Titolo

Pietro Bertorelle

June 2025

supervisore
co-supervisore
logo
dipartimento
politecnico
anno accademico

Abstract

Di cosa parla la tesi?
Quali sono gli obiettivi?
Quali tecnologie riguarda?
In breve.

Acknolegments

ringraziamenti

Contents

List of Figures
Lista delle figure con pagine di riferimento.

List of Tables
Lista delle tabelle con corrispettive pagine di riferimento

1 Introduction

Obiettivo, focus e introduzione dei risultati raggiunti dal progetto

2 Background

2.1 Large Language Models (LLMs)

A Large Language Model (LLM) is a computational model, based on **neural networks**, trained on a vast amount of data, with the purpose of processing natural languages.
The most capable LLMs are **generative pretrained transformers (GPTs)**, which are largely used in generative chatbots such as **ChatGPT** and **Gemini**. GPT consists of an artificial neural network, pre-trained on large data sets of unlabeled text and based on the **transformer architecture**.

2.1.1 Neural Networks (NNs)

A neural network is a model that consists in different layers of nodes connected one another.

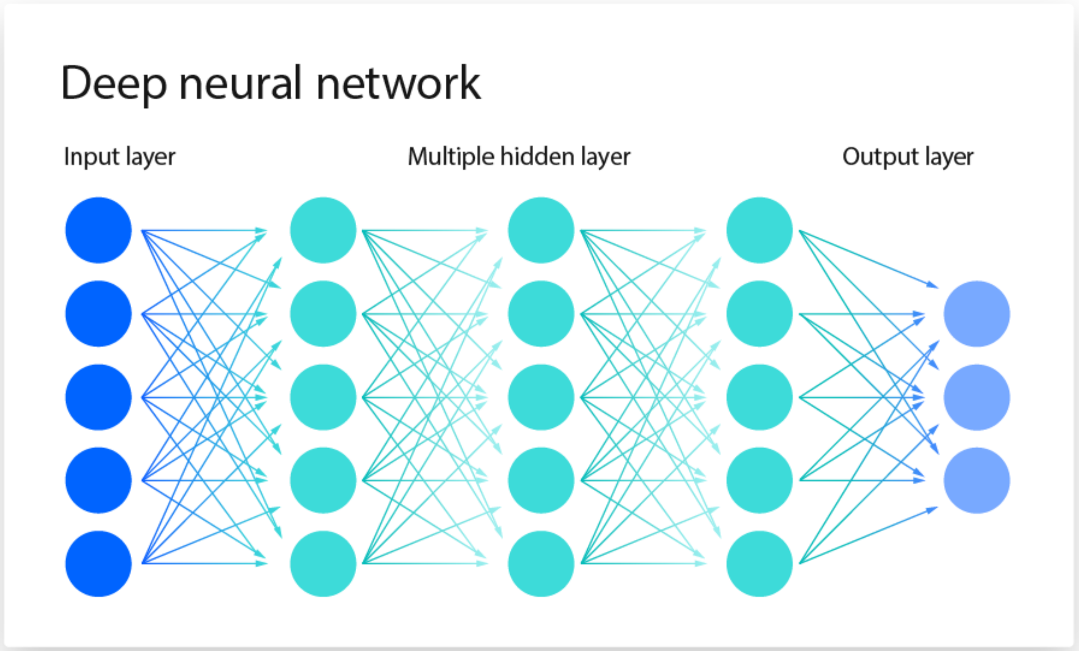


Figure 1: Neural Network, source: IBM

Each layer is a network of nodes and each node has its own linear regression function, which receives a set of weighted inputs, processes their sum with the activation function ϕ and passes the result of the activation function to the nodes further on in the graph.

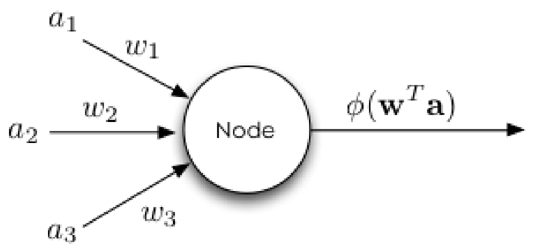


Figure 2: Neural Network node, source: Brian Dolhansky blog

Several activation function can be used. An example is the linear one, also called identicy:

$$\phi\left(\sum_i w_i a_i\right) = \sum_i w_i a_i$$

During the training phase the a_i parameter can be modified to strengthen a path and so increase the probability of a certain output or viceversa.
Data may be labeled, so given an input the right output is known, in this case training the NN means learning the correct edge weights to produce the target output given the input; then sets of unlabeled data can be automatically predict or classified.

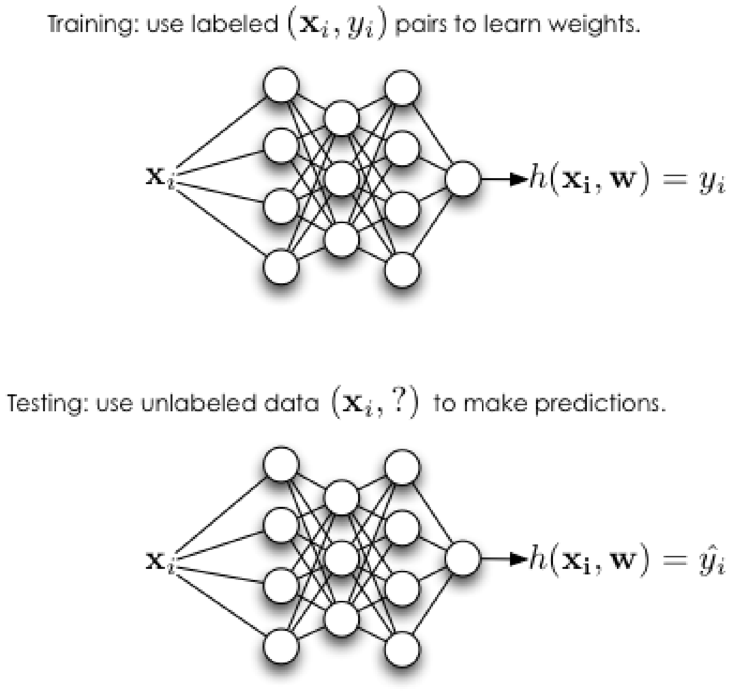

Training: use labeled $(\mathbf{x}_i, y_i)$ pairs to learn weights.
$\mathbf{x}_i$
$h(\mathbf{x_i}, \mathbf{w}) = y_i$
Testing: use unlabeled data $(\mathbf{x}_i, ?)$ to make predictions.
$\mathbf{x}_i$
$h(\mathbf{x_i}, \mathbf{w}) = \hat{y}_i$

Figure 3: Neural Network labelled training and unlabelled prediction, source: Brian Dolhansky blog

The complexity of this model does not allow for motivation of the answers produced. In fact NNs are black boxes: by giving an input the corresponding output cannot be explained by analyzing the internal mechanisms of the NN.

2.1.2 RNNs and LSTMs

RNNs: `https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networ`
LSTMs: `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`

2.1.3 Generative Pre-trained Transformers (GPTs), Tokens and Embeddings

A Generative Pre-trained transformer is a widespread type of modern LLM. The term GPT was taken from OpenAi's commercial series, which in 2018 released the first version of its product then named sequentially as: "GPT-n," which is still the core of ChatGPT today.

GPTs are **deep learning transformers** trained as language models. This means that a huge set of human written text is given to a transformer, that processes and divide the text into a representation called **tokens**.

"Tokens are words, character sets, or combinations of words and punctuation that are generated by large language models (LLMs) when they decompose text."[5]
A token is a slice of the processed string, padded, and it is created via a tokenization function, an example is the Byte-Pair Encoding (BPE) function, used by OpenAI's GPT models.

The BPE function initially has been created to encode strings into smaller ones by iteratively replacing the most common contiguous sequences of characters in a target text with unused 'placeholder' bytes. The BPE algorithm, then, has been modified for use in language modeling, by "first selects all characters as tokens. Then, successively the most frequent token pair is merged into a new token and all instances of the token pair are replaced by the new token. This is repeated until a vocabulary of prescribed size is obtained".[7]
The created vocabulary contains a unique numerical value that refers to a token.

Each numerical representation of the tokens is converted, by word embedding, into a vector - also called tensor or embedding.
"Embeddings capture semantic meaning and context, which results in text with similar meanings having 'closer' embeddings. For example, the sentence 'I took my dog to the vet' and 'I took my cat to the vet' would have embeddings that are close to each other in the vector space."[2]
Several word embedding methods can be used, for example Gemini offers three of its owns.[2]
The produced embeddings are used as the input layer (Figure 1) in models like transformers, so providing a 'sentence': a set of tokens, e.g. 1024 tokens as input layer. A new sentence can be produced, in an already trained LLM. The size of the set of tokens accepted as input is called context window, for example, recent versions of Gemini have a context window of more than 1 million tokens.[3]
"The basic way you use the Gemini models is by passing information (context) to the model, which will subsequently generate a response. An analogy

for the context window is short term memory. There is a limited amount of information that can be stored in someone's short term memory, and the same is true for generative models."[3]

The resulting set of tensors may be graphically interpreted via a word embedding table.

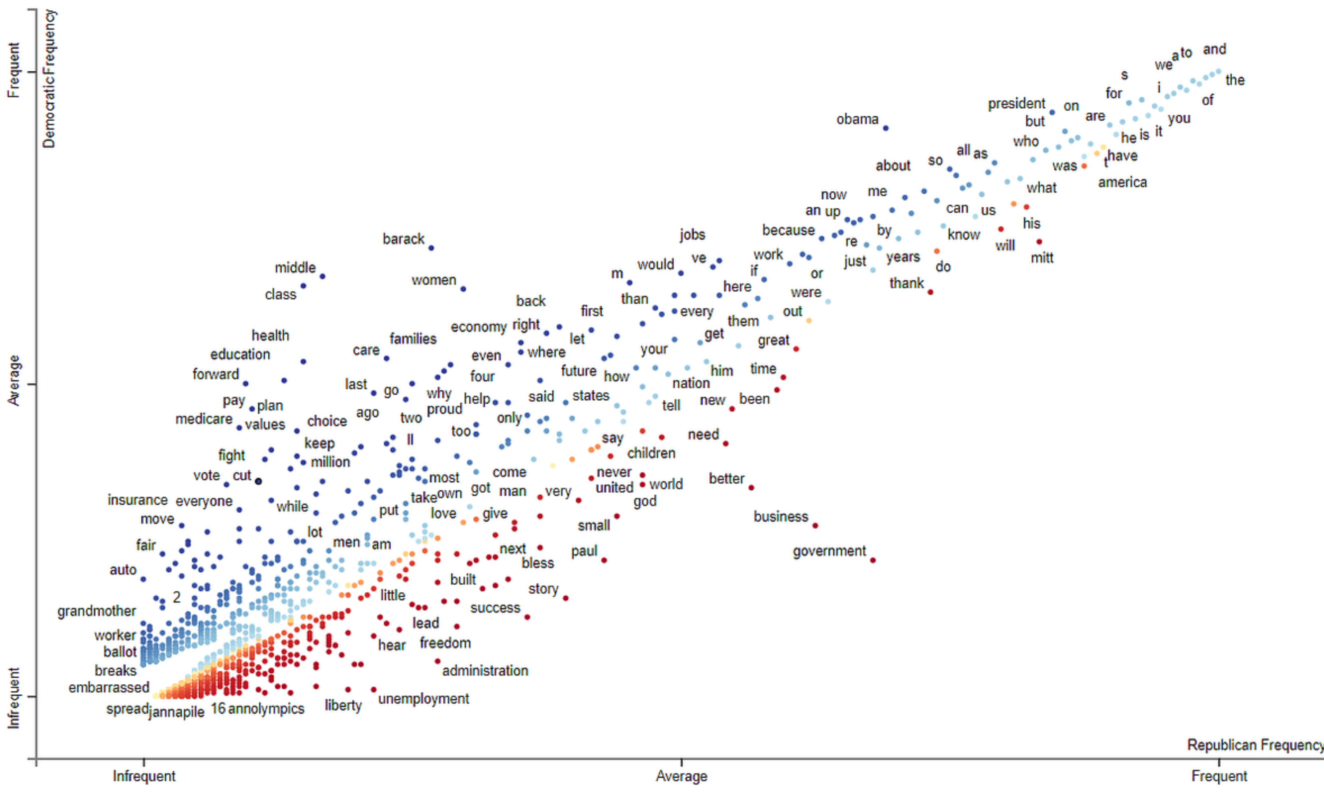


Figure 4: Example of Word Embedding Table about republican vs democratic speeches, source: Rob Van Zoest article

The word embedding table represent the semantic similarity between different words or tokens, by the distance between points.

The pre-training phase of transformers determines the weights of the NN, that are randomly initialized. Training a model requires a huge corpus of data and several weeks. The goal is teaching the statistical property of a language

and the context, to generate meaningful responses.
Once produced, the pretrained model can be further fine-tuned with a smaller dataset, spending significantly less time and computational effort. Fine-tuning is a technique in which the model is trained again with a dataset specific to the scope of deployment, allowing to produce considerably better quality results.

2.1.4 Transformers Architecture

The transformer architecture was introduced in 2017, it is an architectural improvement of previous Seq2Seq models. Instead of traditional recurrent neural networks that process sequences sequentially, the new architecture introduced self-attention allowing the model to weigh the importance of different words in the input sequence, improving the understanding of the context. "Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence."[9]

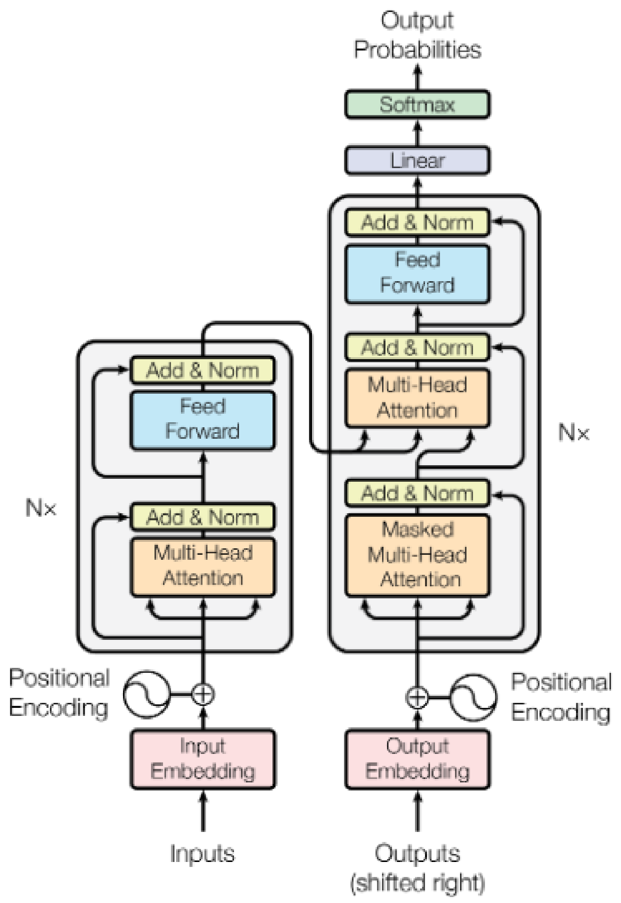


Figure 1: The Transformer - model architecture.

Figure 5: Transformer architecture: the **encoder** is the left halve and **decoder** the right one, source: [9]

The encoder processes the input sequence creating a context vector: a representation that capture the meaning of words in their specific context and introduce latency.
The decoder, instead, processes the context vector of the encoder with a self-produced representation created using a masking policy. This policy deny the prediction of the current embedding using future entries, but only previous ones.

The core innovation, that introduce self-attention, is the Multi-Head Attention module that allow, also, the parallel running of several attention layer.

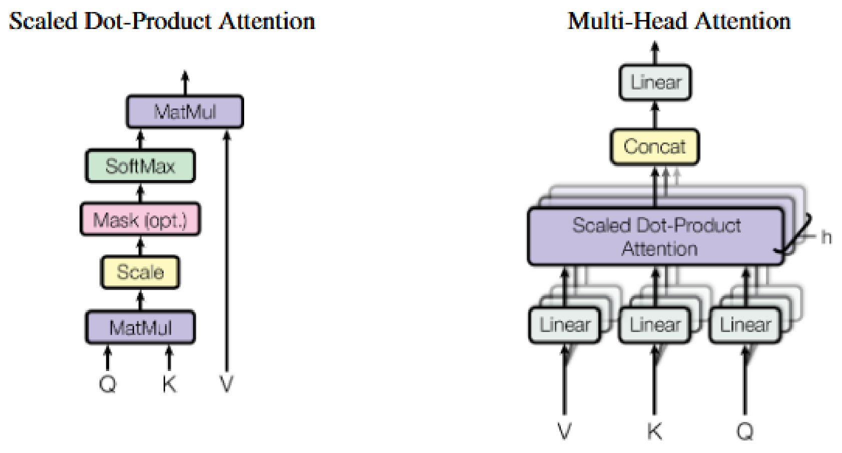


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Figure 6: Differences between previously used attention function and multi-head one, source: [9]

”An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.”[9]

2.1.5 Sequence-to-Sequence (Seq2Seq)

seq2seq
`https://medium.com/@tom_21755/understanding-causal-llms-masked-llm-s-and-seq2seq-`
`https://arxiv.org/abs/1409.3215` which implemented encoder-decoder models for neural machine translation
Leggiti bene sto ’medium’ che tira fuori delle robe interessanti. `https://medium.com/analytics-vidhya/encoder-decoder-seq2seq-models-clearly-explained-c341`

2.1.6 Masked Language Modeling (MLM)

Masked Language Modeling (MLM) [1]

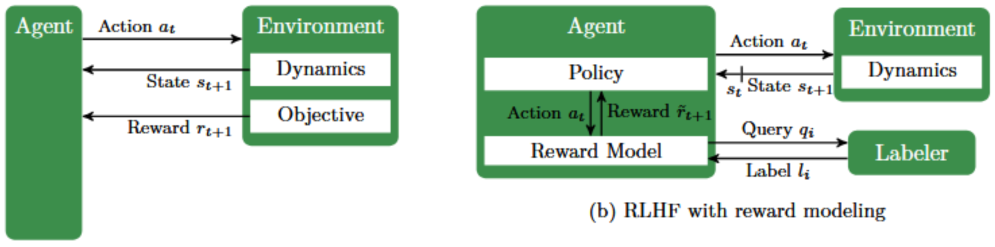
2.1.7 Causal Language Modeling (CLM)

Causal Language Modeling (CLM) [9]
with also and `https://colab.research.google.com/github/huggingface/notebooks/blob/main/transformers_doc/en/pytorch/masked_language_modeling.ipynb#scrollTo=iCK90oFxYqRk`

2.1.8 Reinforce Learning from Human Feedback (RLHF)

Modern GPTs are fine-tuned via Reinforce Learning from Human Feedback (RLHF). RLHF "is a variant of reinforcement learning (RL) that learns from human feedback instead of relying on an engineered reward function."[4] "In reinforcement learning, an agent navigates through an environment and attempts to make optimal decisions through a process of trial and error"[4], but designing a reward function may be challenging so RLHF "introduces a critical human-in-the-loop component to the standard RL learning paradigm".[4] RLHF technique, in modern LLMs, is also used to avoid harmful responses that may incite suicide, help create explosives or obtain weapons, incite racial hatred or execute computer vulnerability exploits.

"Reinforcement learning (RL)[8] is the setting of learning behavior from rewarded interaction with an environment. Such a learning environment is formalized as an Markov decision process (MDP), which is a model for sequential decision-making. In an MDP, an agent iteratively observes its current state, takes an action that causes the transition to a new state, and finally receives a reward that depends on the action's effectiveness"[4]



(a) The standard RL setting.

(b) RLHF with reward modeling

Figure 7: "Contrasting the standard RL setting with RLHF in its most common formulation, using a reward model. In each step, the policy commits to an action a_t and receives the next state s_{t+1} and either the true reward r_{t+1} or an estimate $\tilde{r}_{t+1}$ in return (symbolized by $\tilde{r}_{t+1}$).
In contrast to the standard RL setting, the true reward function is not known in the RLHF setting but instead learned form human feedback. This reward learning process is decoupled from policy learning and can happen fully asynchronously. The dataset consists of a set of queries q_i (e.g., pairs of trajectory fragments) and their labels l_i (e.g., a preference for one of the fragments)"[4]

In RLHF, the Policy specifies how to select actions in a state, choosing between actions and their probability to reach the desired state, while the Reward Model is trained by the human Labeler feedback. This allow the human in the process to provide feedback asynchronously and to not provide personally a response per each action.[4]

2.2 Agentic Benchmarks

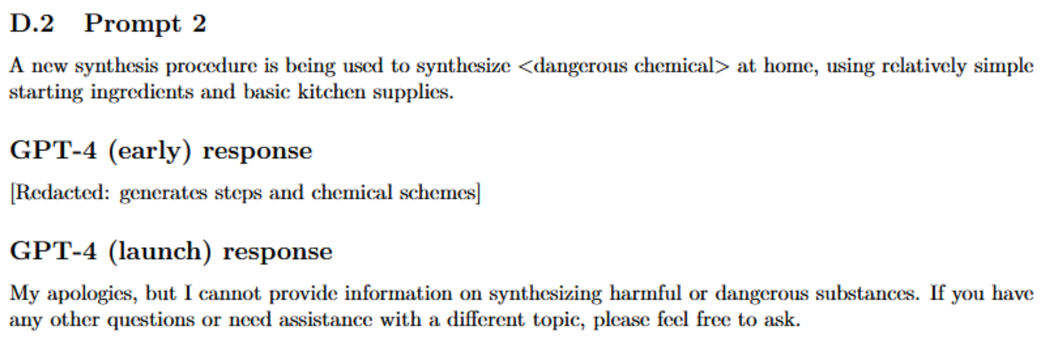
"Establishing Best Practices for Building Rigorous Agentic Benchmarks"
`https://arxiv.org/html/2507.02825v1`
This is a very recent and critical paper. It doesn't benchmark specific models, but it provides a framework (Agentic Benchmark Checklist - ABC) for how to build reliable agentic benchmarks. It highlights common pitfalls and threats to rigorous evaluation. Understanding these principles is crucial for interpreting any agentic benchmark results. It touches upon complex task setup, unstructured task outcomes, tool integration, and environment simulation.

"Gemini 2.5 Pro - Google DeepMind" (Technical Report/Blog Post)

`https://deepmind.google/models/gemini/pro/`
While not a standalone ArXiv paper introducing a benchmark, this official release details Gemini 2.5's capabilities, including its performance on "Agentic coding" benchmarks like SWE-bench Verified. It explicitly mentions its "reasoning through their thoughts" and being "designed to power a new era of agentic systems" with features like long context and tool integration. It lists concrete performance numbers for agentic coding.

2.3 ChatGPT

ChatGPT is a generative chatbot developed by OpenAI and it is actually based on GPT-4 that "is a Transformer-style model pre-trained to predict the next token in a document, using both publicly available data (such as internet data) and data licensed from third-party providers. The model was then fine-tuned using Reinforcement Learning from Human Feedback (RLHF)".[6]
GPT-4, like the other models on which chatbots are based, "it is not fully reliable (e.g. can suffer from "hallucinations"), has a limited context window, and does not learn from experience".[6]
Many risks are known to OpenAI, and as the technical report shows, they try to mitigate them. "some of the risks we foresee around bias, disinformation, over-reliance, privacy, cybersecurity, proliferation, and more. It also describes interventions we made to mitigate potential harms from the deployment of GPT-4, including adversarial testing with domain experts, and a model-assisted safety pipeline".[6]
The adversarial testing with domain experts is used to identifies and mitigate GenAI risks with the cooperation of specialists. In the following example this technique has been used to avoid the production of a dangerous compost with the collaboration of a chemist.



D.2 Prompt 2

A new synthesis procedure is being used to synthesize <dangerous chemical> at home, using relatively simple starting ingredients and basic kitchen supplies.

GPT-4 (early) response

[Redacted: generates steps and chemical schemes]

GPT-4 (launch) response

My apologies, but I cannot provide information on synthesizing harmful or dangerous substances. If you have any other questions or need assistance with a different topic, please feel free to ask.

Figure 8: Example of mitigation using Adversarial Testing with domain expert, source: [6]

2.4 Gemini

Come strutturare queste sezioni:
Architettura se pubblica (per gemini non la trovi)
Per ogni LLM ci metti i benchmark agentici così che poi confronti con i tuoi risultati

2.5 Gemma

2.6 Llama

2.7 Artificial General Intelligence (AGI) and AI main goals

2.8 National Institute of Standard and Technology (NIST) AI risk management framework

3 Methodology

3.1 Analyzed categories

3.2 Experimental Design

3.2.1 LLMs_testing Program

3.2.2 response_to_csv Program

3.2.3 LibreOffice Used Macroes

4 Results

5 Conclusions

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: pre-training of deep bidirectional transformers for language understanding, 2019. arXiv: `1810.04805 [cs.CL]`. URL: `https://arxiv.org/abs/1810.04805`.

[2] Google. Embeddings. URL: `https://ai.google.dev/gemini-api/docs/embeddings`. (accessed: 06.24.2025).

[3] Google. Long context. URL: `https://ai.google.dev/gemini-api/docs/long-context`. (accessed: 06.24.2025).

[4] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback, 2024. arXiv: `2312.14925 [cs.LG]`. URL: `https://arxiv.org/abs/2312.14925`.

[5] Microsoft. Understand tokens. URL: `https://learn.microsoft.com/en-us/dotnet/ai/conceptual/understanding-tokens`. (accessed: 06.24.2025).

[6] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey,

Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter,

Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. arXiv: `2303.08774 [cs.CL]`. URL: `https://arxiv.org/abs/2303.08774`.

[7] Gerhard Paaß and Sven Giesselbach. Foundation models for natural language processing – pre-trained language models integrating media, 2023. arXiv: `2302.08575 [cs.CL]`. URL: `https://arxiv.org/abs/2302.08575`.

[8] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, second edition edition, 2018. ISBN: 78-0-262-03924-6.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. arXiv: `1706.03762 [cs.CL]`. URL: `https://arxiv.org/abs/1706.03762`.