, evaluating how efficiently a probe can encode linguistic labels given a fixed representation. Rather than considering predictive accuracy alone, this approach jointly accounts for both performance and model complexity, penalizing probes that rely on excessive memorization. As a result, Minimum Description Length provides a principled criterion for assessing whether linguistic information is genuinely encoded in the representations, rather than being an artifact of the probe’s capacity (Voita and Titov, 2020).

Minimum Description Length

The **Minimum Description Length** probing was introduced to address fundamental weaknesses in standard probing evaluations, where accuracy is an unreliable indicator: probes often achieve high performance even when representations contain little or no relevant information (Hewitt and Liang, 2019; Zhang and Bowman, 2018). Voita and Titov (2020) propose MDL probing as an alternative that naturally incorporates a penalty for this memorization behavior and measures not only **how well** a probe predicts linguistic labels but **how hard** it is to achieve it.

The MDL methodology originates in information theory and views learning as a compression problem. A representation encodes a linguistic property well if the labels can be transmitted compactly given the representations. Consider the setting shown in Figure 2.10, *Alice* possesses both the representations and their corresponding labels, while *Bob* has only the representations. *Alice*’s task is to transmit labels to *Bob* with as few bits as possible. Any relationship between representations and labels can be exploited both to predict the labels and to compress them, since high-quality predictions correspond to shorter codelengths under optimal coding schemes. In other words,

Better compression $\leftrightarrow$ Stronger regularity $\leftrightarrow$ Representations better encode labels.

Unlike accuracy-based probing, MDL explicitly accounts for the cost of the model, either by transmitting the model parameters or by measuring how efficiently the model can be learned during sequential data transmission.

Formally, given a dataset $\mathcal{D} = (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ with representations $x_{1:n}$ and labels $y_{1:n}$, we want to measure to what extent $x_{1:n}$ encode $y_{1:n}$. Here, *Alice* has access to all (x_i, y_i) pairs from $\mathcal{D}$, while *Bob* has only $x_{1:n}$ and *Alice* wants to communicate $y_{1:n}$ as efficiently as possible.

From an information-theoretic point of view, the task is to encode labels $y_{1:n}$ knowing representations $x_{1:n}$ in an optimal way, i.e., with minimal codelength (in bits) needed to transmit $y_{1:n}$. The resulting minimal number of bits is the Minimum Description Length of labels given representations. Note that since *Bob* does not know the precise model *Alice* is using, she will also have to transmit the model either explicitly or implicitly. Thus, the methodology can be divided into two parts:

1. **Data codelength**, which reflects how well the model predicts the labels;
2. **Model codelength**, which measures the complexity or capacity of the probe, i.e. how costly it is to specify the model to *Bob*.

Data Codelength Let us assume for now Alice and Bob have agreed in advance on a model $p(y | x)$ and both known the inputs $x_{1:n}$. The code to transmit the labels $y_{1:n}$ losslessly has the following codelength:

$$L_p(y_{1:n} | x_{1:n}) = - \sum_{i=1}^n \log_2 p(y_i | x_i). \quad (2.16)$$

This is the Shannon-Huffman code, which gives the optimal bound on the codelength if the data are independent and come from a conditional probability distribution $p(y_i | x_i)$. Note that Equation (2.16) is exactly the categorical cross-entropy loss evaluated on the model p . This shows that the task of compressing labels $y_{1:n}$ is equivalent to learning a model $p(y | x)$ of data: the quality of a learned model $p(y | x)$ is the codelength needed to transmit the data. Compression is usually compared against uniform encoding, which does not require any learning from data. For K classes, it assumes $p(y | x) = p_{\text{unif}}(y | x) = \frac{1}{K}$, and yields codelength:

$$L^{\text{unif}}(y_{1:n} | x_{1:n}) = n \log_2 K \quad (2.17)$$

Model Codelength The model codelength characterizes how hard it is to achieve the final quality of a trained probe. Intuitively, if representations have some clear structure with respect to labels, this structure can be understood with less effort. Minimum Description Length defines two ways of transmitting the model:

1. **Variational code:** it is the explicit transmission and its goal is to find a simple rule explaining the data structure;
2. **Online code:** the rule explaining the structure is revealed with a small amount of data, and thus the model is transmitted implicitly.

As detailed by Voita and Titov (2020), the two codes are equivalent and provide the same results; the variational code can be used to inspect the induced probe architecture, but the online code is easier to implement since it does not need to change much in standard probe-training pipelines. Given that the two codes are equivalent and in this work the experiments in Section 3.3 were conducted with the online code, in the next paragraph only the online code is detailed.

Intuitively, the online code measures the ability to learn from different amounts of data. In this setting, illustrated in Figure 2.11, the data is transmitted in a sequence of portions; at each step, the data transmitted so far is used to understand the regularity in this data and compress the following portion. Formally, Alice and Bob agree on the

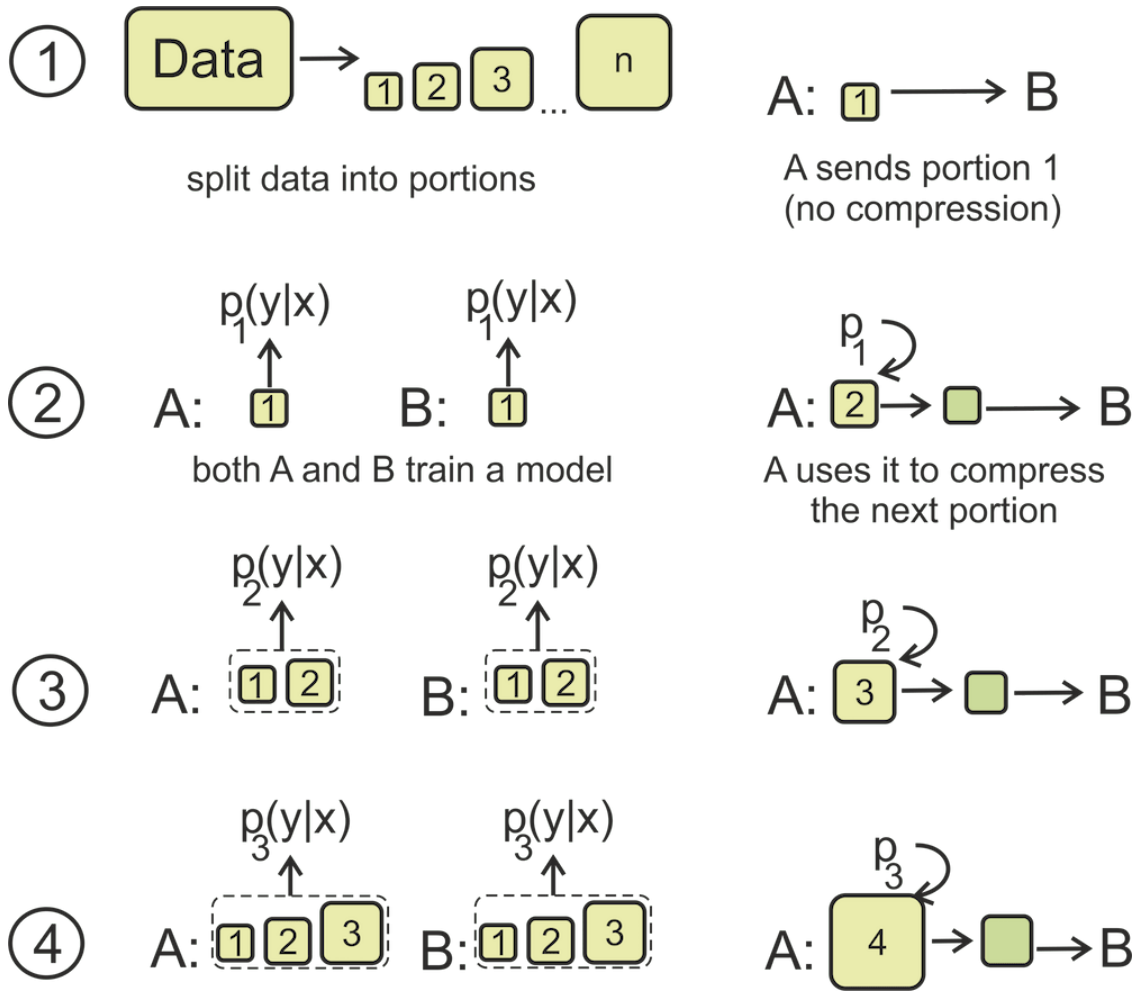


Figure 2.11: Illustration of the MDL online code scheme. (1) The dataset is divided into sequential portions and the first portion is transmitted without compression. (2) After receiving it, both Alice and Bob train an identical model $p_1(y | x)$ on this initial block. (3) Alice uses the learned model to compress the next portion before sending it, and Bob decodes it using the same model. Both parties then retrain on the accumulated data to obtain $p_2(y | x)$. (4) This process continues iteratively: at each step, the current model $p_t(y | x)$ compresses the next block, and both sides update the model with the newly received data. The total online codelength is the sum of all compressed portions, reflecting how quickly structure in the representations can be learned from increasing amounts of data. Adapted from https://lena-voita.github.io/posts/mdl_probes.html

form of the model $p_\theta(y | x)$ with learnable parameters θ , its initial random seeds, and its learning algorithm. They also choose S timesteps $1 = t_0 < t_1 < \dots < t_S = n$ and encode data by blocks. Alice starts by communicating $y_{1:t_1}$ with a uniform code, then, both Alice and Bob learn a model $p_{\theta_1}(y | x)$ that predicts y from x using data $\{(x_i, y_i)\}_{i=1}^{t_1}$, and Alice uses that model to communicate the next data block $y_{t_1+1:t_2}$. Then both Alice and Bob learn a model $p_{\theta_2}(y | x)$ from a larger block $\{(x_i, y_i)\}_{i=1}^{t_2}$ and use it to encode $y_{t_2+1:t_3}$. This process continues until the entire dataset is transmitted. The resulting online codelength

is

$$L^{online}(y_{1:n}|x_{1:n}) = t_1 \log_2 K - \sum_{i=1}^{S-1} \log_2 p_{\theta_i}(y_{i+1:t_{i+1}} | x_{i+1:t_{i+1}}) \quad (2.18)$$

where the first term is equal to the first portion transmitted with the uniform code, and the term inside the summation represents the $(i + 1)$ -th portion compressed with the model trained on all portions from 1 to i . Please note that this second term is equal to the *categorical cross-entropy loss* computed with the trained model over the $(i + 1)$ -th portion. In the online code, the compression metric is thus computed by comparing Equation (2.18) and Equation (2.17) as:

$$C = \frac{L^{unif}}{L^{online}} \quad (2.19)$$

Chapter 3

Understanding LLMs

Understanding how LLMs represent, manipulate, and generalize linguistic information is a central challenge in contemporary computational linguistics. Although LLMs achieve state-of-the-art performance across various tasks, the internal mechanisms supporting this performance remain opaque. This lack of interpretability raises concerns about robustness, reliability, safety, and the broader scientific question of whether these models truly *understand* language or simply exploit statistical regularities at scale.

This thesis leverages the MDL probing technique to evaluate how much knowledge LLMs have over semantic tasks, particularly in detecting figurative language in the form of figures of speech. The task considered has a binary output, where the model has to determine if the sentence contains figurative language, or if it has a literal meaning.

This focus on semantics is motivated by (i) an insufficient investigation by the research community with regard to other linguistic properties, such as grammar and syntax, and (ii) the greater complexity of semantic phenomena. Whereas morphological and syntactic information can often be recovered from surface-level regularities, semantic interpretation requires the model to integrate long-distance dependencies, resolve ambiguity, and construct abstract meaningful representations (Chiang et al., 2020; Saphra and Lopez, 2019). Thus, investigating semantic properties reveals interesting insight into the knowledge of NLMs, their inner workings, and some of their limits.

To better understand how LLMs learn knowledge, the focus is on the so called *learning trajectory*. Given a linguistic property, the learning trajectory is traced while probing multiple subsequent checkpoints saved during the pretraining procedure. At each checkpoint, the same methodology described in Section 3.1 is applied. Trajectory-based analysis renders learning as a process with potentially stable regularities: (i) **ordering effects**, whereby certain capabilities predictably precede others; (ii) **phase structure**, in which qualitatively different solution strategies dominate at different stages; and (iii) **heterogeneous growth**, in which some competencies accelerate, plateau or even regress as optimization reallocates capacity (Choshen et al., 2022).

Following the analysis of these results, it becomes possible to intervene in the model’s training procedure. For example, if a particular linguistic property is acquired early and shows no subsequent improvement, the number of training epochs can be reduced

without loss of performance. Conversely, if the model fails to exhibit a given linguistic property, trajectory-based evidence can reveal whether the property was ever acquired, at which stage it first emerged, and whether it later diminished or was effectively *forgotten* in favor of other competencies.

In addition, training the same model on different corpora also enables the study of how data influence the knowledge encoded by the model. For instance, a corpus rich in literary text would be expected to support the acquisition of figurative language more effectively than a generic web scraped corpus. Finally, learning trajectories allow systematic comparisons across model families and across parameter scales within the same architecture [Chiang et al. \(2020\)](#); [Saphra and Lopez \(2019\)](#). Such comparisons reveal regularities in how linguistic properties emerge and stabilize during training, providing empirical justification for prioritizing specific architectural or scaling choices in future experiments.

Beyond trajectory analysis, it is possible to perform a *layer-wise probing* in which not only the final layer is probed, but each layer of the Transformer stack is associated with an individual probe evaluating the same linguistic task. This analysis highlights how different layers acquire knowledge and use it to produce the final result.

Layer analysis exposes mechanisms that remain invisible to endpoint metrics. It enables *localization*, where certain linguistic properties or information are more readily recoverable from specific layers, indicating where the network concentrates relevant features. Combined with the learning trajectory, it provides more insight into how representations evolve during training: as optimization proceeds, layers may transition from encoding broadly useful statistical regularities to encoding task-, or objective-specific features; adaptation mechanisms such as fine-tuning can redistribute information, amplifying signals that support the target task while attenuating more general-purpose encoding.

Overall, this type of analysis has brought a general intuition that, in Transformer-based architectures, the lower layers predominantly encode syntactic and morphological information, the intermediate layers capture a richer set of semantic features, and the upper layers specialize in task-oriented computations ([Fayyaz et al., 2021](#); [Miaschi et al., 2020](#)).

3.1 Methodology

The methodology adopted leverages the online code of the Minimum Description Length probing method. The probe architecture follows the same structure: given $d \in \mathbb{R}^n$ the language model representation dimension, the probe is made of the following layers:

- A first input layer with d dimensions, with ReLU as activation function;
- A hidden layer that transforms the d -dimensional vector into $\frac{d}{4}$, with ReLU as activation function;
- An output layer that projects the hidden representation into the binary space, with softmax as activation function;

For each checkpoint of each NLM tested, the same methodology is applied.

First, the complete dataset D is divided into the standard train/eval/test splits of 80%/10%/10%, denoted as D_{tr} , D_{ev} , and D_{te} , respectively. Then, the train D_{tr} dataset is divided into incremental portions; as in Voita and Titov (2020), the portions are: 0.1%, 0.2%, 0.4%, 0.8%, 1.6%, 3.2%, 6.25%, 12.5%, 25%, 50%, 100%. For each portion i , a *train* split that contains the examples within the i -th portion (e.g. the 0.2% of data), and an *evaluation* split containing the data from the i -th portion to the $(i + 1)$ -th portion (e.g. the data from the 0.2% to the 0.4%) is defined. For the last portion, in which 100% of the data is used for training, no evaluation split is defined. For example, given a dataset D made of $n = 1250$ sentences, and thus the train dataset D_{tr} with $n_{tr} = 1000$ sentences, the portion 1.6% contains the first 16 examples (with index from 0 to 15) and the evaluation portion contains the following 16 examples, with index from 16 to 31; in Table 3.1 the complete example is reported. This way, each probe is trained on an incremental amount of data and evaluated, in order to compute the MDL score, on new incremental data.

The second phase consists of training the probes over the representation produced by the frozen LLM; the freeze prevents the language model from learning from the data. Each probe is fed with the complete sentence representation, computed as the mean of the token representations. For each portion, a newly initialized probe is trained over the data. Please note that if layer analysis is performed, a new probe is initialized for each layer. Each probe is trained over 40 epochs with a batch size equal to 20; if the loss over the fixed D_{ev} split does not decrease for 4 epochs, an early stopping mechanism terminates the training. Each probe is trained with the Mean Square Error (MSE) loss, with Adam as optimizer and a learning rate equal to 10^{-4} . This hyperparameter remains fixed over all the trials and were decided after some initial runs.

In the third phase, after the training is complete, the data codelength is computed as described in Equation (2.16). In detail, using the best probe weights that achieve the minimum evaluation loss, on the eval split of the current portion is computed the MSE loss. This phase is not executed over the last portion since there is not an eval split. Furthermore, to compare the result of MDL against the classic metric, the accuracy and the F1-score (for both classes) are calculated on the D_{te} test split.

Portion	Training Split		Evaluation Split	
	# Examples	Index	# Examples	Index
0.1%	1	$0 \leq i < 1$	1	$1 \leq i < 2$
0.2%	2	$0 \leq i < 2$	2	$2 \leq i < 4$
0.4%	4	$0 \leq i < 4$	4	$4 \leq i < 8$
0.8%	8	$0 \leq i < 8$	8	$8 \leq i < 16$
1.6%	16	$0 \leq i < 16$	16	$16 \leq i < 32$
3.2%	32	$0 \leq i < 32$	31	$32 \leq i < 63$
6.25%	63	$0 \leq i < 63$	62	$63 \leq i < 125$
12.5%	125	$0 \leq i < 125$	125	$125 \leq i < 250$
25%	250	$0 \leq i < 250$	250	$250 \leq i < 500$
50%	500	$0 \leq i < 500$	500	$500 \leq i < 1000$
100%	1000	$0 \leq i < 1000$	0	—

Table 3.1: A complete example of splitting a dataset D_{tr} with $n_{tr} = 1000$ examples. The index i denotes the position of each example in the dataset. In this setting, the first training portion contains only the first example, while the corresponding evaluation portion contains only the second example. The second training portion includes the first two examples, and the second evaluation portion consists of examples three and four, and so on. For the final portion (corresponding to 100%), the entire dataset is used for training.

Finally, after each portion concludes the training, using each result obtained in the third phase, the MDL online codelength and compression are computed, respectively, with Equation (2.18) and Equation (2.19). Since all tasks have $K = 2$ classes, the formulation is simplified as follows:

$$L^{unif} = n$$

$$L^{online} = t_1 - \sum_{i=1}^{S-1} \log_2 p_{\theta_i}(y_{t_i+1:t_{i+1}} | x_{t_i+1:t_{i+1}})$$

The complete methodology is shown in Figure 3.1, where each yellow block named MDL represents the probing algorithm described above. With the compression results of the final layer of each checkpoint (represented in orange), the learning trajectory is computed. Furthermore, on the final checkpoint, the probing methodology is repeated over the different layers to produce a comprehensive layer analysis.

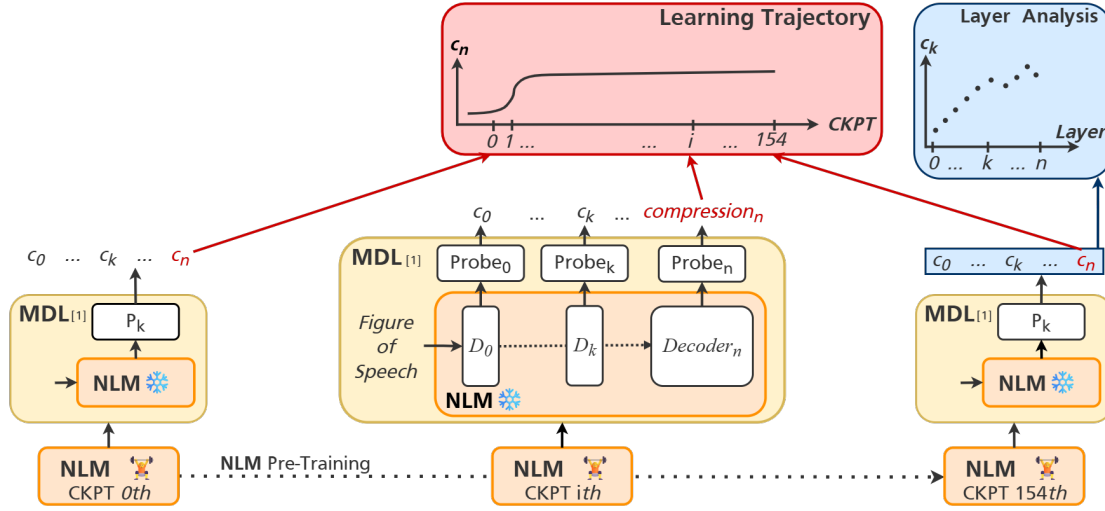


Figure 3.1: Overview of the methodology used to analyze the learning trajectories and the layer-wise behavior in NLMs. For each checkpoint saved during pre-training, the frozen NLM is evaluated through MDL. The last layer value of compression for each checkpoint is used to compute the learning trajectory. Furthermore, for the last checkpoint, the same probing methodology is applied to every layer of each checkpoint.

3.2 Language Models and Tasks

The Pythia Suite

The methodology discussed was tested on several NLMs belonging to the same family, the Pythia suite (Biderman et al., 2023) by EleutherAI, which combines interpretability analysis and scaling laws. These language models were developed to understand how knowledge develops and evolves during training in Transformers.

The Pythia suite was developed with the explicit purpose of enabling research in interpretability, learning dynamics, ethics, and transparency. The key features of the Pythia suite are:

- All models, data, and code used are publicly released, enabling full reproducibility of results;
- All models feature 154 checkpoints saved throughout training, enabling the study of LLMs' learning dynamics;
- All models were trained on the same data in the same order, allowing researchers to explore causal interventions in the training process.

These LLMs leverage the GPT-NeoX architecture by Black et al. (2022), a variant of GPT very similar to GPT-3 (Brown et al., 2020b) and GPT-J (Wang and Komatsuzaki,

# Params	# Layers	Hidden Dim.	# Heads	Head Dim.	Batch Size
14M	6	128	4	32	2M
31M	6	256	8	32	2M
70M	6	512	8	64	2M
160M	12	768	12	64	2M
410M	24	1024	16	64	2M
1.4B	24	2048	16	128	2M
6.9B	32	4096	32	128	2M

Table 3.2: Architectural and optimization hyperparameters for the Pythia model suite, in all its sizes. The configurations illustrate systematic scaling in depth, hidden dimensionality, and attention head structure, maintaining the batch size unchanged. Adapted from <https://github.com/EleutherAI/pythia/tree/main>.

2021). The suite was trained on the Pile (Gao et al., 2021) comprising 825 GiB of English text drawn from heterogeneous components that span academic writing, books, conversational text, legal documents, computer code, mathematics, biomedical abstracts and web forums, thus capturing a wide range of linguistic registers and domains. All LLM sizes are trained on the exact same data, in the exact same order. All models are trained with mixed precision, processing about $299,892,736,000 \approx 300B$ tokens during the full training; this corresponds only to a single epoch on the complete dataset.

The Pythia suite comprises 10 different-size models, ranging from 14M to 12B parameters. These NLMs differ in the number of Transformer decoders, the hidden dimensions, and the number of heads. The only hyperparameter that does not change is the batch size, equal to $2,097,152 \approx 2M$ tokens; this to preserve the training on the exact same data in the exact same order. The complete models configuration are shown in Table 3.2; please note that only the NLMs used in Arici et al. (2025) are displayed.

To promote research on the learning dynamics of LLMs the authors released 154 checkpoints for each model, representing the steps 0 (initialization), 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1000, and then every 1000 subsequent step, up to step 143,000.

Figures of Speech

To investigate the ability of NLMs to recognize figurative language, four distinct figures of speech were probed using the methodology described in Section 3.1. Illustrative examples for each dataset are provided in Table 3.3. In all cases, positive instances correspond to sentences that contain the target figure of speech, while negative instances represent literal, non-figurative interpretations constructed to be syntactically well-formed. All datasets examined are in English, publicly accessible and freely released for research

Dataset	Positive	Negative
Hyperbole	I would give my life for a coffee.	I have lived in this country all my life.
Metaphor	Government is a massive ocean with a thin layer of politics on the top.	The Pacific is a massive ocean
Oxymoron	I'm an honest liar .	An honest person tells the truth, while a liar does not.
Pleonasm	This is a layover stop .	A layover is a short stop between flights.

Table 3.3: Overview of the four figures of speech investigated: hyperbole, metaphor, oxymoron and pleonasm. For each figure, a positive example is reported (i.e. one that contains the relative figure of speech) and a negative example. Words that compose the figure of speech are highlighted in bold.

purposes¹. Together, they cover a broad spectrum of figurative phenomena, differing in lexical structure, semantic complexity, and degree of contextual dependence.

Hyperbole Hyperbole is an expressive form of exaggeration that amplifies or minimizes meaning along qualitative or quantitative scales (Burgers et al., 2016). The dataset originates from Troiano et al. (2018) and was subsequently refined by Schneidermann et al. (2023). It contains 1,396 sentences manually annotated by multiple annotators and balanced between literal and hyperbolic interpretations. Because the examples are drawn from naturally occurring web text, the dataset captures a rich stylistic variety—from overt exaggerations to more subtle rhetorical amplifications. Recognizing hyperbole requires a model to combine lexical cues with real-world plausibility judgments, making it a valuable probe for higher-level semantic reasoning.

Metaphor Metaphor involves expressing a concept in terms of another, merging source and target domains through a perceived semantic resemblance (Strapparava, 2018). This substitution often relies on abstract conceptual mapping rather than surface-level similarity, demanding a deep contextual interpretation from the model. For this task, the Binary LCC dataset is used, a well-established benchmark first introduced by Mohler et al. (2016) and later reformulated by Aghazadeh et al. (2022). From the complete collection of approximately 40,000 annotated sentences, a subset of 1,600 sentences was sampled to maintain comparability with the other figurative datasets. The subset maintains an almost balanced distribution (785 literal, 815 metaphorical) while retaining lexical and conceptual diversity. Success in this task requires the model to move beyond surface cues

¹<https://github.com/nicolarici/learning-trajectories>

and infer deeper semantic relationships, often expressed through subtle domain blending.

Oxymoron An oxymoron is a compact rhetorical device created by juxtaposing two terms with inherently contradictory meanings (e.g., *deafening silence*) (Bolognesi et al., 2024). Because no publicly available dataset offered paired negative examples, a new balanced resource was constructed. Starting from the 287 annotated oxymorons collected by Xu et al. (2023), negative counterparts were generated automatically using the API of GPT-4 by OpenAI. In these sentences, the same lexical items appear in grammatically correct but purely literal contexts. All generated examples were subsequently validated by human reviewers to ensure semantic correctness. The final dataset contains 1,564 sentences, evenly split between oxymoronic and non-oxymoronic instances. A model that performs well on this task must therefore recognize the collision of two incompatible semantic fields.

Pleonasm A pleonasm is a form of lexical redundancy created through the addition of semantically superfluous material (e.g., *free gift*) (Lehmann, 2005). Unlike the other figures of speech considered, pleonasms often lack explicit contextual signals and may require sensitivity to subtle pragmatic conventions. The dataset used here derives from the Semantic Pleonasm Corpus (Kashefi et al., 2018), which includes 3,019 sentences annotated according to redundancy patterns. Because the original corpus uses multi-class labels, it was adapted for binary classification by excluding instances marked as "both words redundant". The resulting dataset consists of 3,002 sentences (1,720 pleonastic and 1,282 non-pleonastic). Compared to the other figurative tasks, pleonasm detection is more lexicalized and less context-dependent, which poses particular challenges for representation-based probing. Furthermore, lexical redundancies of this type are relatively rare in large training corpora, potentially limiting the degree to which NLMs encounter and internalize such patterns during pre-training.

3.3 Experimental Results

This section presents the experimental results obtained from probing pre-trained language models for figurative language recognition. The analysis focuses on learning trajectories observed during training and on the distribution of figurative language knowledge across model layers. Results are reported for different figures of speech and for models of increasing size. The section is structured as follows. First, results on the learning trajectories of the final layer are presented, enabling a comparison across figures of speech and model sizes. Next, a layer-wise analysis is conducted on the final checkpoint, and the

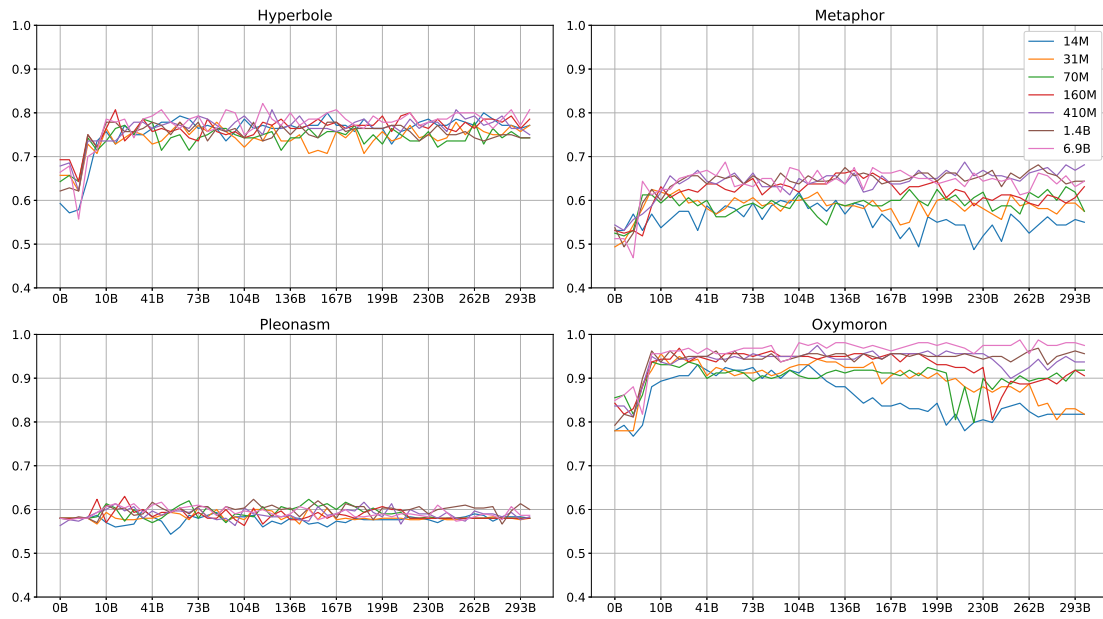


Figure 3.2: Learning trajectories of the GPT-NeoX models considering the last layer in terms of accuracy. Each plot represents a different figure of speech. On the x-axis is reported the number of tokens used for training the model; on the y-axis is shown the accuracy. Each line represents a different model: in blue, the one with 14M parameters, in orange with 31M, in green with 70M, in red with 160M, in purple with 410M, in brown with 1.4B and in pink with 6.9B.

learning trajectories of the best-performing layers are examined. Finally, the impact of pre-training data is analyzed, discussing whether and how different data sources influence performance on figurative language detection tasks.

For each part of the analysis, the results are discussed primarily in terms of Minimum Description Length, which provides a more informative comparison taking into account the representational efficiency. This approach highlights differences among model sizes, training stages, and layers that are not evident under standard evaluation metrics. A straightforward accuracy-based evaluation of figurative language probes offers only limited insight, as performance often appears superficially high and similar across training. For example, as shown in Figure 3.2, hyperbole detection accuracy remains around 70–80% for most of the training process across all model sizes, with little distinction between smaller and larger models. Moreover, accuracy exhibits persistent fluctuations, leading to frequent changes in the best-performing model across checkpoints. This apparent plateau obscures the underlying learning dynamics and capacity-related differences. As noted previously in the literature, with sufficiently large training sets, even weak or random representations can produce relatively high accuracy, masking whether a model genuinely encodes the targeted linguistic knowledge (Hewitt and Liang, 2019; Zhang and Bowman, 2018).

In contrast, the MDL probing framework (Voita and Titov, 2020) provides a more

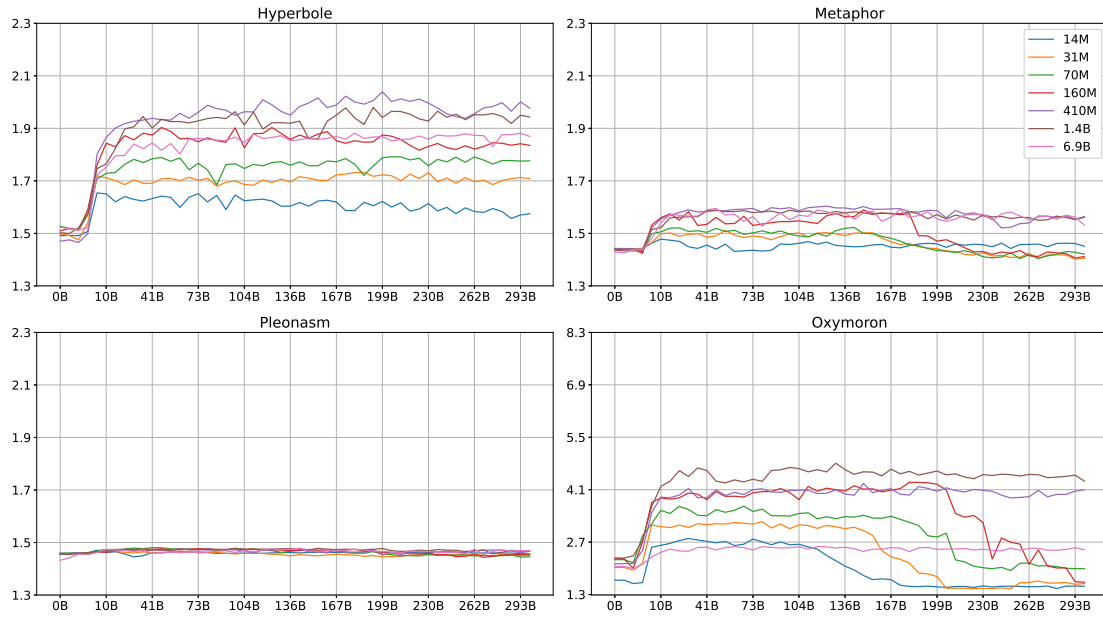


Figure 3.3: Learning trajectories of the GPT-NeoX models considering the last layer in terms of compression. Each plot represents a different figure of speech. On the x-axis is reported the number of tokens used for training the model; on the y-axis compression is shown. Each line represents a different model: in blue, the one with 14M parameters, in orange with 31M, in green with 70M, in red with 160M, in purple with 410M, in brown with 1.4B and in pink with 6.9B.

nuanced view. By jointly considering predictive performance and the amount of data required to achieve it, MDL-based compression reveals clear differences in how and when figurative language knowledge is acquired. For this reason, the analysis focuses on MDL compression trajectories, which offer more stable and interpretable signals of figurative language learning than raw accuracy.

3.3.1 Learning Trajectories of Figurative Language

Using the MDL compression metric, as reported in Figure 3.3, it becomes evident that figures of speech are not learned uniformly, either in terms of difficulty or learning speed. A clear performance gap emerges between oxymoron and hyperbole on the one hand, and metaphor and pleonasm on the other. Across all model sizes, oxymoron recognition appears to be the most learnable task: compression values reach the highest levels, exceeding 4.5 in the largest models, indicating that oxymoronic expressions are comparatively easy to encode and compress. Hyperbole represents the second most accessible figure of speech, with mid-to-large models achieving compression values above 2.0, suggesting that exaggerated expressions align well with recurring patterns in the training data and are therefore learned efficiently.

In contrast, metaphor detection remains substantially more challenging. Even the

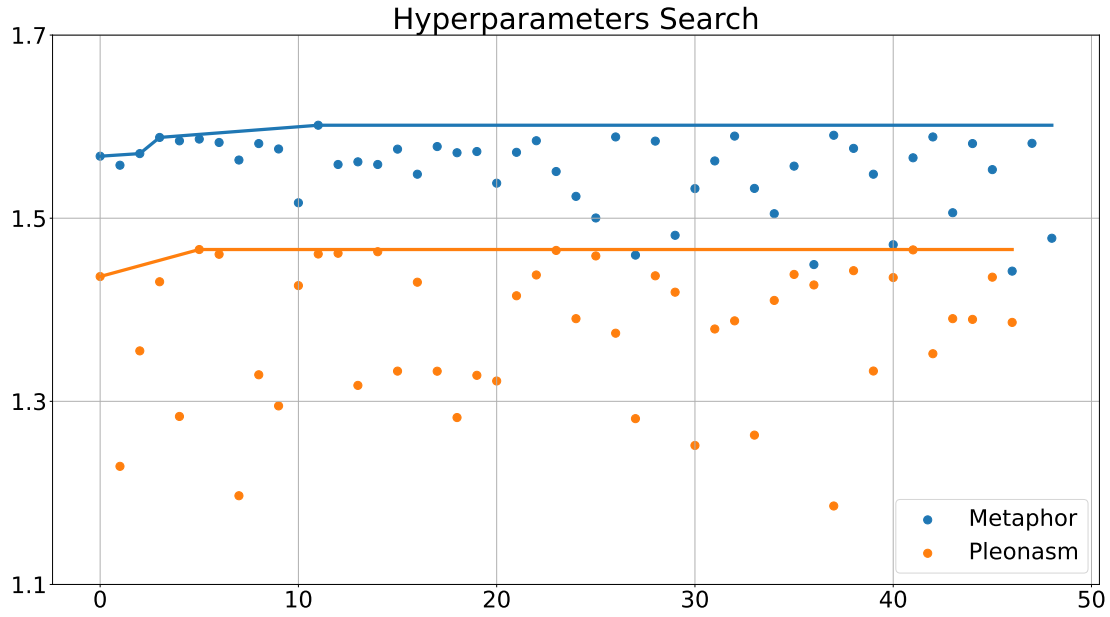


Figure 3.4: Results of the random hyperparameter search conducted on the probe architecture for the metaphor and pleonasm tasks. Each point corresponds to a different probe configuration, evaluated in terms of MDL compression, whereas the solid line indicates the maximum compression achieved across the runs.

largest models, with $6.9B$ parameters, achieve compression values of only around 1.6, markedly lower than those observed for hyperbole and oxymoron. This result reflects the inherent difficulty of capturing metaphorical meaning, which often involves context-dependent semantic shifts rather than surface-level cues. The most challenging task is pleonasm: across all model sizes, compression plateaus near 1.46, a value comparable to that of a randomly initialized model. This indicates an absence of measurable learning, as the models fail to encode information that would allow reliable discrimination among pleonastic and non-pleonastic expressions.

To assess whether the weak results observed for metaphor and pleonasm detection were due to limitations of the probe architecture, an extensive random hyperparameter search was conducted on the probe design. The search explored probes with up to four hidden layers, where each layer could be instantiated with a dimensionality equal to $\frac{d}{2}$, $\frac{d}{4}$, $\frac{d}{8}$, $\frac{d}{16}$ or $\frac{d}{32}$, with d denoting the embedding size of the underlying language model. As reported in Figure 3.4, despite this broad exploration of architectural depth and width, no consistent improvement in MDL compression was observed for metaphor (reaching 1.60 as the maximum value) or pleonasm (which stabilized at 1.46), indicating that probe capacity and architectural expressiveness are not the primary factors limiting performance.

In addition, the impact of the sentence-level pooling strategy was investigated. Since

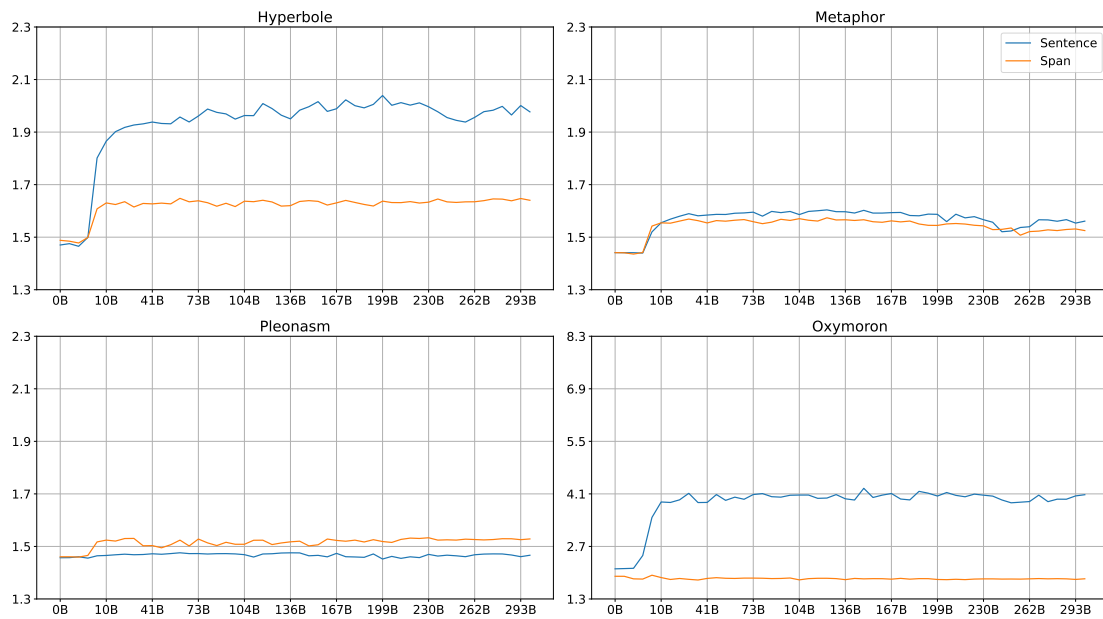


Figure 3.5: Comparison between sentence-level pooling and span-based pooling for the learning trajectories of the 410M GPT-NeoX model at the final layer, measured in terms of MDL compression. Each subplot corresponds to a different figure of speech.

figurative meaning is often localized to specific spans within a sentence, the probe input was restricted to the annotated spans corresponding to the figure of speech, rather than using the average representation of the full sentence. As shown in Figure 3.5, this modification did not improve performance for metaphor or pleonasm, while it degraded for hyperbole and oxymoron.

Finally, experiments were conducted on metaphor with datasets two and four times larger, while the probing architecture and evaluation protocol were kept unchanged. As shown in Figure 3.6, despite this substantial increase, limited improvement in terms of MDL compression was observed, and the overall learning trajectory remained largely unchanged. Metaphor detection continued to saturate early at compression values around 1.6 for larger models, significantly below those obtained for hyperbole and oxymoron. These results indicate that the difficulty of metaphor recognition cannot be attributed primarily to limited training data, but rather reflects intrinsic representational limitations of the models with respect to this figure of speech.

A central result emerging from the MDL trajectories is that figurative-language competence is acquired predominantly in the early stages of pre-training, after which performance largely saturates. Across model sizes and across the figures of speech that show measurable learning (hyperbole, metaphor, and oxymoron), compression rises sharply within the first checkpoints and reaches near-maximal values after roughly 10B tokens. Given that the full pre-training regime spans almost to 300B tokens, this implies

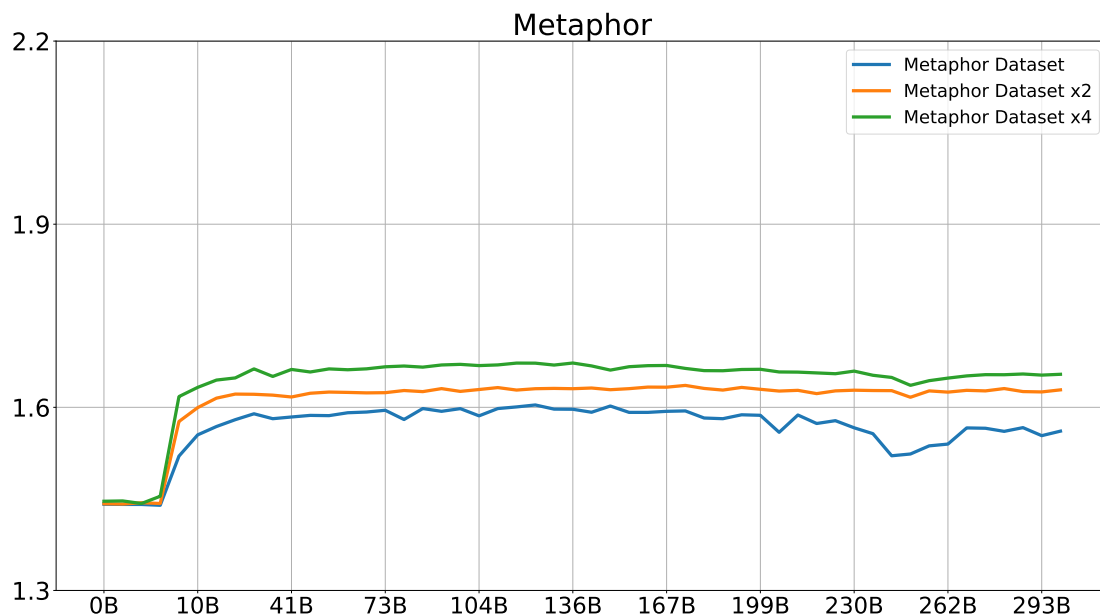


Figure 3.6: Comparison of learning trajectories for metaphor detection using the original dataset and datasets augmented by factors of $\times 2$ and $\times 4$, for the 410M GPT-NeoX model at the final layer, evaluated in terms of MDL compression.

that most of the extractable figurative-language knowledge is learned after only a small fraction of the total training budget. Overall, the MDL trajectories suggest that the cues exploited by the models to recognize certain figures of speech are available in relatively frequent or early-occurring contexts in the pre-training data, enabling rapid acquisition; once this information is learned, continued pre-training yields diminishing returns for these semantic probes. In practical terms, these results indicate that a substantial portion of figurative-language competence does not require the full scale of the training corpus, and that most of the gains are achieved early, with limited benefits from additional exposure.

Across the considered figures of speech, as shown in Figure 3.3, model size is generally associated with higher MDL compression, supporting the intuition that larger models encode figurative language information more efficiently. This trend is particularly clear at small-to-medium scales. For instance, on the hyperbole task, the 31M parameters model reaches a peak compression of approximately 1.73, while the 160M model peaks at 1.85, and the 410M model reaches 2.04. These values suggest a consistent gain when moving from tens of millions to a few hundred million parameters, supporting the intuition that additional capacity enables richer semantic representations. A similar pattern is observed for oxymoron and metaphor, where performance tends to improve as model size increases from 14M up to mid-sized configurations (roughly 160M and 410M parameters).

However, scaling beyond this regime yields diminishing returns. In particular, the

largest models do not substantially improve over the 410M for some tasks: for hyperbole, the 1.4B and 6.9B models remain around 1.9 compression, constantly worse than the 410M model throughout the trajectory, and for metaphor the models with 410M, 1.4B, and 6.9B parameters converge to similar compression values of approximately 1.6. This plateau suggests that, for these probes, much of the figurative-language knowledge is already captured by medium-sized models, and additional parameters provide limited further benefit.

Oxymoron offers a complementary perspective. While multiple models exhibit rapid improvements during the early stages of training, the best-performing configurations achieve substantially higher compression than those observed for hyperbole or metaphor, with the 1.4B model exceeding a compression value of 4.5 at its peak. In contrast, all models up to 160M parameters show a steady reduction in compression during the final stages of training. The point at which this decline begins varies across model sizes: the 14M model begins to degrade after approximately 100B tokens, the 31M model after 142B tokens, and the 70M model after 167B tokens, whereas the 160M model exhibits a similar trend only after around 199B tokens.

The 6.9B model exhibits a distinct behavior: its performance does not show a pronounced peak, remaining approximately constant around a compression value of 2.6 throughout the training, and it does not show the late-stage decline observed in smaller models. Overall, these observations indicate that increased model capacity is neither sufficient nor guaranteed to yield uniform improvements across all semantic phenomena.

Investigating the 7B Anomaly

An intriguing exception to the general scaling trend emerges when considering the largest models in the Pythia suite. Although compression values consistently improve as the size of the model increases from the 14M to 1.4B parameters, the 6.9B model does not follow this pattern, yielding performance comparable to, or even lower than, models with a fraction of their parameter count. This anomalous behavior, observed across multiple figures of speech, motivates a dedicated investigation to determine whether it reflects a property specific to this model or a more general phenomenon shared across large-scale decoder-only architectures. To this end, we evaluate two additional LLMs using the same probing framework: Amber, a 6.7B parameter model by [Liu et al. \(2023\)](#) from the LLaMA family ([Touvron et al., 2023](#)), and OLMo, a Transformer decoder-only 7B parameter model presented in [Groeneveld et al. \(2024\)](#).

Amber ([Liu et al., 2023](#)) is an English model with 6.7B parameters released by LLM360 as part of an initiative toward fully transparent and reproducible open-source LLMs. It adopts the same architecture as LLaMA 7B, incorporating rotary positional embeddings,

	Model	Last Layer							
		2%	4%	6%	8%	10%	20%	50%	100%
Hyperbole	Pythia 6.9B	1.73	1.78	1.79	1.80	1.84	1.80	1.88	1.87
	Amber 6.7B	1.76	1.68	1.78	1.70	1.80	1.81	1.72	1.83
	OLMo 7B	1.83	1.84	1.89	1.90	1.86	1.86	1.88	1.90
Metaphor	Pythia 6.9B	1.51	1.55	1.57	1.57	1.57	1.57	1.58	1.53
	Amber 6.7B	1.58	1.55	1.53	1.52	1.50	1.57	1.56	1.57
	OLMo 7B	1.54	1.53	1.55	1.56	1.54	1.58	1.57	1.55
Oxymoron	Pythia 6.9B	2.37	2.47	2.47	2.46	2.45	2.57	2.57	2.49
	Amber 6.7B	2.47	2.41	2.50	2.56	2.55	2.49	2.55	2.48
	OLMo 7B	2.55	2.49	2.50	2.59	2.53	2.59	2.58	2.53
Pleonasm	Pythia 6.9B	1.47	1.48	1.48	1.49	1.48	1.48	1.51	1.52
	Amber 6.7B	1.47	1.46	1.48	1.47	1.48	1.47	1.47	1.47
	OLMo 7B	1.47	1.47	1.48	1.46	1.48	1.47	1.47	1.47

Table 3.4: Learning trajectory, in terms of compressions, of the Pythia, Amber and OLMo models for all four figures of speech considering the last layer. For each model, the table shows the compression across different training percentages (from 2% to 100%). The best results among the models are highlighted in bold.

32 hidden layers with 32 attention heads each, a hidden size of 4096, and a vocabulary of 32K tokens. The model was trained on approximately 1.26T tokens drawn from a heterogeneous mixture of sources, including books, Wikipedia, RefinedWeb, academic writing from ArXiv, and code repositories such as StarCoder and StackExchange, thereby covering a broad range of domains. As with Pythia (Biderman et al., 2023), a key feature of the LLM360 initiative is its commitment to full reproducibility: model weights, training code, hyperparameters, configurations, and intermediate checkpoints are all publicly released. The pretraining corpus is divided into 360 equal-sized data chunks of approximately 3.5B tokens each, with a checkpoint saved upon completion of each chunk. This results in 360 checkpoints distributed throughout the training, each corresponding to approximately 760 iterations given a batch size of 2240 sequences of 2048 tokens. This level of granularity enables fine-grained analysis of learning dynamics, comparable to that of the Pythia suite, despite the larger training scale.

OLMo (Groeneveld et al., 2024) is a 7B parameter English-language model developed by the Allen Institute for AI (Ai2), designed to support open and reproducible research on language model science. It adopts a decoder-only Transformer architecture with several modifications over standard design choices used in models such as LLaMA (Touvron et al., 2023) and PaLM Chowdhery et al. (2023). The model was trained on the Dolma dataset (Soldaini et al., 2024), a large-scale open corpus of approximately 2T tokens derived from web documents (Common Crawl), GitHub code, Reddit, Semantic Scholar papers, Project

Gutenberg books, and Wikipedia, with careful attention to data quality and filtering. Like Pythia and Amber, OLMo emphasizes full transparency: training code, model weights, optimizer states, training logs, and the complete pretraining dataset are publicly available. Intermediate checkpoints are saved every 1000 training steps, corresponding, given a batch size of approximately $4M$ tokens, to intervals of $4B$ tokens. This results in more than 500 checkpoints across training. Such granularity enables trajectory-based analyses of learning dynamics analogous to those supported by Pythia, while reflecting a substantially larger training scale.

The results reported in Table 3.4 compare the last-layer learning trajectories of Pythia 6.9B, Amber 6.7B, and OLMo 7B across all four figures of speech, expressed as a percentage of pretraining progress to account for differences in training scale. Across tasks, the three models exhibit broadly consistent patterns, suggesting that the anomalous behavior observed for Pythia 6.9B is not an isolated artifact, but rather a general characteristic of large-scale decoder-only architectures of comparable size.

For hyperbole, OLMo 7B achieves the highest compression values throughout training, reaching a final compression of 1.90, while Pythia 6.9B and Amber 6.7B attain comparable values of 1.87 and 1.83, respectively. Amber, however, exhibits greater instability across checkpoints, with compression fluctuating between 1.68 and 1.81. For metaphor, all three models converge to similarly low compression values, clustered between 1.50 and 1.58 throughout the training. No model exhibits a consistent advantage, reinforcing the conclusion that metaphor detection poses an intrinsic representational challenge not resolved by scale alone. For pleonasm, all models remain near the chance level, with compression values oscillating around 1.50. These findings are consistent with those observed in the GPT-NeoX models, supporting the interpretation that pleonasm is not robustly encoded in the representations examined, regardless of model size, architecture, or training data.

The oxymoron task reveals a more nuanced pattern. OLMo 7B consistently achieves the highest compression values, with a peak of 2.59 at 8% and 20% of training. Critically, all three models plateau well below the levels observed in smaller Pythia variants such as 410M and 1.4B, confirming that performance degradation on the 7B scale generalizes across architectures. The underlying mechanism driving this performance degradation at the 7B scale remains an open question; however, the consistency of results across architectures trained on substantially different corpora suggests that it reflects an intrinsic property of models at this parameter scale rather than a model specific artifact.

Taken together, these findings indicate that the anomalous scaling behavior observed for Pythia 6.9B is not specific to that model family, but reflects a broader property of large decoder-only Transformers at this scale. The consistency across Pythia, Amber,

and OLMo, despite differences in architecture, data, and optimization, provides evidence that this behavior is reproducible. Nevertheless, further experiments on larger models, such as OLMo 13B, are required to strengthen this conclusion.

Comparison with other Transformers Architectures

To evaluate whether the learning dynamics observed for GPT-NeoX extend to other Transformer architectures, Table 3.5 compares last-layer MDL compression trajectories with those of BERT and RoBERTa (Liu et al., 2021), with, respectively, 110M and 125M parameters. Since these models differ substantially in training data and checkpoint saving strategies, the learning progress is measured as a function of the percentage of pre-training completed rather than absolute token counts.

Across tasks, the last-layer learning dynamics of BERT and RoBERTa closely mirror those observed for GPT-NeoX. For hyperbole and oxymoron, both models exhibit rapid early learning followed by stabilization, reaching final compression values comparable to mid-to-large GPT-NeoX models. In particular, BERT and RoBERTa attain hyperbole compression values (up to 1.93 and 1.87, respectively) similar to GPT-NeoX 160M and approaching the 1.4B model, while for oxymoron they reach 4.23 and 3.81, comparable to GPT-NeoX 410M and close to the 1.4B model, despite having substantially fewer parameters. Notably, the pronounced late-stage degradation observed in smaller GPT-NeoX models is absent in BERT and considerably attenuated in RoBERTa. In contrast, metaphor detection remains challenging across architectures, with all models converging to low compression values around 1.55 at the final layer, and pleonasm shows no measurable learning, with compression values remaining near 1.50 throughout training.

Overall, these results indicate that BERT and RoBERTa match the last-layer performance of much larger GPT-NeoX models for the more learnable figures of speech, while sharing the same limitations for metaphor and pleonasm. This suggests that the observed learning dynamics and task-specific difficulty patterns are not specific to GPT-NeoX models, but reflect broader properties of Transformer-based language models when figurative language knowledge is accessed from the final layer.

3.3.2 Layer-Wise Distribution of Figurative Language Knowledge

The learning trajectory analysis discussed in the previous section focuses on the final layer and shows that figurative language knowledge is often acquired early and, in some cases, becomes less accessible as training progresses. This observation raises the question of whether the apparent late-stage degradation reflects a genuine loss of information or, rather, a redistribution of where such knowledge is encoded within the model. To address

	Model	Last Layer							
		2%	4%	6%	8%	10%	20%	50%	100%
Hyperbole	BERT	1.81	1.79	1.82	1.83	1.81	1.84	1.86	1.93
	RoBERTa	1.73	1.78	1.83	1.82	1.84	1.86	1.87	1.87
	GPT 14M	1.66	1.64	1.63	1.64	1.63	1.6	1.62	1.57
	GPT 31M	1.69	1.72	1.68	1.69	1.70	1.70	1.71	1.71
	GPT 70M	1.74	1.75	1.79	1.76	1.78	1.79	1.77	1.78
	GPT 160M	1.81	1.87	1.86	1.87	1.86	1.86	1.89	1.84
	GPT 410M	1.81	1.87	1.92	1.92	1.93	1.96	1.99	1.98
	GPT 1.4B	1.75	1.79	1.81	1.90	1.91	1.93	1.90	1.94
	GPT 6.9B	1.73	1.78	1.79	1.80	1.84	1.80	1.88	1.87
Metaphor	BERT	1.55	1.56	1.54	1.56	1.56	1.55	1.55	1.55
	RoBERTa	1.53	1.55	1.57	1.57	1.58	1.57	1.55	1.56
	GPT 14M	1.47	1.48	1.47	1.47	1.45	1.43	1.45	1.45
	GPT 31M	1.47	1.49	1.51	1.49	1.50	1.49	1.49	1.41
	GPT 70M	1.49	1.50	1.52	1.52	1.51	1.51	1.51	1.42
	GPT 160M	1.53	1.56	1.55	1.55	1.58	1.54	1.58	1.41
	GPT 410M	1.54	1.57	1.55	1.58	1.59	1.59	1.58	1.56
	GPT 1.4B	1.50	1.55	1.56	1.56	1.56	1.58	1.57	1.56
	GPT 6.9B	1.51	1.55	1.57	1.57	1.57	1.57	1.58	1.53
Oxymoron	BERT	3.55	3.80	3.91	3.88	4.01	4.29	4.06	4.23
	RoBERTa	3.52	4.25	4.14	4.17	4.37	4.39	4.25	3.81
	GPT 14M	2.56	2.58	2.67	2.74	2.80	2.61	1.80	1.52
	GPT 31M	3.09	3.10	3.05	3.16	3.09	3.22	2.94	1.59
	GPT 70M	3.25	3.59	3.50	3.65	3.57	3.55	3.38	1.99
	GPT 160M	3.87	3.84	3.94	3.88	4.01	3.92	4.22	1.63
	GPT 410M	3.78	3.75	3.91	3.96	4.13	4.03	4.13	4.09
	GPT 1.4B	3.81	4.21	4.27	4.61	4.45	4.36	4.63	4.33
	GPT 6.9B	2.37	2.47	2.47	2.46	2.45	2.57	2.57	2.49
Pleonasm	BERT	1.48	1.48	1.48	1.47	1.47	1.48	1.48	1.47
	RoBERTa	1.48	1.47	1.48	1.48	1.48	1.47	1.48	1.48
	GPT 14M	1.47	1.46	1.46	1.46	1.45	1.46	1.46	1.46
	GPT 31M	1.47	1.47	1.46	1.46	1.46	1.46	1.45	1.45
	GPT 70M	1.46	1.47	1.47	1.48	1.48	1.47	1.47	1.45
	GPT 160M	1.46	1.47	1.47	1.47	1.47	1.47	1.47	1.45
	GPT 410M	1.47	1.47	1.47	1.47	1.47	1.48	1.46	1.47
	GPT 1.4B	1.46	1.47	1.47	1.47	1.47	1.48	1.48	1.47
	GPT 6.9B	1.47	1.48	1.48	1.49	1.48	1.48	1.51	1.52

Table 3.5: Learning trajectory, in terms of compressions, of the GPT-NeoX, BERT and RoBERTa models for all four figures of speech considering the last layer. For each model, the table shows the compression across different training percentages (from 2% to 100%). The best results among the models are highlighted in bold.

this issue, the analysis is extended beyond the final layer to examine probe performance across all layers of the network.

Model	# Layer	Hyperbole		Metaphor		Oxymoron		Pleonasm	
		BL	↑ %	BL	↑ %	BL	↑ %	BL	↑ %
BERT	12	9	5.1	10	1.6	9	34.4	10	0.9
RoBERTa	12	10	9.3	11	2.4	9	59.1	5	0.9
GPT 14M	6	5	9.7	6	0.0	5	28.0	5	0.4
GPT 31M	6	5	21.4	2	6.3	3	91.0	1	1.5
GPT 70M	6	4	24.2	4	7.0	2	87.8	0	2.5
GPT 160M	12	8	25.1	9	11.2	6	206.8	10	2.1
GPT 410M	24	14	9.2	15	3.5	12	50.3	12	2.3
GPT 1.4B	24	13	9.2	11	1.9	11	72.9	10	2.8
GPT 6.9B	32	18	4.8	17	5.1	11	13.7	9	3.6

Table 3.6: Results of the layer analysis conducted on the GPT-NeoX models. Column # Layer gives the total number of layers; the columns of the considered figures of speech give the best performing layer (column BL) and the improvement of the compression metric in percentage with respect to the last layer (column ↑ %).

The MDL-based analysis reveals distinct peaks in compression at specific intermediate layers, indicating that the figurative language information is encoded more efficiently in other layers with respect to the last one. Table 3.6 reports, for each model and task, the best-performing layer and the relative improvement in compression with respect to the final layer. Across model sizes, the layers yielding the highest compression are typically located in the middle or upper-middle portion of the network, rather than at the final layer.

These findings are consistent with prior observations that different linguistic properties are distributed across Transformer layers, with syntactic information concentrated in lower layers, semantic information in intermediate layers, and task-specific signals dominating the top layers (Fayyaz et al., 2021; Miaschi et al., 2020). The present results suggest that figurative language, as a semantic phenomenon, is primarily represented in intermediate layers, and that the final layer often provides a suboptimal view of this information when accessed through probing.

Based on the layer-wise comparison introduced above, the magnitude of the advantage provided by intermediate layers varies substantially across figures of speech, as illustrated in Figure 3.7. For hyperbole, intermediate layers consistently yield markedly higher MDL compression than the final layer across architectures. In medium-sized GPT-NeoX models, the best-performing layers improve compression by approximately 20% relative to the final layer, with peak values observed around layer 8 for the 160M model and around layer 14 for the 410M model. A similar pattern is visible in BERT and RoBERTa, where compression increases steadily from the lower layers and reaches a maximum in the middle of the network (around layer 9 and 10).

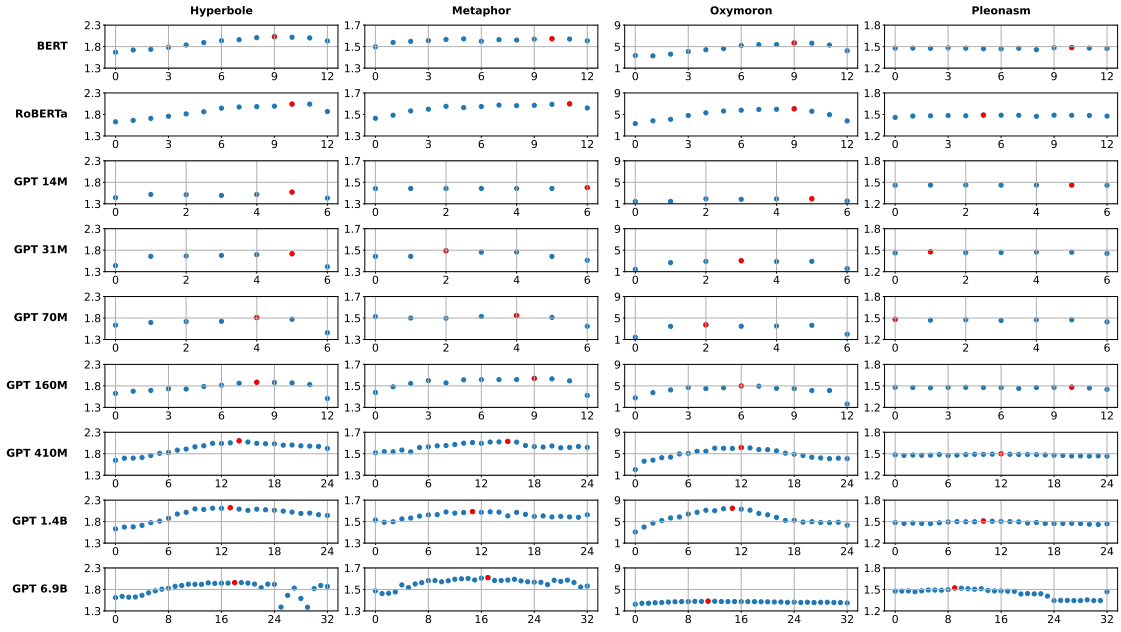


Figure 3.7: MDL compression on the final checkpoint calculated for each layer (on the x-axis of each plot) of all the considered models. For each model (grouped in rows) the figure shows four different plots, one per figure of speech analyzed. The best layer is highlighted in red.

For metaphor, the layer-wise curves show a much flatter profile. Across models, the difference between the best layer and the final layer remains limited, typically within 2–5%. Even in larger GPT-NeoX models (410M, 1.4B, 6.9B), peak compression values around 1.61 occur at intermediate layers (between layers 11 and 17), but the improvement over the final layer is marginal. This indicates that metaphorical information, when present, is weakly distributed across layers rather than concentrated at a specific depth.

The most extreme case is pleonasm, for which the layer-wise curves are nearly flat across all architectures. Compression values remain close to 1.50 throughout the network, and no layer offers a meaningful advantage over the others. This behavior is consistent across BERT, RoBERTa, and all GPT-NeoX variants, reinforcing the conclusion that pleonasm is not robustly encoded in any accessible representation.

Finally, oxymoron displays a distinct layer-wise pattern, characterized by pronounced peaks in intermediate layers and a marked degradation at the final layer in smaller models. For example, in the 160M model, while oxymoron compression at the final layer drops sharply during late training, the sixth layer achieves a compression value that is 206.8% higher than that of the final layer, with a peak compression of 4.99. Comparable recoveries are observed in smaller models, with improvements of approximately 87–91% in the 70M and 31M configurations. Even in the 1.4B model, which does not exhibit clear last-layer forgetting, the best intermediate layer still outperforms the final layer by a margin of 73%.

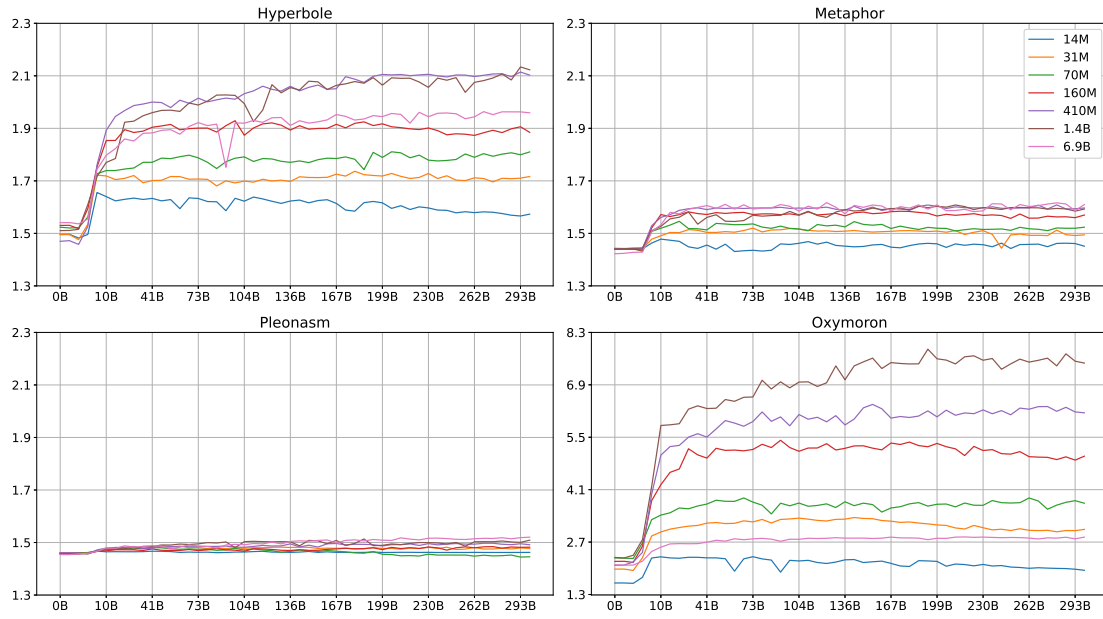


Figure 3.8: Learning trajectories of the GPT-NeoX models considering the best performing layer. Each plot represents a different figure of speech. On the x-axis, we indicate the number of tokens used for training the model; on the y-axis we indicate the compression calculated using MDL. Each line represents a different model: in blue, the one with 14M parameters, in orange with 31M, in green with 70M, in red with 160M, in purple with 410M, in brown with 1.4B and in pink with 6.9B

These effects are illustrated in Figure 3.8, which reports learning trajectories when probing each model at its optimal layer rather than at the final layer. Under this evaluation, in the oxymoron task, none of the models display the severe late-stage degradation observed at the output layer. Instead, compression curves remain stable or continue to increase slightly throughout training. For instance, in the 1.4B model, compression at the best layer rises from 5.82 at approximately 10B tokens to 6.82 at 100B tokens, and further to 7.48 at the end of training. This trend suggests that oxymoron-related information continues to be consolidated in intermediate layers even when it becomes less accessible from the final layer.

The smallest model (14M) represents a partial exception, showing a gradual decline in best layer compression during late training; however, this decrease remains considerably less pronounced than the collapse observed at the final layer. Overall, these findings indicate that apparent late-stage *forgetting* of the figurative language is primarily a representational shift rather than a complete loss of knowledge, with relevant information persisting in intermediate layers despite reduced accessibility in the model’s output.

A similar comparison can be drawn for the remaining figures of speech. For hyperbole, probing the best layer leads to consistently higher compression than the final layer across all model sizes, with early gains already visible after approximately 10B tokens. In larger

	Model	Best Layer							
		2%	4%	6%	8%	10%	20%	50%	100%
Hyperbole	BERT	1.81	1.79	1.83	1.83	1.86	1.91	1.94	2.03
	RoBERTa	1.72	1.78	1.83	1.82	1.83	1.92	1.96	2.04
	GPT 14M	1.65	1.64	1.63	1.63	1.63	1.59	1.63	1.57
	GPT 31M	1.71	1.72	1.69	1.71	1.72	1.72	1.71	1.72
	GPT 70M	1.74	1.75	1.75	1.75	1.75	1.79	1.78	1.81
	GPT 160M	1.81	1.85	1.85	1.90	1.88	1.89	1.91	1.88
	GPT 410M	1.82	1.89	1.94	1.97	1.99	2.01	2.06	2.10
	GPT 1.4B	1.75	1.79	1.86	1.92	1.93	1.96	1.98	2.12
	GPT 6.9B	1.76	1.82	1.83	1.86	1.85	1.88	1.91	1.96
Metaphor	BERT	1.54	1.53	1.55	1.54	1.55	1.53	1.56	1.57
	RoBERTa	1.53	1.55	1.58	1.57	1.56	1.59	1.58	1.60
	GPT 14M	1.47	1.48	1.47	1.47	1.45	1.43	1.45	1.45
	GPT 31M	1.49	1.49	1.50	1.50	1.51	1.50	1.51	1.49
	GPT 70M	1.47	1.52	1.53	1.55	1.52	1.53	1.53	1.52
	GPT 160M	1.53	1.57	1.57	1.57	1.58	1.58	1.58	1.57
	GPT 410M	1.55	1.57	1.57	1.59	1.59	1.60	1.59	1.60
	GPT 1.4B	1.52	1.56	1.56	1.56	1.58	1.55	1.59	1.59
	GPT 6.9B	1.50	1.57	1.57	1.57	1.59	1.59	1.61	1.61
Oxymoron	BERT	3.81	3.93	4.30	4.49	4.13	4.89	5.02	5.68
	RoBERTa	3.91	4.83	4.69	4.86	4.95	5.25	5.80	6.05
	GPT 14M	2.27	2.27	2.24	2.27	2.29	1.92	2.25	1.95
	GPT 31M	2.87	3.00	3.03	3.09	3.13	3.20	3.31	3.04
	GPT 70M	3.30	3.54	3.50	3.61	3.59	3.79	3.61	3.74
	GPT 160M	3.87	4.37	4.60	4.65	5.19	5.16	5.35	4.99
	GPT 410M	4.33	4.94	5.14	5.28	5.50	5.88	6.04	6.15
	GPT 1.4B	4.99	5.46	5.48	5.86	6.25	6.46	7.33	7.48
	GPT 6.9B	2.48	2.62	2.64	2.66	2.66	2.79	2.81	2.83
Pleonasm	BERT	1.48	1.48	1.48	1.49	1.49	1.49	1.49	1.49
	RoBERTa	1.47	1.48	1.48	1.48	1.48	1.48	1.49	1.49
	GPT 14M	1.47	1.46	1.46	1.46	1.46	1.46	1.46	1.46
	GPT 31M	1.47	1.48	1.48	1.48	1.48	1.49	1.48	1.48
	GPT 70M	1.46	1.47	1.47	1.48	1.48	1.47	1.47	1.45
	GPT 160M	1.47	1.47	1.47	1.47	1.47	1.47	1.46	1.48
	GPT 410M	1.48	1.48	1.48	1.48	1.48	1.48	1.49	1.49
	GPT 1.4B	1.45	1.48	1.47	1.48	1.47	1.49	1.49	1.51
	GPT 6.9B	1.47	1.48	1.48	1.49	1.48	1.48	1.51	1.52

Table 3.7: Learning trajectory, in terms of compressions, of the GPT-NeoX, BERT and RoBERTa models for all four figures of speech considering the best performing layer. For each model, the table shows the compression across different training percentages (from 2% to 100%). The best results among the models are highlighted in bold.

models, compression stabilizes around 2.10, compared to final-layer values closer to 2.00, indicating a relative improvement.

For metaphor, the best-layer trajectories closely resemble those observed at the final layer. Compression increases modestly during early training and quickly saturates, with peak values around 1.62 for large models, only marginally higher than the corresponding final-layer values. The case of pleonasm shows the least variation across layers. The best layer compression values remain near chance level throughout training, fluctuating around 1.50 for all models. No clear learning trend emerges, and the best-layer trajectories do not differ significantly from those obtained at the final layer, reinforcing the conclusion that pleonasm is not robustly represented in the models.

Taken together, these best-layer learning trajectories indicate that intermediate layers provide a substantial advantage only for figures of speech that are effectively learned, such as oxymoron and, to a lesser extent, hyperbole. In contrast, for metaphor and pleonasm, selecting the optimal layer yields limited or no benefit, suggesting that the difficulty of these phenomena reflects representational limitations that persist across the entire network rather than being confined to the output layer.

When considering the best-performing layer, as shown in Table 3.7, BERT and RoBERTa exhibit learning trajectories comparable to those of mid- to large-scale GPT-NeoX models. For hyperbole, both models reach final compression values around 2.03, surpassing the 160M GPT-NeoX model and getting close to the 410M. For oxymoron, RoBERTa (best compression value 6.05) and BERT (5.68) outperform smaller GPT-NeoX variants and closely approach the performance of GPT-NeoX 410M, although the 1.4B model remains the strongest overall. In contrast, metaphor remains challenging across architectures, with best-layer compression values tightly clustered around 1.60 even for the largest models, and pleonasm shows no measurable improvement for any architecture.

Taken together, these results reinforce the observation made in Section 3.3.1 that BERT and RoBERTa behave similarly to slightly larger GPT-NeoX models, supporting the hypothesis that differences in architecture are less influential than effective representational capacity when figurative language information is accessed at the optimal layer.

Overall, these layer-wise results confirm that when a figure of speech is effectively learned, such as hyperbole and oxymoron, its representation is concentrated in intermediate layers, yielding substantial gains over the final layer. Conversely, for phenomena that are weakly learned or inherently difficult, such as metaphor and pleonasm, no layer provides a decisive advantage, and improvements remain limited regardless of depth. These observations further support the interpretation that late-stage degradation at the final layer reflects a redistribution of information rather than a complete loss of figurative language knowledge, as previously described in the literature (Tirumala et al., 2022; Wallat et al., 2020).

3.3.3 Impact of Training Data

To investigate the impact of pre-training data on the acquisition of figurative language, a controlled retraining experiment is conducted using selected models from the Pythia suite. In particular, we trained two GPT-NeoX architectures with 70M and 160M parameters from scratch on alternative English language corpora, allowing a direct comparison among different data regimes while keeping model architecture and optimization settings constant.

Two datasets provided by Dolma (Soldaini et al., 2024) are considered, both substantially smaller than the original Pythia training corpus and differing in linguistic characteristics. The first dataset is **Project Gutenberg**, a literary corpus containing approximately 55,000 books, corresponding to 97 GiB of text and approximately 5.3 billion tokens. This corpus consists primarily of narrative and literary prose and is expected to exhibit a higher density of figurative language, including metaphors and stylistic constructions.

The second dataset is a **Wikipedia** dump, comprising approximately 6 million pages, corresponding to 82 GiB of text and approximately 3.8 billion tokens. The Wikipedia text is predominantly encyclopedic and factual, characterized by a more literal and expository style with limited stylistic variation. These datasets provide a contrast between a more literary and a more generic training regime, while remaining comparable in scale.

We trained the GPT-NeoX models from scratch on each corpus following the same procedure and hyperparameter configuration used for the original Pythia models, as described in Biderman et al. (2023). No architectural or optimization parameters are modified, ensuring that observed differences in learning behavior can be attributed to the training data alone.

Training is carried out for a total of 41B tokens, which corresponds to 20,000 iterations of 2 million tokens each. This training setting is selected for two reasons. First, prior learning-trajectory analyses indicate that figurative language performance does not show visible improvement beyond this threshold. Second, it maintains the same checkpoint-saving strategy, thus enabling direct comparison with earlier stages of Pythia training while avoiding excessive overfitting on smaller datasets.

Given the limited size of the Project Gutenberg and Wikipedia corpora, this training budget results in multiple passes over the data, unlike Pythia, which is trained for one epoch of The Pile (Gao et al., 2021). Specifically, the Project Gutenberg models are trained for approximately 7 epochs, while the Wikipedia models are trained for roughly 11 epochs. As a consequence, later stages of training involve repeated exposure to the same data, in contrast to the original Pythia models, which continue to encounter novel text throughout training.

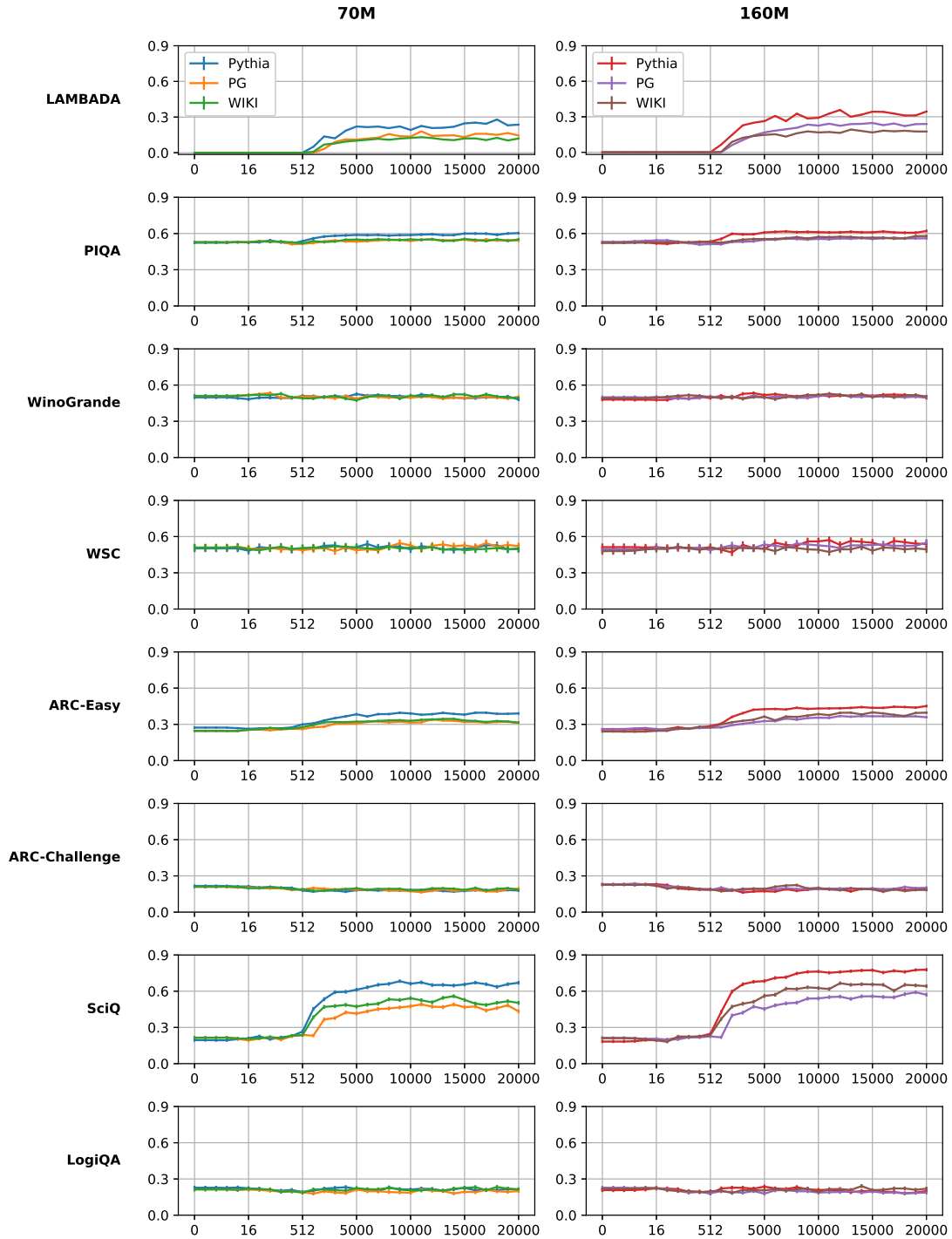


Figure 3.9: Linguistic benchmark evaluation for our model trained from scratch on specific datasets. In the left column we report the 70M models, whereas in the right column the results for the 160M models; for each task we report the accuracy with error bars. Each line represents a different model: in blue the original Pythia 70M model; in orange the 70M model trained on the Project Gutenberg dataset; in green the 70M model trained on Wikipedia; in red the original 160M model; in purple the 160M model trained on Project Gutenberg; in brown the 160M model trained on Wikipedia.

Based on these corpora, four models are trained: two with 70M parameters, one on Project Gutenberg (hereafter *PG-70*) and one on Wikipedia (*Wiki-70*); and two with 160M parameters, referred to as *PG-160* and *Wiki-160*, respectively. For consistency in nomenclature, the original Pythia models trained on The Pile are denoted as *Pythia-70* and *Pythia-160*.

Following the linguistic evaluation protocol introduced in [Biderman et al. \(2023\)](#), the trained models are assessed on the same set of linguistic benchmarks. These include: (i) LAMBADA ([Paperno et al., 2016](#)), which evaluates language modeling and contextual prediction by requiring the prediction of passage endings; (ii) Physical Interaction Question Answering (PIQA) ([Bisk et al., 2020](#)), designed to test physical commonsense reasoning; (iii) WinoGrande ([Sakaguchi et al., 2020](#)), a large-scale benchmark for coreference resolution; (iv) the Winograd Schema Challenge (WSC) ([Levesque, 2011](#)), targeting commonsense reasoning and pronoun resolution; (v–vi) the AI2 Reasoning Challenge (ARC) ([Clark et al., 2018](#)), which includes both Easy and Challenge subsets involving multi-step reasoning over scientific questions; (vii) Science Question Answering (SciQ) ([Welbl et al., 2017](#)), a multiple-choice benchmark assessing understanding of basic scientific concepts; and (viii) LogiQA ([Liu et al., 2020](#)), a dataset focused on logical reasoning and deductive inference.

The evaluation results, reported in Figure 3.9, indicate that models retrained on Project Gutenberg and Wikipedia largely replicate the behavior of the original Pythia models across benchmarks. Performance differences are minimal for WinoGrande, WSC, ARC-Challenge, and LogiQA, suggesting comparable levels of coreference resolution and reasoning ability. In contrast, for LAMBADA, PIQA, ARC-Easy, and SciQ, the original Pythia models achieve slightly higher scores than the retrained variants, consistent with the advantage conferred by training on a larger and more diverse corpus.

Across both best-layer and last-layer evaluations, shown in Figure 3.10, models trained on Project Gutenberg do not outperform their counterparts trained on Wikipedia. In the figure, dotted lines denote the 70M models, while solid lines correspond to the 160M models.

For hyperbole both PG and Wiki models converge to nearly identical compression values. In the best-layer analysis, compression stabilizes around 1.66 for the 70M models and 1.72 for the 160M models. A similar pattern is observed at the last layer, with no meaningful divergence between the two training regimes. The same trend holds for the oxymoron. Both PG and Wiki models reach compression values of approximately 2.30 for both 70M and 160M architectures, regardless of whether the best or the final layer is considered. This consistency indicates that the literary nature of the Project Gutenberg corpus does not provide a measurable advantage over the more generic and encyclopedic

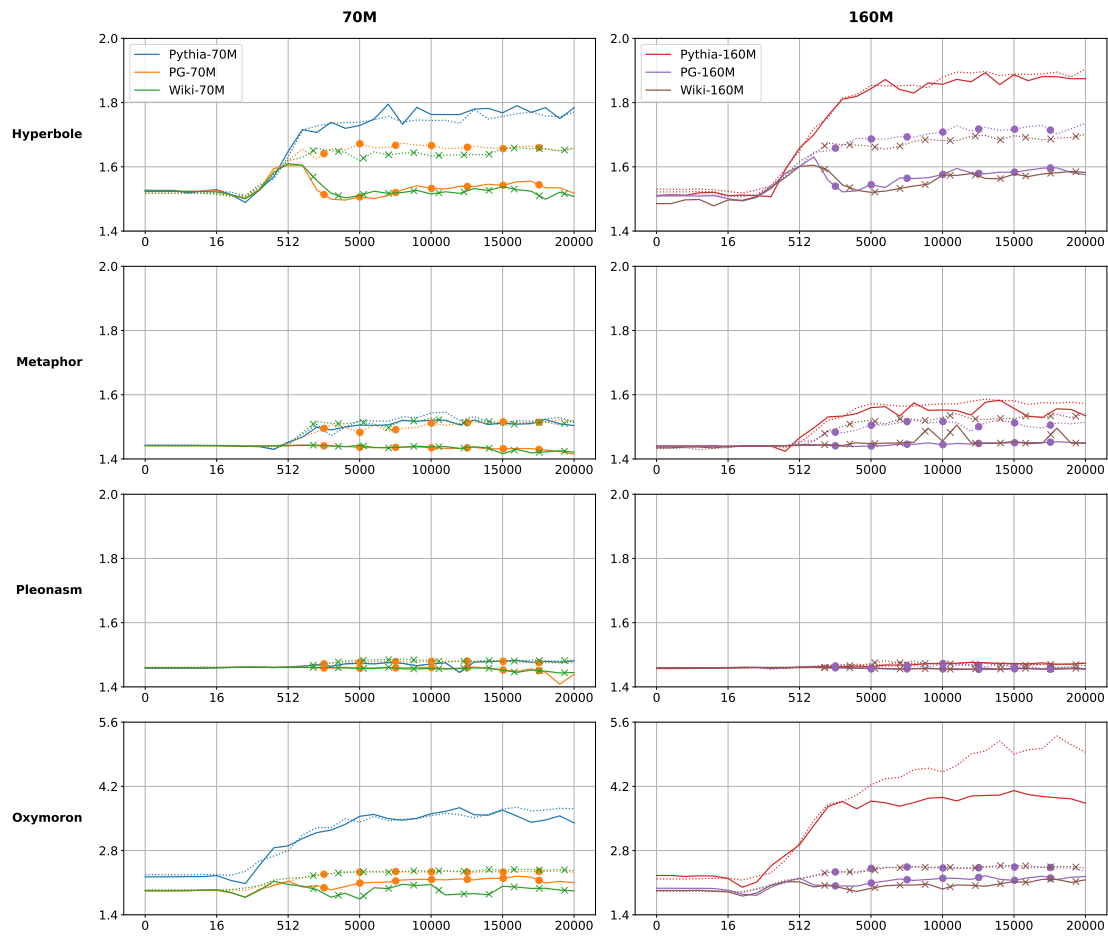


Figure 3.10: Learning trajectories of the last layer (solid line) and the best layer (dotted line) for the GPT-NeoX models trained from scratch. In the left column, the 70M models are shown, whereas in the right column the 160M models are shown. For each task, the compression value is reported for the first 20K iterations. Each line represents a different model: in blue the original 70M model; in orange the 70M model trained on the Project Gutenberg dataset; in green the 70M model trained on Wikipedia; in red the original 160M model; in purple the 160M model trained on Project Gutenberg; in brown the 160M model trained on Wikipedia. The dot in the PG model lines and the cross in the Wiki model lines represent the completion of an epoch.

Wikipedia corpus in terms of figurative language acquisition.

For metaphor and pleonasm, neither dataset leads to notable learning effects. Compression values remain low and close to chance across both layers and model sizes, and no systematic differences emerge between Project Gutenberg and Wikipedia. These results mirror the broader difficulty of these figures of speech observed throughout the study.

This consistency indicates that the literary nature of the Project Gutenberg corpus does not provide a measurable advantage over the more generic and encyclopedic Wikipedia corpus in terms of figurative language acquisition.

In contrast, models trained on the original Pythia corpus consistently achieve higher compression values across both layers and figures of speech. For hyperbole, Pythia-70

reaches a compression of 1.77, while Pythia-160 reaches 1.90, outperforming both PG and Wiki models. For oxymoron, the difference is substantially larger: Pythia-70 achieves a compression of 3.72, and Pythia-160 reaches 4.94, far exceeding the corresponding PG and Wiki results.

This advantage is evident in both the best-layer and last-layer analyses, which show that Pythia models not only reach higher peak compression but also maintain more stable learning trajectories. In contrast, PG and Wiki models tend to stabilize early, often after the first epoch, with limited subsequent improvement.

A plausible explanation for this difference lies in the size and diversity of the training data. The original Pythia corpus is substantially larger than both Project Gutenberg and Wikipedia, enabling continuous exposure to novel linguistic contexts throughout training. Conversely, training on PG and Wiki requires multiple passes over the same data (approximately 7 epochs for Project Gutenberg and 11 epochs for Wikipedia), which contributes to early saturation and limits further representational refinement.

Chapter 4

Implementing LLMs

This chapter presents the work conducted in collaboration with **Mega Italia Media S.p.A.**¹, which co-funded my PhD scholarship. Mega Italia Media is an Italian company operating since 1988 in the corporate training sector, with a strong specialization in mandatory workplace health and safety training, developing and delivering legally regulated eLearning courses to employees of hundreds of client companies. In this context, the term *learner* refers to an employed worker who is required by law to complete training courses ranging from general safety instruction to highly specialized courses, supported by digital services and tools designed to enable large-scale training delivery.

A central element of this ecosystem is *DynDevice LMS*², Mega Italia Media's Learning Management System (LMS), conceived to support the creation, delivery, and documentation of corporate training across heterogeneous modalities, including asynchronous eLearning, classroom sessions, and synchronous web conferences. DynDevice was initially released in 2011 and has since evolved through continuous updates into a modular framework that supports course authoring, training record management, and integrations with complementary organizational services.

Each training course delivered through the platform concludes with a mandatory assessment test, which can consist of either closed-ended questions, automatically graded by the system, or open-ended questions whose evaluation traditionally requires manual correction by human tutors. Passing this test is a necessary condition for learners to obtain the legally valid certificate of course completion, and therefore, the timing of the evaluation directly affects the overall training workflow. In the case of open questions, learners must wait for manual grading before receiving the final outcome of the course, while tutors are required to dedicate a significant portion of their working time to repetitive correction tasks, reducing the time available for other instructional or supervisory activities. In this context, the **automatic scoring system**, described in Section 4.1, focuses on automatic assessment of open-ended learners' answers, with the objective of reducing the manual grading burden on tutors and accelerating course completion procedures for learners, while preserving reliable evaluation standards, high-

¹<https://www.megaitaliamedia.com/it/>

²<https://www.dyndevic.com/it/>

quality feedback and full auditability of assessment decisions.

The second system focuses on the automatic classification and routing of customer assistance tickets generated within the learning platform. When learners encounter issues during course delivery, they can request support by opening a ticket, which may describe heterogeneous problems ranging from course-related questions to technical malfunctions of the platform. These tickets are traditionally reviewed and manually assigned to support staff, a process that is time-consuming and prone to suboptimal routing decisions, as requests are not always directed to the most appropriate personnel. Thus, the **automatic ticket assignment system**, described in Section 4.2, aims to automate the initial ticket classification and assignment process, with the dual objective of reducing the manual workload of support coordinators and improving routing accuracy. By directing tickets more consistently to the appropriate area of competence, the system seeks to shorten response times and increase the overall efficiency of the support workflow.

Taken together, the two systems described in this chapter are part of a coordinated effort to improve the overall experience of learners using the Mega Italia Media platform while at the same time optimizing internal workflows for company staff.

4.1 Automatic Scoring for Workplace Safety Courses

The eLearning courses rely on assessment tests to certify learning outcomes, particularly in regulated domains such as workplace health and safety. In this context, the assessment is not a marginal component, but a mandatory step that determines whether training obligations are formally fulfilled. Although multiple-choice questions can be evaluated automatically with minimal effort, open-ended questions still require human judgment.

Each assessment test typically includes multiple open-ended questions, which may require short factual answers or more articulated explanations based on technical regulations and course material. For each question, the instructors provide a reference answer to support the evaluation process. In practice, these reference answers are often detailed and considerably longer than the responses submitted by learners, making manual comparison slow and effort-intensive. During correction, evaluators must check whether the required key concepts are present in the learner's response, even when they are expressed with different terminology or phrasing, rather than relying on simple textual overlap. The operational impact of this requirement becomes evident on the scale. In the DynDevice eLearning platform developed by Mega Italia Media, open-ended answers are manually corrected by domain experts, with an average correction time of 10 to 15 minutes per test. This issue has become more prominent over the last years; in fact, as reported in Figure 4.1, the number of tests has increased significantly over the years,

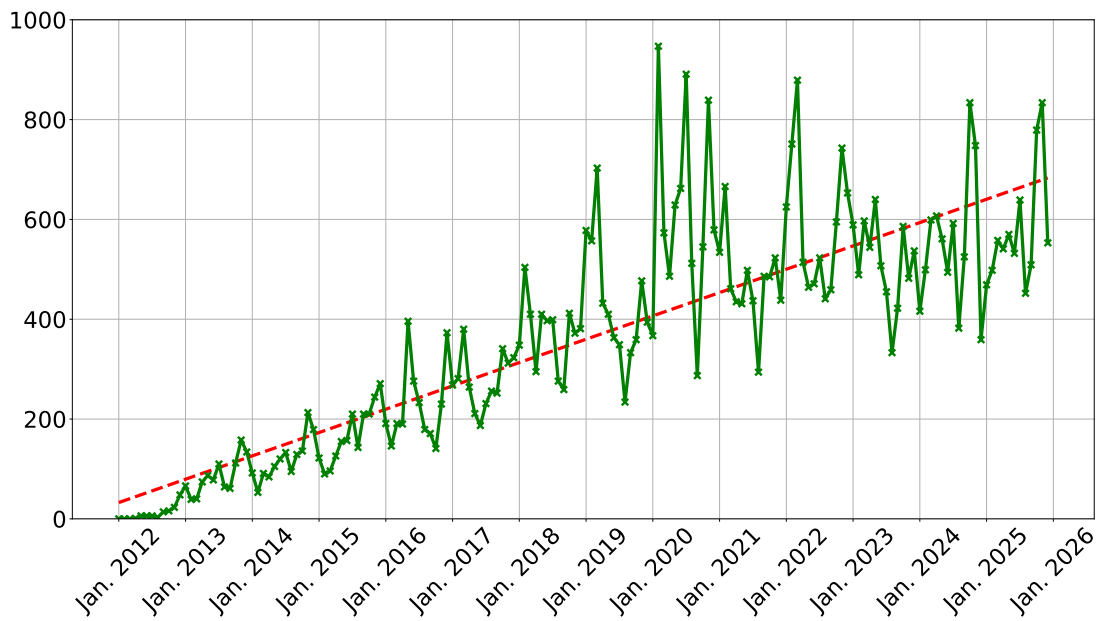


Figure 4.1: Temporal evolution of the number of assessment tests processed on the DynDevice platform from 2012 to 2026. In red is reported a clear increasing trend over time, with a marked acceleration in recent years, highlighting the growing operational burden associated with manual correction of open-ended questions.

from 123 in 2012, the year DynDevice was launched, to 2034 in 2014, to more than 7000 after 2020.

These issues motivate the development of an automatic scoring system designed not to replace human evaluators, but to support them by accelerating routine corrections while preserving expert control over final decisions. The scoring system described in this section is explicitly grounded in real-world deployment requirements, balancing predictive performance with robustness, interpretability, and seamless integration into an existing industrial eLearning workflow.

4.1.1 Dataset and Task Formulation

The dataset originates from a real industrial eLearning platform and reflects the constraints and noise typical of operational environments, rather than a curated benchmark. The data were extracted from the Mega Italia Media database, which contains more than 45000 questions collected over several years. All textual content, including questions, reference answers, and learner responses, is written in Italian. Among these, slightly more than one thousand are open-ended questions accompanied by a reference answer provided by course instructors. After removing duplicates, the set is reduced to 378 unique open-ended questions. The database also stores approximately two million learner responses, of which about 135000 correspond to open-ended questions. Filtering

Type	Text
Question	Can the Head of Prevention and Protection be affected by the consequences of a personal data breach?
Reference Response	Yes, the Head of Prevention and Protection may be involved because the breach could cause physical and psychological harm to the person concerned. He must therefore deal with the possible consequences of inappropriate processing, with possible data breaches.
Learner Response	Yes, for violations concerning data at your disposal
Score	100

Table 4.1: Example of an open-ended assessment item from the workplace safety domain, including the question, the reference answer provided by the instructor, and a learner response. All texts are originally in Italian and are reported here in English translation.

out duplicate entries and responses without an associated score yields a final pool of 7315 scored learner responses. An example consisting of a question, its reference answer, and a corresponding learner response is reported in Table 4.1, together with the score assigned by the human evaluator on a scale from 0 to 100.

An analysis of the textual length reveals systematic differences among questions, learner responses, and reference answers. Questions are typically much shorter than answers, while learner responses are, on average, substantially shorter than the corresponding reference answers; the median number of words is, respectively, 45 and 67. This discrepancy reflects instructional practice: reference answers are designed to be exhaustive and include all relevant information required for full credit, whereas learners tend to express the same concepts more concisely. As a result, both human evaluators and automatic scoring systems cannot rely on direct textual comparison alone, but must assess whether the required key points are present despite differences in length and wording. Furthermore, the score distributions in the dataset are strongly skewed toward high values: at least 75% of the answers receive a score of 80 or higher, and approximately 25% achieve the maximum score.

The initial task formulation followed the human grading procedure by attempting to predict a score between 0 and 100. However, given the strong imbalance in the score distribution, this approach was not suitable. In this setting, a regression model would be driven mainly by high-score examples, with limited information available to learn how to handle low-quality answers. For this reason, the task was reformulated as a binary classification problem, where the model predicts whether a learner response exceeds a given sufficiency threshold. When adopting the standard Italian passing threshold of

60/100, the dataset remained highly unbalanced, with only about 8% of the responses falling below the threshold. Raising the threshold to 80 resulted in a more balanced distribution, with approximately 28% of responses classified as insufficient, making this formulation more suitable for training and evaluation. This binary setup also aligns with the operational constraints of regulated training contexts. Automatic systems cannot autonomously certify the results of the assessment and the final validation must always be performed by a human evaluator. The model is therefore intended to assist tutors by identifying responses that are likely sufficient or insufficient, rather than producing final scores without human oversight.

To facilitate validation by human tutors and avoid direct comparison between reference answers and learner responses, a different input representation was introduced. Instead of using the full reference answer, each question is associated with a set of key *concepts*. These concepts are expressed as short declarative sentences, typically between five and ten words, each representing a single required piece of information. For reference, the concepts associated with Table 4.1 are as follows:

1. Yes, for violations concerning data at your disposal
2. the violation could cause physical and psychological harm to the person involved
3. must deal with the possible consequences

Each concept is assigned a predefined score, and the sum of the scores associated with a question equals 100.

Under this formulation, the model is trained to predict, independently for each concept, whether the information expressed by that concept is present in the learner's response. Then, a post-processing step aggregates the model predictions by summing the scores of the concepts predicted as present, yielding an overall score estimate for the answer. In the operational workflow, the human evaluator reviews the predicted presence of each concept, confirms or corrects it when necessary, and the final score is automatically updated accordingly. This setup reduces manual effort while keeping the final assessment fully under human control.

Due to time constraints, domain experts were able to extract concepts for only 67 questions, producing a total of 114 concepts. Questions without extracted concepts were excluded, resulting in a final dataset of 4,117 responses from learners. Each answer may be associated with multiple concepts; in such cases, the record is duplicated, with each instance paired with a different concept. This procedure yields a total of 10,560 concept-level instances. Finally, the dataset is divided into training, validation, and test sets using an 80%/10%/10% partition.

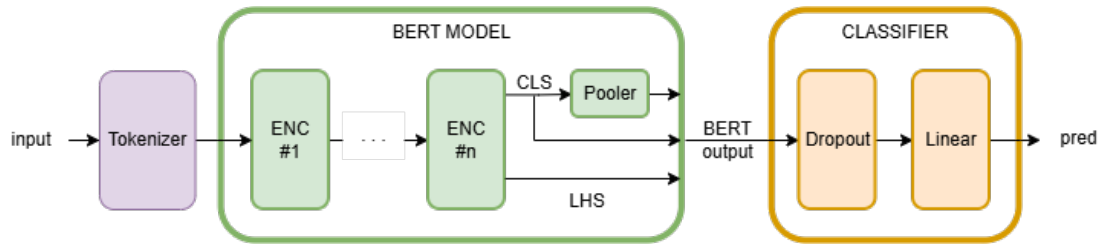


Figure 4.2: Overview of the neural architecture adopted for automatic scoring. A pre-trained BERT encoder processes the tokenized input sequence and produces contextual representations. The embedding derived from the encoder output as the pooler output, the original [CLS] token, or a manipulation (mean, min or max) of the last hidden states (LHS), is then passed through a lightweight classification head, consisting of a dropout layer followed by a linear layer, to generate the final binary prediction.

4.1.2 Model Architecture

This section describes the neural architecture underlying the automatic scoring system, together with the input configurations and training strategies explored to adapt pre-trained language models to the target task.

The architecture, shown in Figure 4.2, is made up of two main components: (i) the BERT language model, to create the sentence representation, and (ii) a lightweight classification head made of a dropout layer followed by a single linear feed-forward layer, which maps the sentence-level representation to a binary prediction (label 1 if the response has a score greater than the threshold of 80 out of 100 or 0 otherwise). The complete architecture follows a standard fine-tuning paradigm, in which the pre-trained encoder is reused and task-specific parameters are learned with minimal additional complexity.

Starting from this base architecture, multiple model variants are defined that differ only in how the input text is constructed. The simplest configuration, called the **Reference Model (REF)**, concatenates the learner’s answer with the reference answer. This setup frames the task as a direct comparison between the two texts. To provide additional contextual information, **Question–Reference Model (QREF)** prepends the original question to this input, allowing the encoder to condition similarity judgments on the instructional prompt.

A parallel set of configurations is introduced for concept-level scoring. In **Concept Model (CONC)**, the learner’s answer is paired with a single concept extracted from the reference answer, and the model predicts whether that concept is expressed in the response. The **Question–Concept Model (QCONC)** further increases this input by including the text of the question. Across all configurations, reported in Figure 4.3, text segments are concatenated using the standard [SEP] token, giving a uniform interface to the encoder.

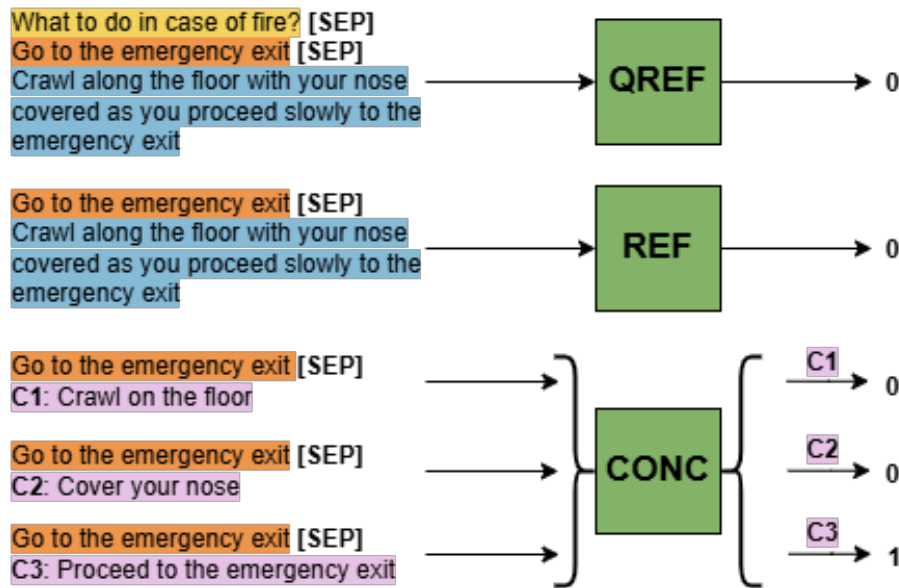


Figure 4.3: Schematic overview of the different input configurations used for automatic scoring. In the **REF** configuration, the learner’s answer, highlighted in orange, is concatenated with the reference answer, highlighted in blue, while **QREF** additionally includes the original question, highlighted in yellow. For concept-level scoring, the **CONC** configuration pairs the learner’s answer with a single concept, in purple, extracted from the reference answer. For clarity and space constraints, the **QCONC** configuration is not explicitly shown, as its structure is analogous to **QREF**, with the question text prepended to the **CONC** input. In all cases, text segments are concatenated using the standard [SEP] token.

All model variants are trained using the same optimization strategy. The objective function is the binary cross-entropy loss, optimized with the AdamW algorithm. Given the pronounced class imbalance in the dataset, model selection and validation are based on the F1-score rather than accuracy.

Beyond input configuration, extensive experimentation is conducted on the choice of the pre-trained encoder, exploring three major model families: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2021), and ELECTRA (Clark et al., 2020). Several Italian BERT variants³ are evaluated, including cased and uncased models, as well as base and XXL versions trained on different corpora. In addition, multilingual BERT models⁴, multilingual RoBERTa⁵, and an Italian ELECTRA⁶ are tested. Although these models share a broadly similar Transformer backbone, they differ substantially in pre-training data and objectives.

To further adapt the models to the workplace safety domain, two forms of domain specialization are explored. First, continued masked language model pre-training is

³<https://huggingface.co/dbmdz/>

⁴<https://huggingface.co/bert-base-multilingual-uncased>

⁵<https://huggingface.co/xlm-roberta-base>

⁶<https://huggingface.co/dbmdz/electra-base-italian-xxl-cased-discriminator>

performed on approximately 18,000 domain-specific sentences extracted from course subtitles. Second, a domain-specific vocabulary extension is introduced. An analysis of the fine-tuning data reveals that a large fraction of domain-relevant terms are split into multiple sub-word units, potentially degrading semantic representations. To address this issue, frequently occurring domain-specific words, such as *isolanti* (insulators), *amianto* (asbestos), or *ustioni* (burns), are added to the tokenizer vocabulary, and their embeddings are trained during additional masked language model pre-training.

Finally, several strategies are explored to compute a fixed-size sentence representation from the encoder outputs. Specifically, we considered: (i) the **pooled output** produced by the BERT pooler layer; (ii) the raw embedding of the **special classification token** [CLS]; and (iii) aggregated representations derived from the last hidden state matrix, obtained by applying element-wise **mean**, **maximum**, or **minimum** operations over the token embeddings. Additional architectural and optimization choices, such as maximum sequence length (128 or 512), dropout rate (choose from 0.1, 0.2 and 0.4), class balancing through subsampling, and freezing versus fine-tuning BERT, are explored systematically with a grid search.

4.1.3 Experimental Results

This section presents the experimental results of a systematic evaluation of the proposed scoring architectures, with the primary objective of identifying the most effective model configuration for the task.

Given the large number of evaluated configurations, results are organized by grouping models according to the type of pre-trained encoder. Specifically, we distinguish among: (i) Italian-only BERT models without additional adaptation; (ii) multilingual BERT models; (iii) Italian BERT models further adapted through continued pre-training, with and without domain-specific vocabulary extensions; and (iv) alternative Transformer architectures, namely ELECTRA and RoBERTa. For each group, only the best-performing configuration is reported to facilitate comparison.

Across groups, as reported in Figure 4.4, Italian BERT models achieve the strongest overall performance, reaching a 0-F1-Score of approximately 0.688. Comparable results are obtained by models subjected to continued domain-specific pre-training, suggesting that additional masked language model training and vocabulary extension provide only marginal gains in this setting. Multilingual BERT models exhibit performance on par with Italian-only models, an outcome plausibly explained by the substantial presence of Italian data in the multilingual pre-training corpus. This result indicates that multilingual encoders represent a viable option when extending the system to additional languages without a substantial loss in accuracy.

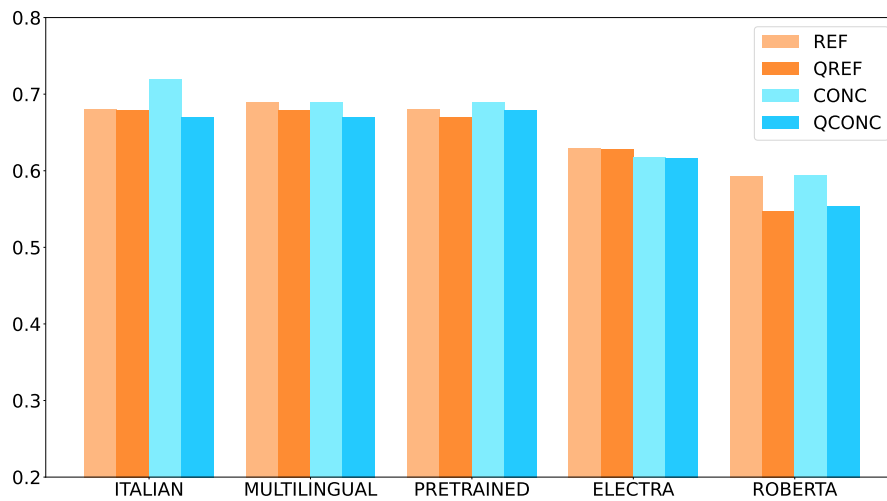


Figure 4.4: Comparison of the best-performing models across encoder families and input configurations. Results are reported in terms of 0-F1-Score for Italian, multilingual, domain-pretrained BERT, ELECTRA, and RoBERTa models. Reference-based models are shown in orange, while concept-based models are shown in blue; darker shades indicate configurations in which the question text is included in the input.

In contrast, ELECTRA and RoBERTa models consistently underperform BERT-based architectures, achieving 0.623 and 0.572 respectively. This degradation is particularly evident for RoBERTa, which achieves the lowest scores among the models tested. A likely explanation lies in architectural and pre-training differences: unlike BERT, these models do not explicitly optimize the [CLS] representation through a next-sentence prediction objective, which may reduce their effectiveness for sentence-level classification tasks.

When comparing input formulations, including the question text in the input does not lead to systematic improvements. In most cases, QREF and QCONC variants perform worse than their counterparts that exclude the question. A plausible explanation is that the longer input sequence introduces additional context that interferes with BERT’s ability to focus on the semantic comparison between the two responses. In contrast, concept-based models achieve performance comparable to or exceeding that of reference-answer-based models. Beyond raw metrics, concept-based formulations offer an important advantage in terms of interpretability: they allow evaluators to inspect which individual concepts are detected as present or absent, rather than providing a single opaque judgment.

Regarding sentence representation, aggregating information from all token embeddings proved more stable than relying on the pooled output or the raw [CLS] vector, with simple pooling strategies on the final hidden states performing best overall. Attempts to balance the training data through subsampling did not yield consistent improvements and often reduced performance due to the smaller training set size, while other hyperparameters, such as dropout rate and maximum sequence length, showed no substantial

Model	BERT	Sentence Rep.	Seq. Len.	Balanced	Dropout	0-F1	1-F1
CONC	IT-XXL-UNC	Pooler	512	True	0.1	0.72	0.87
CONC	PRET-V	Min LHS	128	True	0.1	0.69	0.87
CONC	ITA-UNC	Min LHS	512	False	0.4	0.69	0.86
CONC	MULTI-UNC	Pooler	512	True	0.2	0.69	0.86
REF	MULTI-UNC	Min LHS	512	True	0.2	0.69	0.85

Table 4.2: Best-performing configurations in terms of F1-Score, with best results highlighted in bold. IT-XXL-UNC denotes the Italian XXL uncased version of BERT, PRET-V the domain-pretrained BERT model with expanded vocabulary, and MULTI-UNC the multilingual uncased BERT. The freeze hyperparameter is omitted, as all reported configurations are fully fine-tuned.

change in results.

As reported in Table 4.2, the highest-performing configuration is a CONC model built on the Italian XXL uncased BERT encoder, which reaches a 0-F1-Score of approximately 0.72. This result consistently exceeds that of the best reference-based model, whose performance plateaus at around 0.69. These trends are observed across multiple encoder variants and input settings.

Taken together, these results indicate that concept-based architectures built on Italian or multilingual BERT encoders, fine-tuned end-to-end and using token-level pooling representations, provide the most effective and operationally suitable solution for automatic scoring in this setting. Based on these results, the best configuration was selected for deployment and is the one evaluated in the following section through real-world integration and usage.

4.1.4 Real World Implementation

This section describes the real-world deployment of the proposed automatic scoring system within the Mega Italia Media eLearning platform and evaluates its impact under operational conditions. Unlike the controlled experimental setting discussed earlier, this phase focuses on system integration, human–AI interaction, and practical effectiveness in reducing correction time and user waiting periods.

The scoring system has been integrated into DynDevice, the proprietary eLearning platform developed by Mega Italia Media. DynDevice is a web-based application implemented in PHP and JavaScript, accessible through standard browsers and mobile devices. After completing a course, users are required to take an assessment test. Tests composed exclusively of multiple-choice questions are corrected automatically and returned almost instantaneously, whereas tests containing open-ended questions traditionally require

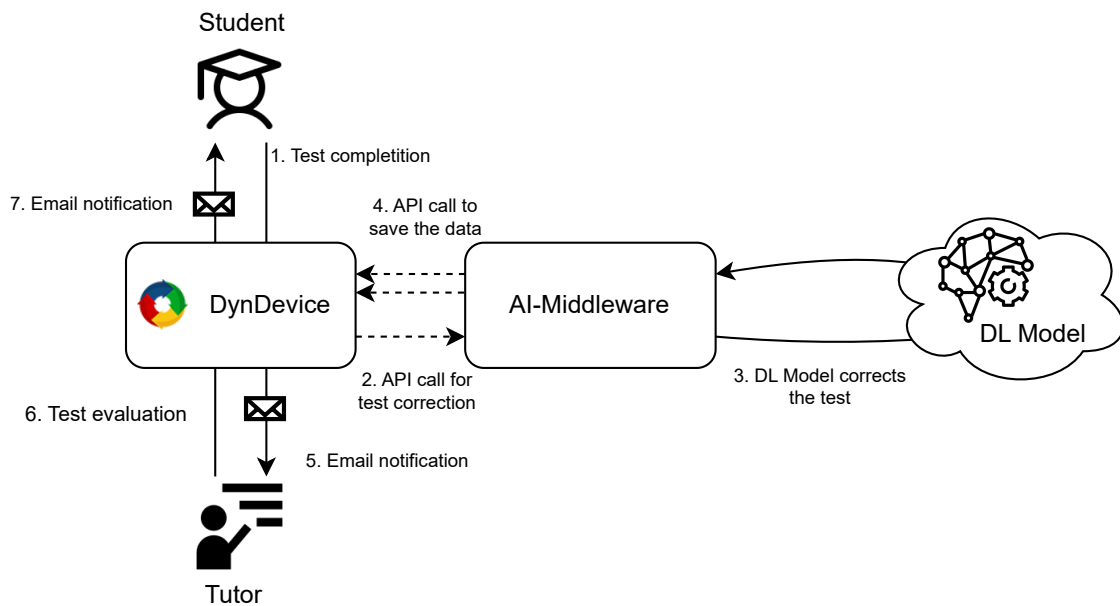


Figure 4.5: Architecture implemented for test correction. It is composed of three main components: the DynDevice platform, the AI-Middleware, and the deep learning model. The arrows represent how these components interact with each other and with the users (both the learner and the tutor). Straight arrows indicate interactions through a web platform or email, whereas dashed arrows indicate API calls. The numbered steps required by the architecture to correct a test are also shown.

manual correction by human tutors. In the original workflow, tutors were notified by email upon test submission and were responsible for evaluating all open-ended responses before results could be released to users.

The introduction of the automatic scoring system modifies this workflow by inserting an intermediate software layer, called *AI-Middleware*. This component is implemented as a lightweight Python service using the Flask framework and is responsible for managing communication between the DynDevice platform and the deep learning models. The middleware encapsulates all model-related operations, thus decoupling the web platform from the underlying machine learning infrastructure.

A key design constraint arises from the client–server nature of the platform. Since DynDevice is a web application, prolonged blocking of client requests is undesirable and may incur penalties related to search engine optimization and user experience. Test evaluation times can range from several seconds to more than a minute, depending on the number of open-ended responses, the number of associated concepts, and the conditions of the network. To address this constraint, the system adopts a dual-API communication strategy. Instead of keeping the client waiting for a response, DynDevice sends an asynchronous request to the AI-Middleware to initiate test correction. Once processing is complete, the middleware independently communicates the results back to

	Precision	Recall	F1-Score	Support
Concept Absent	0.35	0.54	0.43	360
Concept Present	0.90	0.81	0.85	1851
macro avg	0.63	0.67	0.64	2211
weighted avg	0.81	0.76	0.78	2211

Table 4.3: Evaluation of the best concept-based models on tests submitted by users in the last six months. The support column indicates the number of instances belonging to each class and, for averaged metrics, the total number of test instances.

DynDevice through authenticated API calls.

The resulting correction workflow consists of the following steps shown in Figure 4.5. After a user submits a test, DynDevice sends the relevant answers and associated concepts to the AI-Middleware in the form of a JSON payload. The middleware applies the trained concept-based scoring model to predict the presence or absence of each concept. The predictions are then transmitted back to DynDevice and stored in the company database. The pre-evaluated test is forwarded to a human tutor, who reviews the suggested corrections and confirms or overrides them as required. Finally, the user is notified of the test outcome by email. Although this workflow introduces additional technical steps compared to the original process, it substantially reduces the time required for human correction.

Real World Evaluation

Following deployment, the system has been used continuously for several months to assist in the evaluation of open-ended tests, while the company progressively expanded the set of questions for which concepts were defined. To assess real-world performance, model predictions were compared against human judgments of concept presence. The results, summarized in Table 4.3, reveal a pronounced class imbalance, with a relatively small proportion of concepts labeled absent by human evaluators (about 16%). Despite this imbalance, the system achieves strong performance in identifying present concepts, while the performance on absent concepts is lower.

These findings are broadly consistent with the offline experimental results, although a degradation in performance is observed in the negative class. Two factors likely contribute to this effect. First, the deployed system encounters new questions and concepts that were not present during training, increasing task difficulty, and reducing generalization performance. Second, the labeling procedure differs between the prototype dataset and real-world evaluation. During training, concept labels were inferred indirectly from

overall question scores, whereas in deployment each concept is explicitly annotated by human evaluators. This mismatch introduces noise relative to the original supervision signal and can negatively affect performance.

Despite these limitations, the system satisfies the primary operational requirement identified by the company: reliably detecting when a concept is present in a learner's response. High precision in the positive class ensures that most automatically identified concepts are correct, allowing tutors to focus their attention on more uncertain cases. As a result, human intervention is required for only a subset of predictions, significantly reducing the overall workload.

From an operational perspective, domain experts report that the introduction of the automatic scoring system reduces manual correction time by approximately two thirds. This improvement translates into tangible benefits in terms of cost and scalability. Moreover, the system has a direct impact on the user experience by reducing the waiting time between the submission of the test and the notification of the results. An internal analysis shows that average waiting times decreased from approximately 12 hours under the fully manual process to approximately 4 hours when the tutors were assisted by the automatic system. Although precise attribution of time savings to individual correction actions is difficult, the overall reduction in turnaround time represents a substantial improvement.

4.2 LLM-based Ticket Assignment

Modern providers of information technology solutions are required not only to develop and deploy software systems, but also to ensure their continuous operation throughout the software lifecycle. This includes corrective maintenance, the resolution of malfunctions, the introduction of new functionalities, and the handling of heterogeneous user requests. In industrial settings, these activities generate a persistent operational workload that must be addressed on a daily basis by specialized technicians.

User-reported issues and service requests are commonly collected through ticketing systems. Within these systems, users submit short textual reports describing encountered problems or requested services. Once submitted, tickets must be assigned to maintenance personnel responsible for addressing the request.

In organizations that manage complex software platforms or multiple products, maintenance teams are rarely homogeneous. Instead, employees are usually organized according to distinct areas of expertise, reflecting differences in technical skills, domain knowledge, and responsibilities. As a consequence, incorrect or delayed ticket assignment can lead to longer resolution times, inefficient use of resources, and increased operational

CATEGORY: *G. More on e-Learning/training solutions*

SUBJECT: *Certificate with exam in presence HTML5*

DESCRIPTION: *"Certificate with Examination in Attendance" I turned it into HTML 5. Everything is fine except for the column rows that do not appear. What could be the problem? Thank you*

AREA: *SW*

Example 4.1: Example of a support ticket from the dataset. The ticket includes the user-selected category, a short subject line, a free-form textual description of the issue, and the macro-area responsible for its resolution. Here the example is translated from Italian to English.

costs. Ensuring that each ticket is routed to the appropriate team, therefore, represents an important step in the support workflow.

Traditionally, ticket assignment is performed manually. One or more human operators read the incoming tickets, interpret the user's request, and decide which internal team should handle the issue. Although effective, this process is inherently time-consuming and scales poorly with ticket volume. For large organizations, manual triage quickly becomes a bottleneck, motivating the adoption of automated or semi-automated solutions.

Recent advances in LLMs suggest a promising alternative to support this task. Pre-trained on large-scale textual data, LLMs exhibit strong zero-shot and few-shot capabilities, enabling them to perform classification and routing tasks based on natural language instructions, with minimal or no task-specific training data. This opens up the possibility of deploying automated ticket assignment systems with a reduced development effort and lower data requirements.

To explore this potential in a realistic setting, a real-world industrial scenario drawn from the operations of DynDevice, the Mega Italia Media LMS, is considered. The platform generates a continuous stream of user-submitted support tickets, written in Italian, characterized by heterogeneous content and varying levels of technical detail. Each ticket must be assigned to one of several internal macro-areas responsible for maintenance and support, providing an operationally relevant testbed for evaluating LLM-based ticket assignment.

4.2.1 Task and Data

The automatic ticket assignment problem is formulated as a supervised multi-class classification task. Given a support ticket submitted by a user, the objective is to predict the internal macro-area responsible for handling the request.

Each ticket, as shown in Example 4.1, is represented as a tuple of textual and categorical fields, including a predefined category selected by the user, a short subject line, and a free-form natural language description. These inputs jointly encode the user's intent and

the nature of the reported issue. The output of the system is a single label drawn from a finite set of macro-areas, corresponding to distinct internal responsibilities within the organization.

In the considered operational setting, tickets are assigned to one of three macro-areas: **eLEARNING**, **TECH**, and **SW**. The eLEARNING area handles requests related to course content, learning processes, and training-related functionalities. The TECH area is responsible for technical infrastructure issues, including deployment, system configuration, and platform availability. The SW area covers software-related problems such as bugs, feature requests, and application-level malfunctions. These labels reflect stable organizational roles and are directly associated with different groups of maintenance specialists.

The dataset used is derived from the production ticketing system adopted by the company to support users of its LMS. Since the introduction of the platform in 2011, the company has progressively transitioned from direct support channels, such as phone calls and emails, to a structured ticket-based workflow in order to handle the increasing volume of user requests and maintain traceability of support interactions.

Over a period of approximately ten years, the company has collected more than 10,000 tickets. However, the platform has undergone significant evolution in recent years, including interface redesigns, the introduction of video call functionalities, and the deployment of artificial intelligence-based features. To ensure consistency between the dataset and the current operational context, only tickets submitted in the most recent months were considered. This filtering results in a dataset of approximately 1,300 tickets, reflecting the current structure, services, and usage patterns of the platform.

For each ticket, only the information relevant to the assignment task is retained. Specifically, each instance includes: (i) the ticket category selected by the user from a predefined list of eleven categories, (ii) a short subject line, (iii) a free-form textual description of the issue, and (iv) the macro-area responsible for resolving the ticket. Additional metadata, such as timestamps, ticket identifiers, and conversation history, is discarded. Please note that all tickets are written in Italian.

The category field of each ticket is selected at submission time from a predefined and fixed menu consisting of 11 categories. These categories are:

- A. Problems in using a course
- B. Inability to access a course
- C. Clarification of the didactics of a course
- D. Problems in generating reports

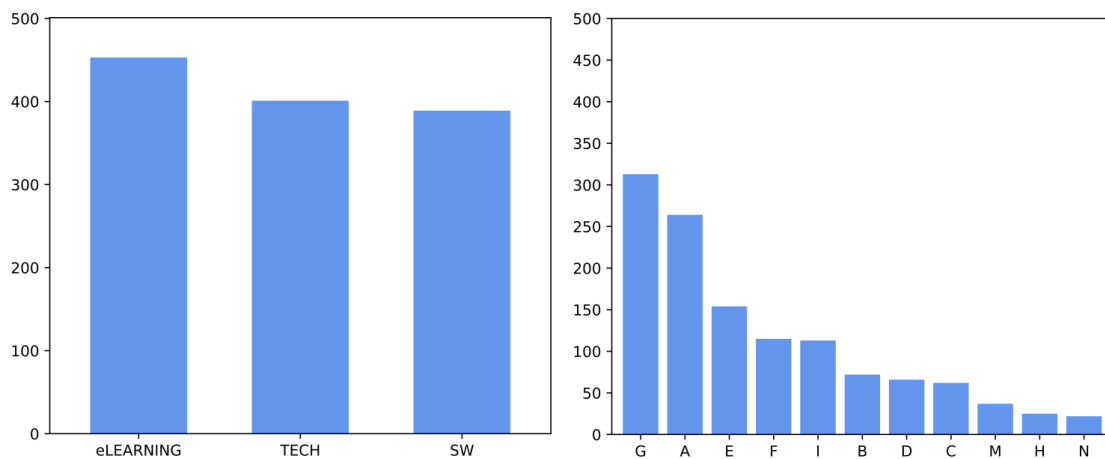


Figure 4.6: Distribution of tickets in the dataset. The left panel shows the number of tickets assigned to each macro-area (eLEARNING, TECH, and SW), whereas the right panel reports the distribution of tickets across the eleven predefined categories; for clarity only the category letter (from A to N) is reported.

- E. Problem in generating/recovering documents
- F. User registration problem
- G. Other on e-learning/training solutions
- H. HR management
- I. e-Commerce and website management
- M. School of Security
- N. Commercial

Although the ticket category represents a key piece of information for guiding the assignment process, individual categories are not uniquely associated with a single operational area and may be handled by different areas depending on the specific request.

Minimal preprocessing is applied to the textual fields. HTML tags, URLs, and special characters introduced by the ticketing system are removed, while the original wording and structure of the text are preserved. Furthermore, an anonymization procedure is applied to remove sensitive information, such as full user names and email addresses, among other personally identifiable data.

The distribution of tickets across macro-areas is approximately balanced, as shown in Figure 4.6. In contrast, ticket categories exhibit a markedly unbalanced distribution, with some categories accounting for several hundred tickets and others appearing only rarely. Category G, in particular, is the most frequent, with more than 300 instances.

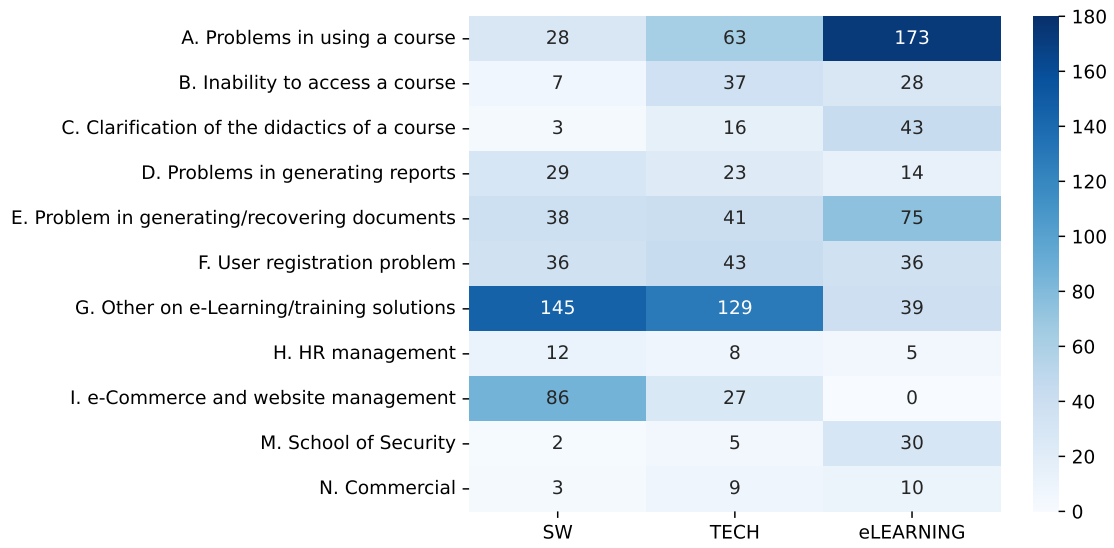


Figure 4.7: Cross-tabulation of ticket categories and macro-areas responsible for ticket resolution. Rows correspond to the eleven predefined ticket categories, while columns represent the three macro-areas (eLEARNING, TECH, and SW).

A cross-tabulation analysis among ticket categories and macro-areas reveals strong correlations for certain category–area pairs. For example, category *A* is strongly associated with the eLEARNING area, while categories such as *G* show high correlation with both TECH and SW. Other categories display weaker or no clear associations, either due to limited sample size or because tickets are evenly distributed across macro-areas. These correlations are illustrated in Figure 4.7 and are later exploited in prompt design.

Finally, for evaluation purposes, approximately 250 tickets are sampled using stratification based on macro-area distribution. Five distinct test sets are constructed, each containing 18 tickets assigned to eLEARNING, 15 to TECH, and 14 to SW. This strategy preserves both label balance and the observed category–area correlations across evaluation splits.

4.2.2 Methodology

The automatic ticket assignment problem is addressed as a multi-class text classification task, where each ticket must be routed to one of three macro-areas (eLEARNING, TECH, SW). Rather than training a task-specific classifier, this approach leverages pre-trained LLMs through natural language prompting, treating the model as a configurable decision component within an operational workflow.

All experiments are conducted using the OpenAI Chat Completions API⁷, leveraging the GPT-4 model, which supports instruction-based interaction through structured

⁷<https://platform.openai.com/docs/api-reference/chat>

You are the manager of a service center whose task is to divide tickets between the various human operators. The available operators, contained in square brackets, are as follows: [eLEARNING, TECH, SW]. The tickets are divided into categories, listed by letters of the alphabet, contained in round brackets, which are as follows: (A. Problems in using a course, B. ...). Each ticket consists of a subject and a description. Your task is to assign the ticket to the most suitable operator, answering only with the operator's name.

Example 4.2: Baseline system prompt used for zero-shot ticket assignment. The prompt specifies the classification task, enumerates the valid macro-area labels, describes the structure of the input ticket, and constrains the model output to a single macro-area name.

prompting. Each request consists of two parts: a *system prompt*, which defines the task and provides contextual information, and a *user prompt*, which contains the ticket to be classified. To minimize output variability and ensure deterministic behavior, decoding is performed with temperature set to zero and the model is constrained to output only one of the valid macro-area labels.

Three prompting paradigms are investigated: zero-shot prompting, few-shot prompting, and ensemble prompting. These paradigms differ in how task knowledge and prior information are conveyed to the model.

Zero-shot Prompting In the zero-shot setting, the model is not provided with examples. Instead, all task-relevant information is encoded directly in the system prompt. A baseline prompt, shown in Example 4.2, is used in all configurations, describing the assignment task, the available macro-areas, the ticket structure, and the expected output format. All additional zero-shot prompts extend this baseline by concatenation.

Several variants of zero-shot prompts, built over the Baseline prompt, are explored, each injecting a different type of domain knowledge:

- **Human:** includes expert-authored descriptions of the responsibilities and typical issues handled by each macro-area, reflecting institutional knowledge used by human operators. Example: *SW handles technical problems with software code, new customization and developments and ICT (Information and Communication Technologies) and SEO (Search Engine Optimization) issues.*
- **Category:** incorporates empirical assignment statistics aggregated by ticket category, specifying for each category the percentage of tickets historically assigned to each macro-area. Example: *66% of the tickets in the category "A. Problems in using a course" are assigned to eLEARNING, 24% to TECH and 11% to SW*
- **Area:** provides the inverse statistics, describing for each macro-area the distribution of ticket categories it typically handles. Example: *38% of the tickets in category A*

are assigned to *eLEARNING*, 8% in *B*, 9% in *C*, 3% in *D*, 17% in *E*, 8% in *F*, 8% in *G*, 1% in *H*, 0% in *I*, 7% in *M* and 2% in *N*.

- **Summaries:** uses short textual summaries describing the types of issues solved by each macro-area. These summaries are generated offline by GPT-4 from a small random sample of training tickets and are then injected into the prompt without exposing the original examples. Example: *eLEARNING solves problems related to the activation of courses, changes of certificates, platform access problems, cancellation of courses, issuing of certificates, downloading of certificates and approval of enrollment forms.*

These prompt variants allow the analysis of how different forms of structured and unstructured contextual information affect assignment performance in the absence of explicit examples.

Few-shot prompting As a second approach, few-shot prompting is employed by augmenting the Baseline prompt with a small number of labeled examples. Each example follows the same format as real tickets and includes the correct macro-area assignment. No additional contextual information beyond the baseline task description is provided.

This setup allows the model to infer decision patterns directly from exemplars, rather than from explicit domain descriptions or statistical priors. The few-shot setting is evaluated with varying numbers of examples, with 1, 3, 5, and 10 tickets, in order to assess its robustness and data efficiency in comparison to zero-shot prompting.

Ensemble Prompting Finally, an ensemble prompting strategy is introduced to combine the strengths of different prompt configurations. For each ticket, multiple independent model calls are issued using different prompts, while keeping the model and decoding parameters fixed. Each call produces a single macro-area prediction, and the final assignment is determined by majority voting across prompts. In the case of a tie, the assignment is resolved by uniform random selection among the tied macro-areas.

This ensemble approach avoids excessively long prompts while enabling the aggregation of heterogeneous sources of information. In the experimental evaluation, ensembles composed of three and four prompt variants are considered, selected based on their individual performance in previous experiments.

4.2.3 Experimental Results

This section reports the results of the experimental evaluation of the proposed LLM-based ticket assignment strategies. Performance is assessed using two standard classification

Approach	Prompt	Accuracy	Macro F1
Zero Shot	Baseline	0.34 ± 0.05	0.29 ± 0.05
Zero Shot	Summaries	0.50 ± 0.07	0.48 ± 0.06
Zero Shot	Categories	0.53 ± 0.06	0.49 ± 0.09
Zero Shot	All Information	0.54 ± 0.05	0.49 ± 0.07
Zero Shot	Areas	0.56 ± 0.03	0.50 ± 0.04
Zero Shot	Human	0.57 ± 0.03	0.54 ± 0.04
Few Shot	One Example	0.19 ± 0.05	0.14 ± 0.05
Few Shot	Three Examples	0.44 ± 0.02	0.36 ± 0.04
Few Shot	Five Examples	0.44 ± 0.05	0.34 ± 0.05
Few Shot	Ten Examples	0.56 ± 0.04	0.51 ± 0.04
BERT	Only Ticket	0.62 ± 0.07	0.62 ± 0.07
BERT	Full	0.65 ± 0.07	0.65 ± 0.07
Ensemble Learning	Three Prompts	0.57 ± 0.05	0.52 ± 0.06
Ensemble Learning	Four Prompts	0.61 ± 0.05	0.55 ± 0.08

Table 4.4: Results of the automatic ticket assignment experiments. The table reports accuracy and macro-averaged F1-Score (mean ± standard deviation) computed over five stratified test sets for zero-shot, few-shot, ensemble prompting strategies, and the supervised BERT baseline. Best results within each group are highlighted in bold.

metrics: accuracy and macro-averaged F1-Score. For each configuration, results are reported as the mean and standard deviation computed over five stratified test sets. A summary of all results is reported in Table 4.4.

Two baseline systems are considered for comparison. The first is a zero-shot LLM configuration using only the baseline prompt, which provides a minimal task description without additional contextual information. This configuration achieves low performance, with an accuracy of approximately 0.34 and a macro F1-Score of 0.29, confirming that naive zero-shot prompting is insufficient for this task.

The second baseline is a supervised approach based on fine-tuning a pre-trained Italian BERT model. The model is fine-tuned on 907 labeled tickets for 20 epochs, with model selection performed on a validation set of 101 examples. A lightweight feed-forward classification head with three output neurons is added on top of BERT, preceded by a dropout layer. Two input configurations are evaluated: one using only the ticket description, and one concatenating category, subject, and description. Overall, the BERT-based classifier achieves substantially higher performance, with accuracy and macro F1-Scores of approximately 0.65, and serves as a strong supervised baseline.

Injecting structured information into zero-shot prompts consistently improves performance over the baseline configuration. Prompts encoding category–area correlations yield moderate gains, with the Categories and Areas prompts reaching accuracies of approximately 0.53 and 0.56, respectively. Their limited effectiveness is likely due to the reliance on extensive numerical information and the absence of explicit semantic descriptions of ticket content.

More stable results are obtained when semantic information is provided. The Human prompt, which includes expert-authored descriptions of macro-area responsibilities, achieves the best zero-shot performance (approximately 0.57 accuracy and 0.54 macro F1) and the lowest variance. In contrast, the Summaries prompt performs less reliably, with lower accuracy and higher variance, likely because the summaries are generated from a small and potentially unrepresentative sample of tickets. Combining all zero-shot information into a single prompt leads to a performance drop, suggesting that excessive prompt length hinders effective information integration.

Few-shot prompting proves ineffective with very small numbers of examples. With a single example, performance degrades substantially, falling below the zero-shot baseline. Increasing the number of examples yields only modest improvements. Even with ten examples, accuracy reaches approximately 0.56, remaining comparable to but not clearly surpassing the best zero-shot configurations. This pattern suggests that the high variability of ticket content limits the effectiveness of small exemplar sets.

The ensemble prompting strategy yields the strongest results among LLM-based approaches. By aggregating predictions from multiple zero-shot prompts through majority voting, the ensemble exploits complementary information without increasing prompt length. An ensemble of three prompts reaches approximately 0.57 accuracy, while extending it to four prompts improves performance to about 0.61, with lower variance than the supervised BERT baseline. This demonstrates that ensemble prompting can substantially reduce the gap among zero-shot LLM methods and supervised fine-tuning, while avoiding the need for labeled training data.

A final experiment compares the performance of GPT-4 and GPT-3.5 under identical prompting configurations. This comparison is motivated not only by model capability but also by economic considerations, as GPT-3.5 is significantly cheaper to deploy.

As shown in Figure 4.8, GPT-4 consistently outperforms GPT-3.5 across all prompt variants. The best GPT-3.5 result is obtained with the Categories prompt, achieving an accuracy of approximately 0.50, about five percentage points lower than GPT-4. Larger performance drops are observed for the Human and Summaries prompts, while the Areas prompt suffers the most pronounced degradation, with accuracy decreasing to approximately 0.37. This behavior is likely due to the complexity and length of the

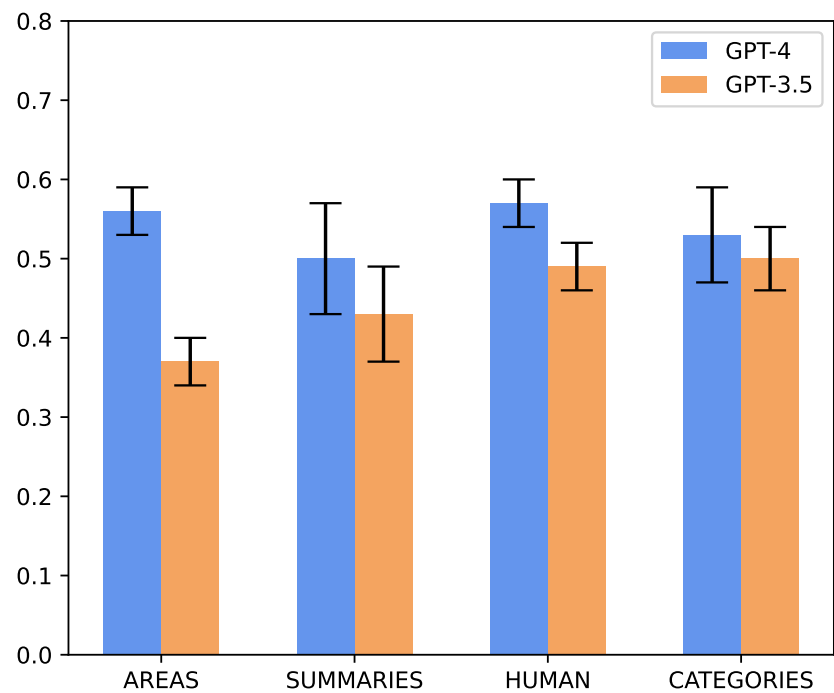


Figure 4.8: Comparison between GPT-4 and GPT-3.5 across different zero-shot prompting strategies. The figure reports classification accuracy for each prompt configuration, together with the variance computed over the five testsets.

numerical information encoded in that prompt.

No substantial differences are observed in terms of variance across models. On average, excluding the weakest prompt configuration, GPT-4 outperforms GPT-3.5 by approximately six accuracy points. Given this gap, the improved performance of GPT-4 may justify its higher cost in deployment scenarios where accuracy is critical.

Chapter 5

Conclusion

In this thesis we described the main research activities carried out during my PhD and their experimental results. The work developed along two complementary directions: the first focused on understanding what Large Language Models encode and how semantic abilities emerge during pretraining, while the second investigated how Transformer-based language models can be integrated into real-world systems, such as the DynDevice LMS developed by Mega Italia Media.

The first contribution of this thesis concerned the analysis of how semantic knowledge is acquired by Large Language Models during pretraining, with a particular focus on figurative language recognition. To study this phenomenon, we adopted the Minimum Description Length probing framework on a suite of models and on multiple checkpoints saved throughout training. Then, we extended this analysis with a layer-wise study to investigate where this knowledge is most strongly encoded within the Transformer stack. Finally, we complemented the investigation with controlled experiments to assess how pretraining data affect the acquisition of figurative knowledge.

The experimental results showed that figurative language is not learned as a single uniform capability: different figures of speech exhibited different acquisition dynamics, with heterogeneous trends across tasks. Across the figures of speech that show measurable learning, model size is generally associated with higher MDL compression, particularly when moving from small to medium scales, suggesting that increased capacity supports richer semantic representations; however, scaling beyond this regime often yields diminishing returns on several tasks. Learning trajectories further suggested that most of the extractable figurative-language knowledge is acquired early during pretraining and then stabilizes, producing a plateau in which additional training yields limited improvements. The layer-wise analysis indicated that figurative knowledge is not maximally recoverable from the final layer but is instead concentrated in intermediate or upper-intermediate layers, often yielding substantial gains when the best-performing layer is considered. This distinction is especially salient for oxymoron recognition: at the final layer, smaller GPT-NeoX models exhibited a late-stage decline in compression, whose beginning is size-dependent, whereas best-layer evaluations mitigate this apparent forgetting, supporting the interpretation that late-stage degradation at the final layer

reflects a redistribution of information rather than a complete loss. Finally, with respect to pretraining data, our controlled comparisons suggested that training on a larger and more diverse corpus is beneficial, while using a literary-focused corpus did not provide measurable improvements compared to a general-purpose corpus for figurative language recognition.

As a second contribution, this thesis presented two industrial systems developed with Mega Italia Media and integrated into DynDevice. The first system addresses the automatic assessment of open-ended questions in workplace safety courses by modeling correction as concept-level detection in learner responses, and it was deployed to support the operational workflow; despite performance degradation under real-world conditions due to unseen questions and concepts, it satisfies the primary requirement of reliably detecting target concepts and, according to domain experts, reduces manual correction time by approximately two thirds and average waiting times from about 12 to 4 hours. The second system addresses automatic ticket assignment in technical support pipelines through an LLM-based approach based on GPT-4 prompting, showing that naive zero-shot prompting is insufficient, while structured prompts and ensemble prompting substantially improve performance, approaching that of a supervised BERT classifier, while remaining more robust to evolving issues without requiring retraining.

As future work, we plan to extend both the analytical and the applied directions of this thesis. On the analysis side, we aim to broaden the study to additional figures of speech, datasets, and languages, including multilingual models, in order to better understand how figurative knowledge emerges and whether it is language-specific or shared across languages. We also intend to move beyond controlled model suites by evaluating larger LLMs that are only accessible as black-box systems, to assess how well the observed trends generalize to state-of-the-art models. On the application side, we plan to strengthen the deployed systems by adopting more robust LLM-based techniques that better handle distribution shifts and new data, and by refining ticket routing and other DynDevice workflows. More broadly, we aim to continue the collaboration with Mega Italia Media by developing new projects that integrate Generative AI capabilities into DynDevice to support teachers and instructional designers in producing course materials more efficiently, while preserving traceability and quality control.

Bibliography

- Y. Adi, E. Kermany, Y. Belinkov, O. Lavi, and Y. Goldberg. 2017. [Analysis of sentence embedding models using prediction tasks in natural language processing](#). *IBM Journal of Research and Development*, 61(4/5):3:1–3:9.
- Ehsan Aghazadeh, Mohsen Fayyaz, and Yadollah Yaghoobzadeh. 2022. [Metaphors in pre-trained language models: Probing and generalization across datasets and languages](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2050, Dublin, Ireland. Association for Computational Linguistics.
- Nicola Arici, Luca Putelli, Ejdis Gjinika, Ivan Serina, and Alfonso Emilio Gerevini. 2025. [Learning trajectories of figurative language for pre-trained language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 14440–14461, Suzhou, China. Association for Computational Linguistics.
- Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. [Layer normalization](#). *CoRR*, abs/1607.06450.
- Yonatan Belinkov. 2022. [Probing classifiers: Promises, shortcomings, and advances](#). *Computational Linguistics*, 48(1):207–219.
- Yonatan Belinkov and James Glass. 2019. [Analysis methods in neural language processing: A survey](#). *Transactions of the Association for Computational Linguistics*, 7:49–72.
- Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O’Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. [Pythia: A suite for analyzing large language models across training and scaling](#). In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. [PIQA: reasoning about physical commonsense in natural language](#). In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on*

Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 7432–7439. AAAI Press.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. [GPT-NeoX-20B: An open-source autoregressive language model](#). In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin. Association for Computational Linguistics.

Marianna M. Bolognesi, Claudia Roberta Combei, Marta La Pietra, and Francesca Masini. 2024. [What makes an awfully good oxymoron?](#) *Language and Cognition*, 16(1):242–262.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020a. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020b. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Christian Burgers, Britta C Brugman, Kiki Y Renardel de Lavalette, and Gerard J Steen. 2016. [Hip: A method for linguistic hyperbole identification in discourse](#). *Metaphor and Symbol*, 31(3):163–178.

Cheng-Han Chiang, Sung-Feng Huang, and Hung-yi Lee. 2020. [Pretrained language model embryology: The birth of ALBERT](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6813–6828, Online. Association for Computational Linguistics.

Leshem Choshen, Guy Hacohen, Daphna Weinshall, and Omri Abend. 2022. [The grammar-learning trajectories of neural language models](#). In *Proceedings of the 60th Annual*

- Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8281–8297, Dublin, Ireland. Association for Computational Linguistics.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2023. [Palm: Scaling language modeling with pathways](#). *J. Mach. Learn. Res.*, 24:240:1–240:113.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [ELECTRA: pre-training text encoders as discriminators rather than generators](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. [Think you have solved question answering? try arc, the AI2 reasoning challenge](#). *CoRR*, abs/1803.05457.
- Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. [What you can cram into a single \\$&!#* vector: Probing sentence embeddings for linguistic properties](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Mohsen Fayyaz, Ehsan Aghazadeh, Ali Modarressi, Hosein Mohebbi, and Mohammad Taher Pilehvar. 2021. [Not all models localize linguistic knowledge in the same place: A layer-wise probing on BERToids’ representations](#). In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 375–388, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021. [The pile: An 800gb dataset of diverse text for language modeling](#). *CoRR*, abs/2101.00027.

- Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, and 24 others. 2024. [OLMo: Accelerating the science of language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand. Association for Computational Linguistics.
- Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. [Distributional vectors encode referential attributes](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 12–21, Lisbon, Portugal. Association for Computational Linguistics.
- John Hewitt and Percy Liang. 2019. [Designing and interpreting probes with control tasks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. [TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.
- Omid Kashefi, Andrew T. Lucas, and Rebecca Hwa. 2018. [Semantic pleonasm detection](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 225–230, New Orleans, Louisiana. Association for Computational Linguistics.
- Arne Köhn. 2015. [What’s in an embedding? analyzing word embeddings through multilingual evaluation](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073, Lisbon, Portugal. Association for Computational Linguistics.
- Christian Lehmann. 2005. [Pleonasm and hypercharacterisation](#). In Geert Booij and Jaap Van Marle, editors, *Yearbook of Morphology 2005*, pages 119–154. Springer Netherlands, Dordrecht.
- Hector J. Levesque. 2011. [The winograd schema challenge](#). In *Logical Formalizations*

- of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011.* AAAI.
- Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. [Logiqa: A challenge dataset for machine reading comprehension with logical reasoning](#). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. [ijcai.org](#).
- Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Tan, Tianhua Tao, Junbo Li, Yuqi Wang, Suqi Sun, Omkar Pangarkar, Richard Fan, Yi Gu, Victor Miller, Yonghao Zhuang, Guowei He, Haonan Li, Fajri Koto, Liping Tang, Nikhil Ranjan, and 9 others. 2023. [LLM360: towards fully transparent open-source llms](#). *CoRR*, abs/2312.06550.
- Zhuang Liu, Wayne Lin, Ya Shi, and Jun Zhao. 2021. [A robustly optimized BERT pre-training approach with post-training](#). In *Chinese Computational Linguistics - 20th China National Conference, CCL 2021, Hohhot, China, August 13-15, 2021, Proceedings*, volume 12869 of *Lecture Notes in Computer Science*, pages 471–484. Springer.
- Alessio Miaschi, Dominique Brunato, Felice Dell’Orletta, and Giulia Venturi. 2020. [Linguistic profiling of a neural language model](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 745–756, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Michael Mohler, Mary Brunson, Bryan Rink, and Marc Tomlinson. 2016. [Introducing the LCC metaphor datasets](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pages 4221–4227, Portorož, Slovenia. European Language Resources Association (ELRA).
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. [The LAMBADA dataset: Word prediction requiring a broad discourse context](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#).

- In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. [Improving language understanding by generative pre-training](#). *OpenAI blog*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. [Language models are unsupervised multitask learners](#). *OpenAI blog*.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. [Wino-grande: An adversarial winograd schema challenge at scale](#). In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press.
- Naomi Saphra and Adam Lopez. 2019. [Understanding learning dynamics of language models with SVCCA](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3257–3267, Minneapolis, Minnesota. Association for Computational Linguistics.
- Nina Schneidermann, Daniel Hershcovich, and Bolette Pedersen. 2023. [Probing for hyperbole in pre-trained language models](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 200–211, Toronto, Canada. Association for Computational Linguistics.
- Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxin Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, and 17 others. 2024. [Dolma: an open corpus of three trillion tokens for language model pretraining research](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788, Bangkok, Thailand. Association for Computational Linguistics.
- Carlo Strapparava. 2018. [Metaphor: A computational perspective by tony veale, ekaterina Shutova and beata beigman klebanov](#). *Computational Linguistics*, 44(1):191–192.
- Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. [Memorization without overfitting: Analyzing the training dynamics of large language](#)

- [models](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 38274–38290. Curran Associates, Inc.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. [Llama: Open and efficient foundation language models](#). *arXiv preprint arXiv:2302.13971*.
- Enrica Troiano, Carlo Strapparava, Gözde Özbal, and Serra Sinem Tekiroğlu. 2018. [A computational exploration of exaggeration](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3296–3304, Brussels, Belgium. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.
- Jesse Vig and Yonatan Belinkov. 2019. [Analyzing the structure of attention in a transformer language model](#). In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.
- Elena Voita and Ivan Titov. 2020. [Information-theoretic probing with minimum description length](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.
- Jonas Wallat, Jaspreet Singh, and Avishek Anand. 2020. [BERTnesia: Investigating the capture and forgetting of knowledge in BERT](#). In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 174–183, Online. Association for Computational Linguistics.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. [GLUE: A multi-task benchmark and analysis platform for natural language understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.
- Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. <https://github.com/kingoflolz/mesh-transformer-jax>.

- Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. [Crowdsourcing multiple choice science questions](#). In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.
- Fan Xu, Ziyun Zhu, and Xiaojun Wan. 2023. [Creative destruction: Can language models interpret oxymorons?](#) In *Natural Language Processing and Chinese Computing - 12th National CCF Conference, NLPCC 2023, Foshan, China, October 12-15, 2023, Proceedings, Part I*, volume 14302 of *Lecture Notes in Computer Science*, pages 645–656. Springer.
- Kelly Zhang and Samuel Bowman. 2018. [Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361, Brussels, Belgium. Association for Computational Linguistics.
- Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. [Aligning books and movies: Towards story-like visual explanations by watching movies and reading books](#). In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society.