

UNIVERSITÀ DEGLI STUDI DI ROMA TOR VERGATA

MACROAREA/FACOLTÀ DI INGEGNERIA



CORSO DI STUDIO IN
INGEGNERIA GESTIONALE

TESI DI LAUREA IN
INTELLIGENZA ARTIFICIALE 2

TITOLO

*Realisation of a chatbot for Fundamental Analysis generation combining LLMs
with semantically enriched data*

Relatore:

Chiar.mo Prof.
Armando Stellato

Laureando:

matricola: 0326999
Claudio Coltellacci

Anno Accademico 2023/2024

INDEX

INTRODUCTION	4
1 THEORETICAL FOUNDATION	6
1.1 Financial Analysis	6
1.1.1 Fundamental Analysis	6
1.1.2 Strategic Analysis - PEST Analysis	7
1.1.3 Key Financial Indicators Analysis - ROI, ROE, ROA.....	7
1.1.4 Income Statement and Balance Sheet Forecast.....	12
1.1.5 Firm Evaluation.....	13
1.2 Artificial Intelligence	15
1.2.1 Machine Learning	16
1.2.2 Deep Learning-Neural Networks	22
1.2.3 Recurrent Neural Networks	29
1.2.4 LSTM: Long-Short Term Memory Architecture.....	36
1.2.5 The Attention Mechanism and the Transformer	39
1.2.6 Decoder-only models: the birth of Large Language Models	46
1.2.7 How to train Large Language Models	50
1.2.8 Aligning LLMs with the user's will: Chain of Thought and Instruct-GPT	52
1.2.9 LLMs' criticalities.....	55
1.3 Ontologies and Knowledge Graphs	60
1.3.1 GraphRAG applied to Knowledge Graphs	62
1.3.2 SPARQL and Ontology-based Query check	65
EXPERIMENTAL SETUP	67
2 DATA COLLECTION	67
2.1 Scenario.....	67
2.1.1 Yahoo! Finance's Python Package - Data cleaning.....	67

2.1.2	Damodaran’s Data Archives.....	68
2.1.3	Worldometer.....	69
2.1.4	Morningstar and Investor Relations.....	70
2.1.5	Other sources.....	70
2.2	Notes on Data Quality.....	72
3	RESULTS.....	74
3.1	The experiment	74
3.2	The tools.....	74
3.3	The Knowledge Graph.....	76
3.3.1	Classes.....	77
3.3.2	Properties	78
3.4	The AI Assistant’s Architecture.....	85
3.5	Final results.....	87
	CONCLUSION.....	101
	REFERENCES.....	102
	SPECIAL THANKS	110

INTRODUCTION

At the end of 2022, OpenAI released ChatGPT, a chatbot capable of engaging in natural language conversations with its users in a human-like manner. This new app drew worldwide attention to the field of Artificial Intelligence, which rapidly became a mass phenomenon. Firms around the world, fascinated by its potential, implemented it within their business process with great return: according to Google, 74% organisations are currently seeing ROI from their gen AI investments, 86% of organisations using AI expect 6% or more gains to overall annual company revenue, and 84% of organisations observe an accelerated time-to-value, transforming a gen AI use case idea into production within 6 months (Google Cloud, 2024). Furthermore, Artificial Intelligence holds the potential to improve humans' quality of life, facilitating healthcare delivery, offering new and improved services, and helping ordinary people with chores such as planning appointments or trips. Many people are also frightened about the dangers this technology can bring. In a recent interview with the business magazine "Fortune", Klarna's CEO Sebastian Siematkowski expressed his intention to reduce the firm's workforce from 5000 to 2000 employees, stating that operations which previously required significant amounts of time can be now done much faster by fewer people with the help of ChatGPT (Siematkowski, 2024). Moreover, European institutions showed concern towards the uncontrolled adoption of AI into business, which resulted in the AI Act, an attempt to regulate this technology and ensure people's safety against biometric categorisation systems, social scoring, AI-based deceptive techniques deployment and other threats. (European Parliament, 2025). This work's goal is to better understand the applicability of generative AI on a firm's operations by attempting to realise a chatbot assistant which could be integrated into a financial firm's business process. The choice of the Financial Industry is justified by its nature: it is a knowledge-intensive sector, which could benefit from the implementation of gen AI. This is also proven by Google's report on generative AI's implementation in Financial Services: 90% of financial firms that adopted this technology report revenue gains of more than 6%; 57% financial companies implemented gen AI for increasing individual productivity (which is the focus of our work); 50% of these organisations indicate employee productivity has at least doubled thanks to Large Language Models (LLM); and 60% financial services corporations intend to allocate at least half of their AI budget toward gen AI (Google Cloud, 2024). The chatbot will be intended as an assistant for Financial Analysis, capable of retrieving information from a data source and, possibly, manipulate these data to return a report of a firm's financial situation. To achieve this, this work will combine Retrieval-Augmented Generation (RAG), the best way to improve an LLM's performance in knowledge-intensive tasks, as recent literature

confirms, and RDF Ontologies, as their structure can further enhance the reading capabilities of an AI model. The contribution of this work is twofold:

- The realisation of a Financial OWL Ontology and the creation of a dataset extracted from freely accessible online sources modelled according to this vocabulary. This was possible thanks to VocBench, a web-based collaborative development platform for managing OWL ontologies, which enabled the development of a visualisation and management dashboard for financial data organisation.
- The exploitation of such dataset within an AI-powered architecture which, given a Natural Language question, explores the available information and generates a Natural Language report of a firm's financial condition. The AI models powering the architecture are provided by the French company Mistral AI whose API is freely accessible.

Furthermore, the project includes a brief economic analysis of the whole system, in terms of data and inference costs, using market data retrieved from several reports and Mistral's dashboard for API usage. This analysis is intended to assess whether this system can be implemented into a business process operating at a larger scale and with better performing models. Ultimately, this prototype serves as a practical case study of how generative AI along with semantically enriched data can become a valuable decision-support tool in knowledge-intensive industries, contributing to the ongoing discussion of responsible AI integration into complex business environments.

1 THEORETICAL FOUNDATION

1.1 Financial Analysis

According to the Borsa Italiana glossary (Borsa Italiana, 2025), financial analysis can be categorised into three types: ESG (Environmental, Social and Governance) or Extra-Financial Analysis, which evaluates a firm's commitment to environmental, social, and sustainable governance themes through its financial statements; Technical Analysis, which examines current price trends based on historical data, to predict their evolution in the future; and Fundamental Analysis, which will be the focus of the GenAI model in this study.

1.1.1 Fundamental Analysis

Fundamental Analysis identifies and predicts the economic and financial variables influencing stock value behaviour. More specifically, on the one hand, it examines macroeconomic indicators related to the monetary system firms operate in, such as the GDP of a country and the inflation rate; on the other hand, it gathers information on the financial solidity and profitability of companies, in relation to their market value, considering their future growth prospects within their respective industries. The Fundamental Analysis process is structured as follows.

- Strategic Analysis of the Target Firm, including a General Economic Analysis and an Evaluation of the Reference Sector
- Key Financial Indicator Analysis (ROE, ROI, ROA) and Comparison to Historical Values and Competitors Within the Same Sector to Determine How the Company Creates Value, which will be used in the Forecasting Phase
- Income Statement and Balance Sheet Forecast, whose result will be utilised to estimate the firm value, using various techniques like Free Cash Flow (FCF), Dividends or Anomalous Operating Result
- Firm evaluation based on forecasted data obtained in the previous phase, employing different evaluation methods, including the Discounted Cash Flow method (DCF), Economic Value Added (EVA), Dividend Discount Method (DDM), and Gordon Growth Model. DCF and EVA estimate the firm's Enterprise Value, while DDM and the Gordon Growth Method return the company's Equity Value.

Fundamental Analysis differs from Technical Analysis, as its objective is to identify the best investment opportunities, whereas the latter focusses on determining the best timing for investments. The large language model deployed for this work will follow this exact methodology for financial analysis. In the following paragraphs, each step of the Fundamental Analysis will be explored, explaining which analysis tools and indicators will be used by the Generative Model.

1.1.2 Strategic Analysis - PEST Analysis

Strategic analysis aims to evaluate the business conditions under which the firm operates, differentiating between the internal and external environments. The internal environment analysis explores the strengths and weaknesses of a firm to understand its behaviour and potential on the market (the most popular technique for internal analysis is SWOT Analysis); the External Environment's Analysis typically identifies forces and variables that influence any other firm in the same sector, such as economic trends, current legislation, prevailing political ideologies, the socio-cultural environment, and technological advancements. PEST Analysis is one of the most widely used techniques for monitoring the external environment to understand and predict its evolution. The name of the analysis stems from the initials of each macroenvironment variable it examines, which are:

- Political variables, which refer to how governments can influence firm activity through market regulations, taxation, environmental policies, and investments in R&D, among others.
- Economic variables, such as GDP and Inflation.
- Social variables, which reflect how socio-cultural values can influence the firm's market behaviour.
- Technological variables, assessing the level of technological advancement in the country where the firm operates (A. La Bella, 2016)

Regarding internal environment analysis, since no information on the companies' internal policies is available, the LLM will not be required to retrieve such data. Therefore, the Generative AI Model will utilise the Knowledge Graph to conduct a PEST analysis, among other tasks.

1.1.3 Key Financial Indicators Analysis - ROI, ROE, ROA

Financial Indicators provide an overview of how the firm exploits its resources. More specifically, they assess a company's ability to generate income through their Equity and Debt. The most basic indicator, which also serves as a starting point for more advanced financial metrics, is EBIT (Earnings

Before Interest and Taxes). As its name suggests, EBIT represents a firm's net operating income, calculated as the difference between revenues and the sum of operating costs, labour costs, amortization and depreciation. This indicator is closely related to EBITDA (Earnings Before Interest, Taxes, Depreciation and Amortization) as the latter also includes the aforementioned amortisation and depreciation. Its normalised version, Normalised or Adjusted EBITDA, incorporates discretionary adjustments to remove the effect of non-recurring items and irregular events (Wall Street Prep, 2025). While these values provide a simplistic representation of the results of a company's core business performance, they serve as the basis for calculating more comprehensive financial indicators. Net Operating Income is influenced by several factors, including Business Activity conditions, which encompass the firm's structure and size, determining what to produce and how; Elasticity, defined as the balance between organisational rigidity and management flexibility, in response to market fluctuations; and Efficiency, which can be categorised as Internal - also referred as Productivity, measuring resource utilisation - External - also referred as Competitiveness, reflecting how the firm interacts with markets in acquiring -raw materials or selling its product - and Financial - the firm's ability to negotiate in Financial Markets in order to secure resources for sustaining the business (G. Ferrero, 2006). The financial indicator which quantitatively summarises the ratio between the operating income for the period and the corresponding average invested capital over the same timeframe is ROI (Return on Investment), calculated as

$$ROI = \frac{R_o}{K}$$

where R_o represents Net Operating Income and K denotes Average Invested Capital (G. Ferrero, 2006). Invested Capital, also referred to as Operating Assets includes Cash, Accounts Receivable, Inventory, Machinery, Plants and other productive asset (Garrison R.H., 2012). Therefore, it is a representation of a firm's efficiency in utilising its resources, signalling its capability to generate profits from its Operating Assets. The greater this value is, the more effective the firm is at generating income with fewer resources. Thanks to the concept of Financial Leverage, it is also possible to determine a threshold to assess whether an ROI value is positive or not. Financial Leverage generally measures the extent to which a firm relies on debt financing, and its formula is:

$$ROE = \left[ROI + (ROI - i) * \frac{D}{E} \right] * (1 - t)$$

where $(1 - t)$ is the portion of income remaining after taxes, D is the amount of debt, E is the firm's Equity, i is the interest rate for debt, and ROE is the Return on Equity ratio. ROE is given by the ratio between Net Profit and Total Net Worth:

$$ROE = \frac{R_n}{E}$$

This ratio evaluates how effectively a firm generates profit from its own equity (R.N. Anthony, 2010). If multiplied and divided by the difference between Net Operating Income and Financial Costs, assuming no Financial Income, it becomes:

$$ROE = \frac{R_n}{R_o - FC} * \frac{R_o - FC}{E}$$

Further, by multiplying and dividing by the amount of Average Invested Capital, the result is:

$$ROE = \frac{K * (R_o - FC)}{E * K} * \frac{R_n}{R_o - FC} = \frac{K * R_o - FC * K}{E * K} * \frac{R_n}{R_o - FC}$$

Since the Average Invested Capital is the sum of Net Total Worth and the company's debt, it's possible to rewrite the equation as:

$$\begin{aligned} ROE &= \frac{R_o * E + R_o * D - K * FC}{K * E} * \frac{R_n}{R_o - FC} = \left(\frac{R_o}{K} + \frac{D * R_o}{E * K} - \frac{FC}{E} \right) * \frac{R_n}{R_o - FC} = \\ &= \left[\frac{R_o}{K} + \frac{D}{E} * \left(\frac{R_o}{K} - \frac{FC}{D} \right) \right] * \frac{R_n}{R_o - FC} = \left[ROI + (ROI - i) * \frac{D}{E} \right] * (1 - t) \end{aligned}$$

This demonstrates the ROE ratio's equivalence with the earlier described Financial Leverage equation. Therefore, to determine whether an ROI value is positive, it suffices to verify if it exceeds the ROE and Interest Rate value. If the ROI is higher than the ROE, it means that it is also higher than the interest rate, indicating that the firm is able to rely on efficiently financing debt. Moreover, thanks to the Capital Asset Pricing Model (CAPM), it is possible to determine a threshold for ROE as well. The CAPM states that if the market portfolio M is efficient, the expected return \bar{r}_i of any asset i satisfies:

$$\bar{r}_i - r_f = \beta_i(\bar{r}_M - r_f)$$

where \bar{r}_M is the expected value of the market rate of return, r_f is a risk-free asset's return (such as a bond) and β_i is determined by:

$$\beta_i = \frac{\sigma_{iM}}{\sigma_M^2}$$

where σ_{iM} is the covariance of the asset with the market portfolio, and σ_M^2 is the variance of the market's rate of return. This follows from the One-fund theory, based on Markowitz's modelling for minimum-variance portfolios. Assuming a risky asset can be described by its expected rate of return and variance, an investment portfolio can be defined as the linear combination of n assets, according to n weights whose sum must be equal to 1, with an associated rate of return and variance. The 2^n possible assets' combinations define the Feasible Set of portfolios. Since investors seek to minimise risk (variance), efficient portfolios must lie on the Minimum-Variance Set, the Feasible Set's portion with the lowest variance values. However, investors also aim to maximise return and, given that expected rates of return decrease with variance, the efficient set of portfolios, or Efficient Frontier, is the subset of the Minimum-Variance Set where portfolios achieve higher rates of return than the Minimum-Variance Portfolio. The Efficient Frontier is obtained by solving the following optimisation problem:

$$\min \frac{1}{2} \sum_{i,j=1}^n w_i w_j \sigma_{ij}$$

subject to:

$$\sum_{i=1}^n w_i \bar{r}_i = \bar{r}$$

$$\sum_{i=1}^n w_i = 1$$

where \bar{r} is the investor's desired rate of return. The Two-fund theorem states that two efficient funds (portfolios) can be established so that any efficient portfolio can be duplicated, in terms of mean and variance, as a combination of these two. Therefore, all investors seeking efficient portfolios need only invest in combinations of these two funds. Furthermore, investors can reduce variance by including risk-free assets. The One-fund theorem extends this by stating that there exists a single fund F of risky assets such that any efficient portfolio can be constructed as a combination of the fund F and a risk-free asset. The CAPM states that fund F is the combination of every possible risky asset, that is the market portfolio, assuming market equilibrium - every investor is a mean-variance optimiser-, investors' rationality – everyone wants to optimise its return - and information symmetry – everyone agrees on the probabilistic structure of assets (mean values and variances), leading to the standard CAPM equation for expected return. Because of this, since it would be rather burdensome for an individual to assemble the market portfolio, mutual funds, termed Index funds, have been designed to closely match the market portfolio (Ex.: S&P500). However, the model's assumptions are unrealistic as real-world markets exhibit Information Asymmetry and Investors' Irrationality. This suggests that active management can outperform the market. Since ROE is the actual rate of return of the asset “i”, obtained from accounting data, and \bar{r}_i is the expected rate of return of the same asset ROE should be:

$$ROE \geq r_f + \beta_i (\bar{r}_M - r_f)$$

where the difference between the first and second terms of the equation is equal to Jensen's index,

which measures the asset's performance compared to the market (Luenberger, 2009). For a comprehensive assessment, in order to measure the asset's efficiency, it's necessary to analyse its performance relative to the market. This is the purpose of Treynor's Index, which measures the excess return over a risk-free asset achieved by a fund or a market portfolio per unit of systematic risk. Treynor's Index is given by:

$$T = \frac{r - r_f}{\beta_i}$$

where r represents the return of the asset or fund under evaluation (Borsa Italiana, 2025). Investors can compare Treynor's Index values between an Index fund and the asset in question to evaluate the asset's efficiency. Another important return metric is ROA, given by the following ratio:

$$ROA = \frac{R_o + R_a + R_f}{K}$$

where R_o represents Net Operating Income, R_a represents Atypical Income, R_f corresponds to Financial Income and K denotes the Average Invested Capital (G. Ferrero, 2006). ROA serves as an alternative to ROI when the latter fails to accurately reflect a firm's ability to generate income, particularly when a significant portion of income stems from atypical sources - that is, non-operating activities. In this case, Income is equal to the EBIT. Comparing ROI and ROA helps investors assess how the firm generates income, offering insights into its reliance on market-driven versus non-operating activities. The LLM will be tested on the computation of these metrics with data provided by the Knowledge Graph. The rate of return of the market will be given by Blackrock's BGF World Technology Fund Average rate over the last 10 years (since all firms belong to the Technology sector), while the risk-free asset will be represented by the US Treasury Bond. β_i per Industries will be retrieved from Damodaran's Data Archives in their unlevered and corrected per cash form (which means it is computed net of tax rate and debt-equity ratio).

1.1.4 Income Statement and Balance Sheet Forecast

Income Statement and Balance Sheet Forecast involve creating a series of new financial documents based on revenue and cost variation estimates for up to the next 5 years from the date the original

Income Statement and Balance Sheet were prepared. According to Organismo Italiano di Valutazione's Discussion Paper n. 1/2021, the financial analyst should verify:

- Why and how the information was produced.
- Whether the prospective information was prepared according to the accounting principles adopted by the evaluated firm.
- Whether the prospective information is consistent with other projections utilised for other analyses.
- Whether the prospective information is up to date.
- Whether any sources support the reasonableness of the estimates.
- The list of sources used in preparing the estimates.

The Knowledge Graph will provide a series of forecasts on revenue and cost increases (or decreases) for each firm, which will be utilised by the LLM to prepare the Income Statement and Balance Sheet Forecast.

1.1.5 Firm Evaluation

The projections for the following years ultimately serve as a basis for the firm evaluation. Given the existence of several methods for firm evaluation, numerous debates over the appropriate methodology to value investment properties have evolved across different jurisdictions and sectors over a long period. The outcome of such debates is that many varying practices have surged in different countries, with explicit Discounted Cash Flow (DCF) prevailing in some while being used for different bases in others. (Royal Institution of Chartered Surveyors (RICS), 2023). The DCF method determines the asset's Present Value by discounting forecasted cash flows back to the valuation date. In some circumstances, for long-lived or perpetual-lived assets, DCF may include a Terminal Value, representing the asset's worth at the end of the explicit projection period. The key steps in the DCF method are:

- Selecting the most appropriate type of cash flow for the nature of the subject asset and the assignment,
- Defining the most appropriate explicit period, if any, over which the cash flow will be forecast,
- Preparing cash flow forecasts for that period,

- Assessing whether a Terminal Value is appropriate for the nature of the considered asset at the end of the explicit forecast period and, if so, determining the appropriate Terminal Value for the nature of the asset,
- Identifying the appropriate discount rate,
- Applying the discount rate to the forecasted future cash flow, including the Terminal Value, if applicable. (International Valuation Standards Council, 2016)

According to Borsa Italiana's Glossary (2025), there are two primary approaches to a DCF-based firm evaluation. The first one only considers the Equity Value, discounting cash flows available to shareholders by the cost of their equity to obtain each share's value. The second evaluates the firm as a whole, discounting Free Cash Flows (representing operating cash flows available to all investors) by the Weighted Average Cost of Capital (WACC). The difference between the resulting firm's value and the amount of Debt represents the Equity's value which, when divided by the number of shares, determines the per-share value. Free Cash Flows, for each year, are calculated as follows:

$$FCF = EBIT + Amortizations/Provisions - Inv - \Delta COC - T$$

Where "Inv" represents the firm's investments during the forecast period (e.g, the firm planned the acquisition of new machinery after three years), T denotes the amount of taxes to be paid and ΔCOC is the variation of Circulating Operating Capital - which is the difference between the firm's Credits and Debts - through the years. The Terminal Value is calculated using Free Cash Flows, following the formula:

$$TV = \frac{FCF_n * (1 + g)}{WACC - g}$$

Where g represents the esteemed growth rate, and FCF_n is the Free Cash Flow of the final projection year. As its name suggests, the WACC is a weighted average of the firm's costs of equity and debt (C_e and C_d), expressed as:

$$WACC = C_e * \frac{E}{E + D} + C_d * (1 - T) * \frac{D}{D + E}$$

Where E represents Equity, D denotes Debt, and T is the corporate tax rate. The cost of Equity is determined using the CAPM, which equates it to the asset's expected return:

$$C_e = r_f + \beta_i(\bar{r}_M - r_f)$$

While the Cost of Debt corresponds to the interest rate paid on the firm's debt. Ultimately, the firm's value is calculated as:

$$PV = \sum_i^n \frac{FCF_i}{(1 + WACC)^i} + \frac{TV}{(1 + WACC)^{n+1}} - NFP$$

Where NFP (Net Financial Position) is the net of the firm's available liquidity and long-term debt. The LLM ought to use this methodology to derive a firm's value, exploiting data contained in the Knowledge Graph.

1.2 Artificial Intelligence

Artificial Intelligence is the discipline that studies whether it is possible to replicate the Human cognitive process via an unanimated object. The need for such a subject derives directly from the invention of the computer at the hands of the British mathematician Charles Babbage in 1833. Alan Turing provided a test in 1950 to assess whether a machine can be mistaken for a human being. The test can be described in terms of a game – also known as “Imitation Game” – played with 3 people: a man (A), a woman (B), and an interrogator (C) who may be of either sex. The interrogator is in a room apart from the other players and can only communicate with them in a typewritten form. His goal is to determine which of the two players is a man and which is a woman. A's goal in the game is to try and cause C to make the wrong identification, while B's goal is to help C recognise the two. Turing's question was whether C would decide wrongly as often when A's role is taken by a machine as he does when the game is played between a man and a woman. As a result of this formulation, it is possible to avoid asking whether machines can think (which suffers from the ambiguities stemming from the concept of “thinking”) and answer a closely related question in relatively unambiguous words. This game and the article in which Turing formulated it, “Computing Machinery and Intelligence” (1950), make the foundations for modern Artificial Intelligence studies. Today, thanks to Large Language Models, Artificial Intelligence is closer than ever at being a valid player for the Imitation Game Turing proposed 75 years ago. In this part, the technologies and disciplines which enabled this achievement will be described, with a final review of the State of the Art of Artificial Intelligence.

1.2.1 Machine Learning

Machine Learning is a branch of Artificial Intelligence which focuses on enabling computers and machines to imitate the way that humans learn, to perform tasks autonomously, and to improve their performance and accuracy through experience and exposure to more data (International Business Machine Corporation, 2025). However, the concept of “learning” needs a definition. Tom M. Mitchell, in 1997, provided a formal definition of machine learning algorithms, that addresses this need: “A computer program is said to learn from experience E with respect to some class of Tasks T and performance measure P , if its performance measure at tasks in T , as measured by P , improves with experience E .” At this point, it is necessary to define what experience, tasks, and performance measures are. Experience refers to a set of examples (or data points) that the learner will study. This set, or the Training Set, can contain labelled or unlabelled data. In the first case, the algorithm learns the relationship between data points and labels (or targets) to predict which label can be associated with each feature of another unlabelled set, known as the Test Set. This approach is named Supervised Learning. In the second case, the algorithm experiences the dataset to learn its structure, stemming data clusters (in the case of Cluster Analysis) or a probability distribution. The notion of experience is closely related to tasks, as the former’s definition is completed by the latter. In machine learning, Tasks (T) are generally described in terms of how the system should process data points, typically represented as vectors $\mathbf{x} \in \mathbb{R}^n$ where each entry x_i is a data point feature (for example: given a Real Estate Training Set, each data point is a different house whose features are its size, the number of floors, the presence of a garden etc.). The most common tasks being solved via machine learning are:

- **Classification:** the computer program is asked to specify which of k categories some input belongs to. Therefore, the algorithm is supposed to produce a function $f: \mathbb{R}^n \rightarrow 1, \dots, k$. When $y = f(\mathbf{x})$, the model associates input vector \mathbf{x} with a category identified by the number y . There are variants to this task, such as Probabilistic Classification, where f outputs a probability distribution over classes, and Classification with missing inputs where not every measurement for the input vector is provided. Examples of Classification tasks are Object Recognition and Medical Diagnosis.
- **Regression:** the computer program is asked to predict a numerical value given some input. Therefore, the learning algorithm is supposed to output a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$. The Regression function can be Linear:

$$y(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^n w_i x_i$$

Or be extended to a linear combination of base functions defined on \mathbb{R}^d

$$y(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^n w_i \phi_i(\mathbf{x})$$

Where base functions can either be simple polynomial functions or even more complex functions, such as the Gaussian or the Sigmoid. Examples of Regression tasks are the prediction of Future Prices of Securities and Algorithmic Trading.

- Transcription: the computer program is asked to observe a relatively unstructured representation of data and transcribe it into textual form. One example of this type of task is Speech Recognition.
- Machine Translation: the computer program is asked to convert a sequence of symbols in some language into a sequence of symbols in another language. This is generally applied to natural languages, such as translating from English to Italian.
- Structured output: the computer program is asked to convert the input in an output that is a vector. One example is parsing – mapping a natural language sentence into a tree that describes its grammatical structure and tagging nodes of the trees as being verbs, nouns, or adverbs.
- Anomaly detection: the computer program is asked to sift through a set of events or objects and flags some of them as being unusual or atypical. An example of this type of task is Fault Detection in Predictive Maintenance: by taking as an input a machine state's data, the algorithm should be able to identify any anomalous parameter (such as a higher-than-the-norm temperature or a higher number of residuals inside the lubricating oil) and predict whether and when the machine is going to fail.
- Synthesis and sampling: the computer program is asked to generate new examples that are similar to those in the training data. This is becoming more and more useful in the context of data scarcity.
- Imputation of missing values: the computer program, given a new example $\mathbf{x} \in \mathbb{R}^n$ with missing x_i entries, is asked to predict the missing values.

- Denoising: given a corrupted example $\tilde{x} \in \mathbb{R}^n$ obtained by an unknown corruption process from a clean example $x \in \mathbb{R}^n$, the computer program is asked to predict the clean example from its corrupted version.
- Density estimation or probability mass function estimation: the computer program is asked to learn a function $p_{model} : \mathbb{R}^n \rightarrow \mathbb{R}$ which can be seen as a probability density function if the input $x \in \mathbb{R}^n$ is continuous, or a probability mass function if x is discrete. (I. Goodfellow, 2016)

One important sub-branch of Machine Learning is Language Modelling, whose goal is to design a Machine Learning model which is able to predict upcoming words. More specifically, a Language Model assigns a probability to each possible next word or gives a probability distribution over possible next words (D. Jurafsky, 2025). Following Mitchell's definition, it is necessary to use a Performance metric to evaluate if an algorithm is learning from Experience to resolve Tasks. Generally, the Performance measure is specific to the task T carried out by the algorithm. Tasks such as Classification require measures of accuracy, that is, the number of correct classifications over the total number of classifications.

$$Accuracy = \frac{\#Correct\ Classifications}{\#Total\ Classifications}$$

Equivalent information can be obtained by measuring the error rate, the number of incorrect classifications over the total number of classifications.

$$Error\ Rate = \frac{\#Incorrect\ Classifications}{\#Total\ Classifications}$$

Regression tasks typically use metrics such as the Mean Squared Error:

$$MSE = \frac{1}{2} \sum_{i=1}^n (y(x_i, w) - t_i)^2$$

Or the Mean Absolute Error:

$$MAE = \frac{1}{2} \sum_{i=1}^n |y(x_i, w) - t_i|$$

So, these metrics are used to evaluate the Machine Learning algorithm performance on the Training Set used to derive the prediction function first, and then on the Test Set, which the algorithm has not seen before. Therefore, performance measures become objective functions to optimise (in the case of the Mean Error or the Error Rate, a cost function to be minimised). The most popular algorithm for Machine Learning optimisation is Stochastic Gradient Descent, an extension of the Gradient Descent algorithm. Gradient Descent, also known as the Method of Steepest Descent, exploits the properties of directional derivatives to move the function in the direction which optimises the objective function value. The directional derivative of a function f in direction \mathbf{u} (unit vector) is the slope of the function f in the direction \mathbf{u} . This can be seen as the derivative of the function $f(\mathbf{x} + \alpha\mathbf{u})$ with respect to α , evaluated at $\alpha = 0$. Using the chain rule, for $\alpha = 0$:

$$\frac{\partial}{\partial \alpha} f(\mathbf{x} + \alpha\mathbf{u}) = \mathbf{u}^T \nabla_{\mathbf{x}} f(\mathbf{x})$$

Assuming f needs to be minimised, the direction in which f decreases the fastest can be computed thanks to the directional derivative:

$$\min_{\mathbf{u}, \mathbf{u}^T \mathbf{u} = 1} \mathbf{u}^T \nabla_{\mathbf{x}} f(\mathbf{x}) = \min_{\mathbf{u}, \mathbf{u}^T \mathbf{u} = 1} \|\mathbf{u}\| \|\nabla_{\mathbf{x}} f(\mathbf{x})\| \cos \theta$$

Where θ is the angle between \mathbf{u} and the gradient. This expression is minimised when \mathbf{u} points in the opposite direction as the gradient (which means $\cos \theta = -1$). Therefore, the objective function can be decreased by moving in the direction of the negative gradient. The Gradient Descent algorithm, starting from an arbitrarily decided $\mathbf{x}^{(0)} \in \mathbb{R}^n$, for each iteration k :

- Computes the negative gradient $-\nabla_{\mathbf{x}} f(\mathbf{x}^{(k)})$
- Computes a Learning Rate ϵ
- Computes $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x}^{(k)})$

The algorithm stops when $\|\nabla_{\mathbf{x}} f(\mathbf{x}^{(k)})\| = 0$, which means it has found a stationary point (that is a minimum, a maximum or a saddle point). This allows to find $\mathbf{x} = \operatorname{argmin} f(\mathbf{x})$. The Learning Rate ϵ is a positive scalar that can be chosen in several different ways. One popular strategy for choosing ϵ is the Backtracking-Armijo Line Search, which consists of evaluating $f(\mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x}))$ for several times, until it satisfies Armijo's condition, that is:

$$f(\mathbf{x}^{(k)} + \epsilon^{(k)} \mathbf{p}^{(k)}) \leq f(\mathbf{x}^{(k)}) + \epsilon^{(k)} \beta * [\mathbf{g}^k]^T \mathbf{p}^k$$

For some fixed $\beta \in (0,1)$ and \mathbf{p} descent direction, which, in this case, is the Negative Gradient (Hauser, 2007). The algorithm can be readapted to search for maximum points. Gradient Descent's main weakness lies in its stop condition: when $\|\nabla_{\mathbf{x}} f(\mathbf{x}^{(k)})\| = 0$ the gradient provides no information about which direction to move, but the stationary point the algorithm has found, not only can it be a saddle point, which is neither a maximum or a minimum, but it can also be a local optimum. Ideally, one would prefer to arrive at the global optimum, but this might not be possible. Furthermore, there are local optimums that perform nearly as well as the global optimum and others which perform poorly and should be avoided. Moreover, a recurring problem in Machine Learning is that large Training Sets, required for performing accurately on previously unobserved inputs, are computationally expensive, making optimisation a hard task. This is where Stochastic Gradient Descent comes into play: by assuming that the gradient is an expectation, the algorithm estimates it using a small set of examples. More specifically, on each of the algorithm's steps, a minibatch of examples $\mathbb{B} = \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m')}$ is uniformly sampled from the Training Set. \mathbb{B} 's size m' is typically chosen to be relatively small compared to the total number of examples provided. A Training Set with billions of examples could be fit using updates computed on only a hundred examples. This significantly reduces computational costs and allows models to be trained on large datasets that would otherwise be unfeasible. Thus, SGD's computational cost is $O(m'd/\epsilon)$ (where ϵ is the target accuracy and d the number of features), compared to GD's $O(nd * \log(1/\epsilon))$, which is much larger for large datasets (R. Zadeh, 2015). While evaluating the performance of the computer program on the Training Set provides insight into how well the model has adapted to the data it was trained on, the true measure of an algorithm's effectiveness lies in its ability to predict the target value from unseen data correctly. Thus, performance on the Test Set is a more reliable indicator of the model's real-world applicability. The ability to perform well on previously unobserved inputs is known as generalisation. A well-generalised model performs accurately not only on training data but also on real-world data, which is the goal of Machine Learning. However, evaluating performance on a single Test Set can be misleading, and in some cases, a separate Test Set may not be available for validation. To solve this problem, a Validation Set is needed. Typically, this dataset is constructed using Training data by splitting the Training Set into two disjoint subsets (generally, one uses 80% of the Training Set for Training and the remaining 20% for Validation). Another solution is K-Fold Cross Validation, which consists of splitting the Training Set into k subsets and iteratively using each subset as a Validation

Set. In each iteration, a different subset is designated as the Validation Set, while the remaining subsets serve as the Training Set. The test error may then be estimated as the average error over k trials. This prevents the use of overly small test sets, reducing statistical uncertainty in error estimation and improving the reliability of algorithm comparisons. Achieving optimal generalisation is hindered by two phenomena: Overfitting and Underfitting. Overfitting occurs when the gap between the training error and the test error is too large. This implies that the algorithm generates a function that is exceptionally good at predicting the Training Set target but achieves poor results when exposed to real-world data, as it fails to generalise beyond data it has already memorised. This occurs in excessively complex models, such as high-degree polynomial base functions and can be mitigated by increasing the amount of training data or by using regularisation techniques, such as Lasso and Ridge, which, by assuming the error can be expressed as the linear combination of the error dependent on the dataset (and the parameters) and the error dependent from the parameters alone, introduce a regularisation term, thereby changing the objective function as:

$$E_D(w) + \lambda E_w(w) = \frac{1}{2} \sum_{i=1}^n (t_i - w^T \phi(x_i))^2 + \frac{\lambda}{2} \sum_{j=1}^m |w_j|^q$$

Where the second term of the equation was written considering Mean Squared Error as objective function and q can be either 1 or 2, depending on the adopted regularisation technique (1 for Lasso, 2 for Ridge). Another regularisation technique is Early Stopping: when training large models with sufficient representational capacity to overfit the task, the error on the Training Set will decrease over time, whereas the error on the Validation Set will increase after a certain number of training iterations over the dataset, also known as epochs. Early Stopping consists of returning to the parameter setting at the point in time with the lowest Validation Set error. Underfitting occurs when the model is not complex enough and, thus, cannot obtain a sufficiently low error rate on the Training Set. These two phenomena are related to the bias-variance trade-off, which describes the fundamental sources of error in Machine Learning. Bias occurs when the learner is biased towards a term that is not related to the concept to be learned, therefore making the model not expressive enough to describe the concept. Variance occurs when the model cannot generalise well on new data, implying inconsistent performance on different datasets. Thus, Bias implies Underfitting, whereas Variance causes Overfitting. To avoid incurring one of these two events, a good Machine Learning model should have Low Bias and Variance. However, since Model Complexity is necessary to solve the Machine

Learning task effectively, there must be a trade-off between Bias and Variance, as High Variance implies Low Bias and vice versa, as represented by Figure 2.1:

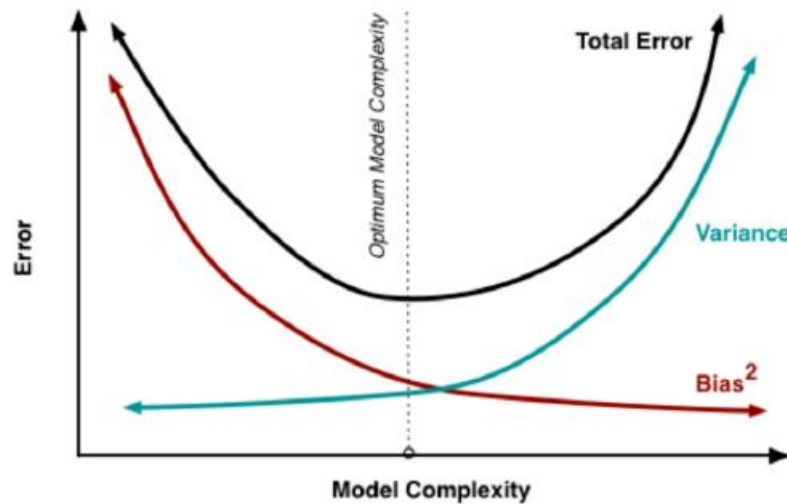


Figure 1.1: Representation of the Bias-Variance Tradeoff. As the plot suggests, the Total Error Curve given by the combination of Bias and Variance Curves has a minimum which is the optimal tradeoff between Bias and Variance.

1.2.2 Deep Learning-Neural Networks

Deep Learning is the branch of Machine Learning which tries to solve those tasks that are easy for people to perform but hard to describe formally, such as speech recognition, image recognition or generating content based on a given context. Traditional Machine Learning algorithms perform well on a wide variety of tasks but fail to solve these apparently intuitive problems. Deep Learning tries to allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, where each concept is defined by a relation between simpler concepts. This also allows human operators not to specify all the knowledge that the computer needs, since it can learn from experience. With this, the computer will be able to learn complicated concepts by building them out of base concepts. The reason this approach has been named Deep Learning resides within its graphical representation: by drawing a graph which shows how concepts are built on top of each other, the graph is deep, with many layers. This helps to introduce one of the challenges of Deep Learning: the curse of dimensionality. This consists of the exponential increase in the number of possible distinct configurations of the set of variables as the number of variables increases, therefore imposing high computational costs. The nature of the issue is statistical: the number of possible configurations of \mathbf{x} is much larger than the number of training examples. When generalising to a new data point, the target

value can be returned by inspecting training examples (for example, by averaging the other target values, in the case of regression, or by returning the most common class of examples, in the case of classification). But in high-dimensional spaces, this might not be possible, since the great number of configurations makes it likely to incur calculating a target value for which there is no example. Therefore, it is impossible to say anything meaningful about these configurations. Deep Learning also attempts to solve this statistical problem, by using advanced algorithms. The quintessential Deep Learning models are the Deep Feedforward Networks, also known as Multilayer Perceptron (MLP) or Neural Networks (NN). The feedforward algorithm's goal is to find the best approximation of a function f^* (which is the same goal as Machine Learning's). Thus, it defines a mapping $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ and learns the values of the parameters $\boldsymbol{\theta}$ that result in the best function approximation. It is known as feedforward, since there are no feedback connections in which the output of the model is fed back into itself. This type of model stems from the Perceptron Model presented in 1943 by W.S. McCulloch and W. Pitts "A Logical Calculus of the Ideas Immanent in Nervous Activity". More specifically, F. Rosenblatt, in "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", describes the rules of organisation of a typical photo-perceptron as follows:

- Stimuli impinge on a retina of sensory units (S-points) which are assumed to respond on all-or-nothing basis, in some models, or with a pulse amplitude or frequency proportional to the stimulus intensity, in other models.
- Impulses are transmitted to a set of association cells (A-units) in a "projection area". The cells in the projection area each receive several connections from the sensory points. The set of S-points transmitting impulses to a particular A-unit will be called the origin points of that A-point. These origin points may be either excitatory or inhibitory in their effect on the A-unit. If the algebraic sum of excitatory and inhibitory impulse intensities is equal to or greater than the threshold ϑ of the A-unit, then the A-unit fires again. The origin points of the A-units in the projection area tend to be clustered or focalised, about some central point, corresponding to each A-point.
- Between the projection area and the association area, connections are assumed to be random.
- The responses are cells (or sets of cells) which respond in the same fashion as the A-units.

Thus, the model can be mathematically represented as follows:

$$h(\mathbf{x}) = g\left(\sum_n \theta_n x_n + b\right)$$

where x_i is the single sensory unit, θ_i is the single association cell, b is the algorithm bias and g is the projection area, also known as the activation function. This model can be represented as a graph, such as in Figure 2.2.

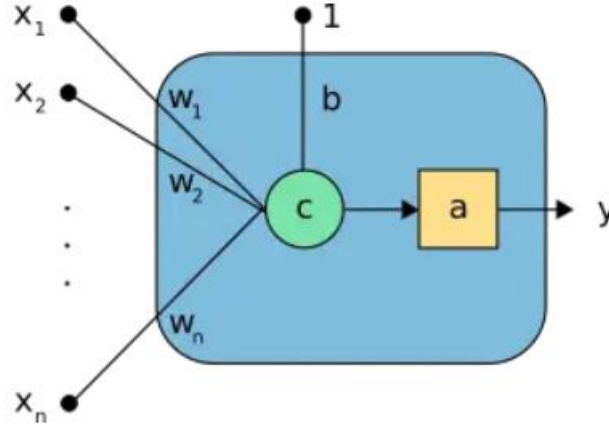


Figure 1.2: Representation of a Perceptron.

A Multilayer Perceptron consists of multiple layers of perceptrons through which the input passes. Therefore, this model type is typically composed of many different functions connected in a chain. For example, given 3 functions $f^{(1)}, f^{(2)}, f^{(3)}$, the model connects the functions as follows:

$$h(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x}; \boldsymbol{\theta}^{(1)}, b^{(1)}); \boldsymbol{\theta}^{(2)}, b^{(2)}); \boldsymbol{\theta}^{(3)}, b^{(3)})$$

In this case, $f^{(1)}$ is the first layer of the Neural Network, $f^{(2)}$ is the second layer, and $f^{(3)}$ is the third layer. In the general case, a standard Neural Network comprises an Input Layer consisting of units (or Input Neurons), an Output Layer, which is the last layer of the Network, and Hidden Layers, which are all the other layers of the MLP. The name of this part of the NN is due to the fact that the learning algorithm must decide how to use these layers to best implement an approximation of f^* , but the training data do not say what each layer should do. The training examples only specify what the Output Layer at each point \mathbf{x} , that is a value close to $y \approx f^*(\mathbf{x})$. Eventually, $h(\mathbf{x})$ will be driven to match $f^*(\mathbf{x})$. In its most generic representation, and given a sequence of functions $g^{(1)}, g^{(2)}, \dots, g^{(k)}$, a neural network is defined as:

$$\begin{aligned}
h(\mathbf{x}) &= g^{(k)}(g^{(k-1)}(\dots g^{(1)}(\mathbf{x}; \theta^{(1)}, b^{(1)}); \dots \theta^{(k-1)} b^{(k-1)}); \theta^{(k)}, b^{(k)}) \\
&= g^{(k)}(\theta^{(k)} g^{(k-1)}(\theta^{(k-1)} \dots g^{(1)}(\theta^{(1)} x + b^{(1)}) \dots + b^{(k-1)}) + b^{(k)})
\end{aligned}$$

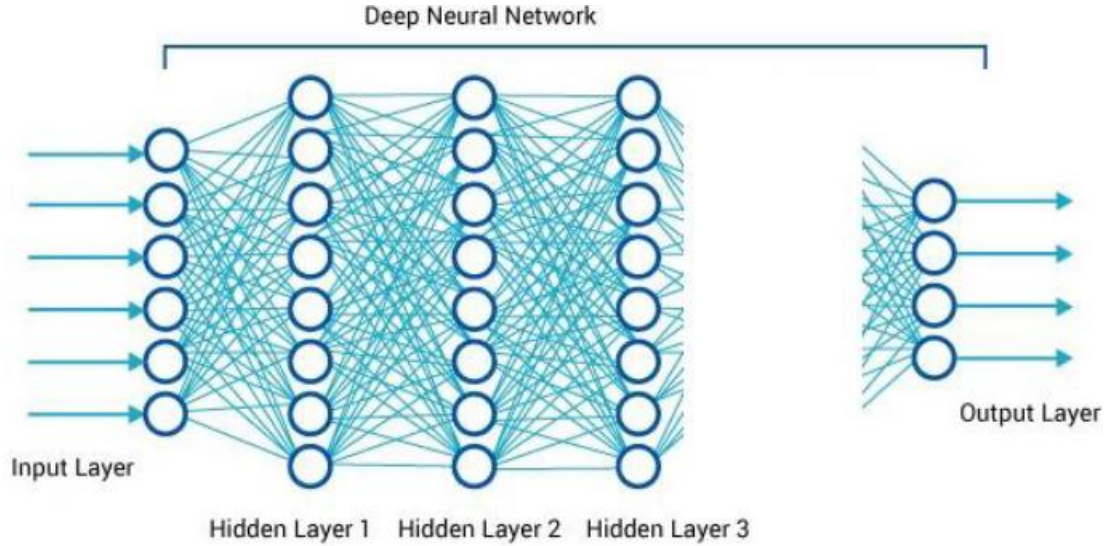


Figure 1.3: Representation of a Feedforward Neural Network.

Feedforward networks have introduced the concept of hidden layers, which requires choosing an activation function, used to compute the hidden layer values. In modern Neural Networks, the default activation function is ReLU (Rectified Linear Unit, we will explore why later), defined by:

$$g(z) = \max(0, z)$$

There are other options as well:

- Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$
- Leaky ReLU: $g(z) = \max(0.1, z)$
- Parametric ReLU: $g(z) = \begin{cases} z & \text{if } z > 0 \\ az & \text{if } z \leq 0 \end{cases}$
- Tanh: $g(z) = \frac{e^{-z} - e^{-z}}{e^z + e^{-z}}$

Neural Networks can either be used with linear models – efficient and reliable, but limited to linear functions, so the model cannot understand interactions between couples of variables – or by applying the same linear model onto a transformed input $\phi(\mathbf{x})$. This way, the model can represent nonlinear

functions as well. Thus, ϕ must be chosen. In Deep Learning, the typical strategy to choose the mapping is to learn ϕ . This means that, given a model $y = f(\mathbf{x}; \boldsymbol{\theta}, \mathbf{w}) = \phi(\mathbf{x}; \boldsymbol{\theta})^T \mathbf{w}$, \mathbf{w} is used to map $\phi(\mathbf{x})$ to the desired output, while $\phi(\mathbf{x})$ is learnt from a broad class of functions using the parameters $\boldsymbol{\theta}$. The optimisation algorithm is used to find the $\boldsymbol{\theta}$ that corresponds to a good representation, after parametrising $\phi(\mathbf{x}; \boldsymbol{\theta})$. The main advantage of this approach is that the human designer does not need to find the exact function, but, rather, the right general function family. The optimisation algorithm is the same Stochastic Gradient Descent algorithm used for traditional Machine Learning, with the main difference being the nonlinearity of a Neural Network. This causes the optimisation function to become nonconvex and, thus, harder to minimise or maximise. Deep Learning generally tackles this problem with the Backpropagation algorithm. When using a feedforward Neural Network, the input \mathbf{x} provides the initial information which, then, propagates up to the hidden units and eventually produces $\hat{\mathbf{y}}$. This is called Forward Propagation. During training, forward propagation can continue onward until it produces a scalar cost $J(\boldsymbol{\theta})$. The Backpropagation algorithm exploits the information gained thanks to the scalar cost by making it flow backwards in order to compute the gradient. Therefore, given that the cost for a single training example is $J(\boldsymbol{\theta}, b; \mathbf{x}, y)$ and that for the whole training set, the cost is the mean of the errors on the training examples:

$$J(\boldsymbol{\theta}, b) = \frac{1}{m} \sum_{i=1}^m J(\boldsymbol{\theta}, b; \mathbf{x}^{(i)}, \mathbf{y}^{(i)})$$

the algorithm can be described through the following steps:

- Step 1: for each example, compute $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\boldsymbol{\theta}, b; \mathbf{x}^{(i)}, \mathbf{y}^{(i)})$
- Step 2: compute a Gradient Descent Forward Pass for an example $\mathbf{x}^{(i)}, \mathbf{y}^{(i)}$

$$\theta_{ij}^{(l)} = \theta_{ij}^{(l)} - \alpha \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\boldsymbol{\theta}, b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\boldsymbol{\theta}, b)$$

- Step 3: for each node i in the first layer, compute an error term δ_i^1 .

Assuming the scalar cost is: $J(\boldsymbol{\theta}, b; \mathbf{x}, y) = \frac{1}{2} |h_{\boldsymbol{\theta}, b}(\mathbf{x}) - y|^2$

$$\delta_i^1 = \frac{\partial}{\partial z_i^1} \frac{1}{2} |y - h_{\boldsymbol{\theta}, b}(\mathbf{x})|^2 = -(y_i - h_i^1) * g'(z_i^1)$$

The error of an output node is the difference between the true output value and the predicted one.

- Step 4: for $l = 2, \dots, n_1 - 1$, for each node i in layer l

$$\delta_i^l = \left(\sum_{j=1}^{s_{l+1}} \theta_{ji}^l \delta_j^{(l+1)} \right) g'(z_i^{(l)})$$

For the intermediate layer l , a node receives a portion of the error stemming from the layer $l + 1$.

- Step 5: given the error terms, the partial derivatives will be computed as:

$$\begin{aligned} \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta, b; x, y) &= h_j^{(l)} \delta_i^{(l+1)} \\ \frac{\partial}{\partial b_i^{(l)}} J(\theta, b; x, y) &= \delta_i^{(l+1)} \end{aligned}$$

Thus, given that SGD's computational cost is $O(m'd/\epsilon)$ and the Backpropagation algorithm's computational cost depends on the number of edges of the Neural Network and that the Neural Network is a directed acyclic graph, which has at most $n - 1$ edges (n being the number of nodes/neurons of the network), the computational cost of the whole optimisation process is $O(n)$. The success of Neural Networks resides within their ability to generalise better than a standard Machine Learning algorithm, as they are the culmination of centuries of progress on the general function approximation task. Today, gradient-based learning in feedforward networks is used as a tool to develop probabilistic models, such as the autoencoder, and many other machine learning tasks. Furthermore, it can be regularised in several different ways to achieve faster and more accurate outputs. Apart from the aforementioned Ridge and Lasso regularisation techniques or Early Stopping, Neural Networks can rely on other methods such as Dropout. This methodology is based on the Bagging technique for combining several different models, which consists of training these models separately and then have them vote on the output for test examples. The final output will be an average of the outputs of each model. Dropout “generates” several Neural Networks by dropping out nonoutput units from an underlying base network and trains the ensemble consisting of all those subnetworks.

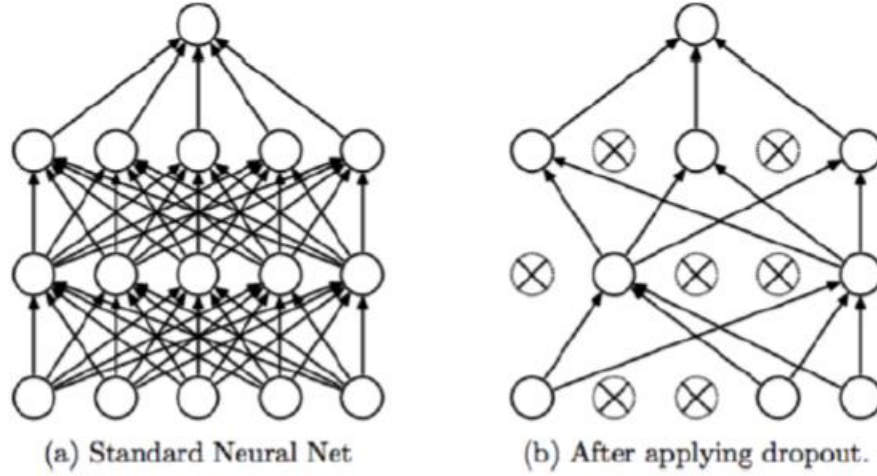


Figure 1.4: Neural Network before and after Dropout

The reason Dropout and Bagging work is that different models will usually not make the same errors on the Test Set. Given a set of k regression models, assuming each model makes an error ϵ_i on each example, with the errors drawn from a zero-mean multivariate normal distribution with variances $\mathbb{E}[\epsilon_i^2] = v$ and covariances $\mathbb{E}[\epsilon_i \epsilon_j] = c$, the error made by the average prediction of all the ensemble models is $\frac{1}{k} \sum_i \epsilon_i$. The expected squared error of the ensemble predictor is:

$$\mathbb{E} \left[\left(\frac{1}{k} \sum_i \epsilon_i \right)^2 \right] = \frac{1}{k^2} \mathbb{E} \left[\sum_i \left(\epsilon_i^2 + \sum_{j \neq i} \epsilon_i \epsilon_j \right) \right] = \frac{1}{k} v + \frac{k-1}{k} c$$

In the case where the errors are perfectly correlated and $c = v$, the mean squared error reduces to v , so the model averaging does not help at all. In the case where the errors are perfectly uncorrelated and $c = 0$, the expected squared error of the ensemble is inversely proportional to the ensemble size. This means that, in the worst case, the model will perform as well as any of its members, while, in the best case, it will perform much better than them. To this, Dropout adds two other significant advantages: it is computationally cheap, and it does not significantly limit the type of model or training procedure that can be used. Using Dropout requires only $O(n)$ computation per example per update in training, to generate n random binary numbers and multiply them by the state and it works well with nearly any model that uses a distributed representation and can be trained with Stochastic Gradient Descent.

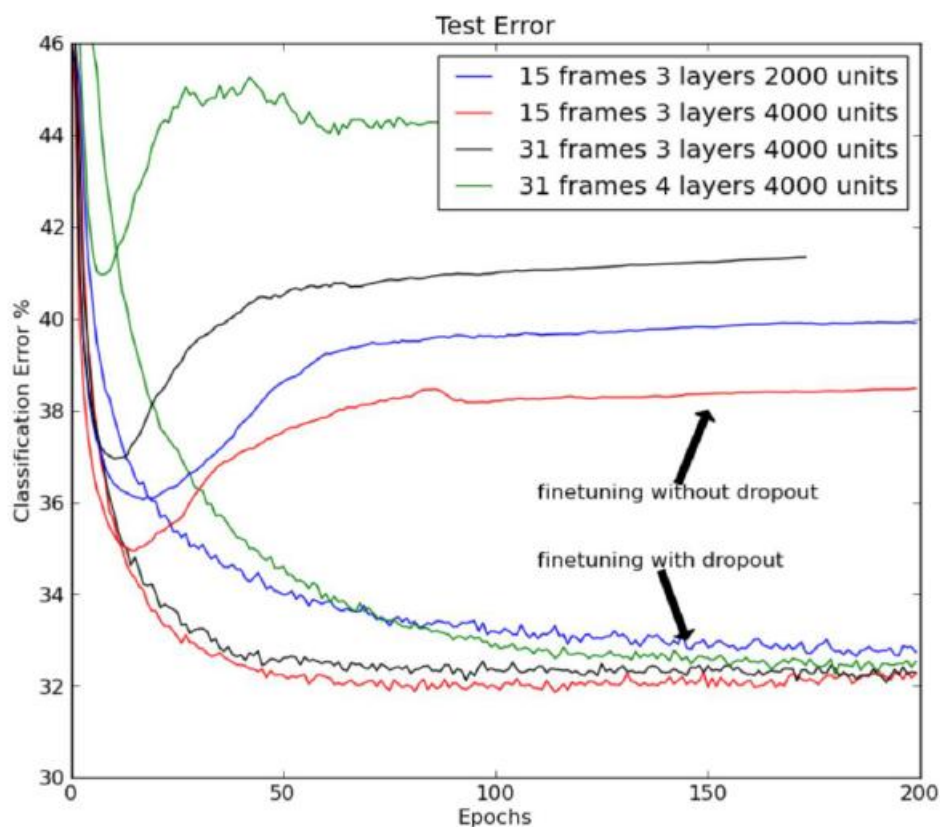


Figure 1.5: The frame classification error rate on the core test set of the TIMIT benchmark. Comparison of standard and dropout finetuning for different network architectures. Dropout adds noise to the error rate but greatly improves the model's performance.

1.2.3 Recurrent Neural Networks

Feedforward Neural Networks excel in generalising functions and, therefore, capturing the relation between different variables or classifying data units. However, when it comes to sequence-based tasks, such as language modelling or machine translation, they struggle. This is because they have no memory of what they have been doing. One Machine Learning's key idea introduced in 1980 is Parameter Sharing, that is sharing parameter across different parts of the model. This idea makes it possible to extend and apply the model to examples of different forms and lengths and generalise across them. Having separate parameters for each value of a time index makes it impossible to sequence lengths not seen during training, nor share statistical strength across different sequence lengths and across different positions in time. For example, given the two sentences "I paid a visit to some friends of mine in London in November" and "I went to London in November to pay a visit to some friends of mine", if the model is asked to extract where the narrator was in November, it would be better for it to recognise that the relevant piece of information is the city of London, regardless of where it appears. If a Feedforward Network were trained for processing sentences, it would have to

learn all the rules of the language used at each position in the sentence. This is because the traditional Neural Network has separate parameters for each input feature. Recurrent Neural Networks (RNN), being based on the idea of Parameter sharing, can process sequences of variable length, sharing the same weights across several time steps. In a basic RNN, the output at each time step depends on both the current input and the hidden state from the previous time step (Stryker, 2024). Thus, it can be represented as a dynamic system:

$$\mathbf{s}^{(t)} = f(\mathbf{s}^{(t-1)}; \boldsymbol{\theta})$$

$$\mathbf{s}^{(t)} = f(\mathbf{s}^{(t-1)}, \mathbf{x}^{(t)}; \boldsymbol{\theta})$$

In this case, \mathbf{s} stands for the state of the system and t is the time step index.

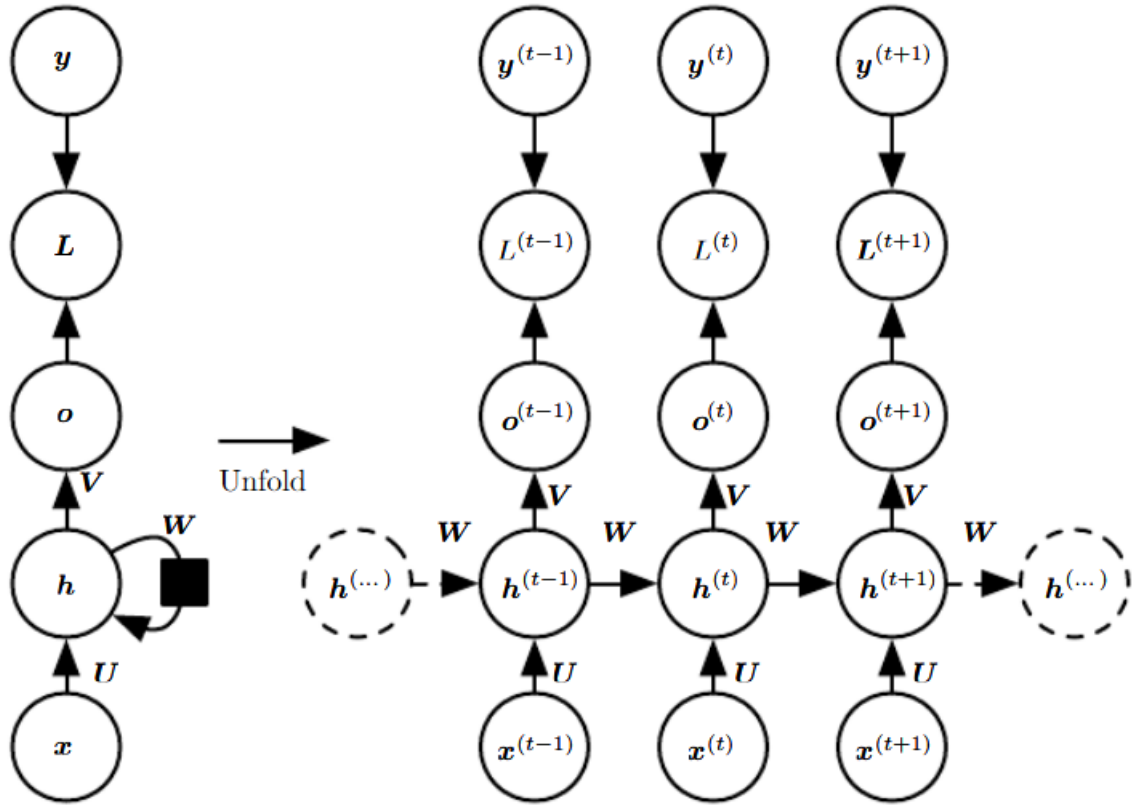


Figure 1.6: The computational graph to compute the training loss of a recurrent network that maps an input sequence of \mathbf{x} values to a corresponding sequence of output \mathbf{o} values. The distance between each \mathbf{o} and the training target \mathbf{y} is measured by a loss L . When using softmax outputs, \mathbf{o} is assumed as an unnormalized log probability. Thus, the Loss function internally computes $\hat{\mathbf{y}} = \text{softmax}(\mathbf{o})$ and compares it to the target \mathbf{y} .

Typically, Recurrent Neural Networks are built by following a design pattern which makes them produce at each time step an output and have recurrent connections between hidden units, as illustrated in Figure 2.5. This is because any function computable by a Turing machine - mathematical tool that can infallibly recognise undecidable propositions, i.e. those mathematical statements that, within a given formal axiom system, cannot be shown to be either true or false, (Encyclopaedia Britannica, 2025) – can be computed by such a Recurrent Network of a finite size. As in every other Neural Network, in RNNs the output is generated via Forward Propagation, with the main difference being that it also depends on the state of the system. Therefore, Forward Propagation begins with a specification of the initial state $\mathbf{h}^{(0)}$ and, for each time step in $(1, \tau)$ interval of time, the following update equations are applied:

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)}$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)})$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}$$

$$\hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{o}^{(t)})$$

Where \mathbf{x} is the input, \mathbf{o} is the output represented as unnormalized log probabilities of each possible value, $\hat{\mathbf{y}}$ is a vector of normalised probabilities over the output (explaining softmax), \mathbf{b} and \mathbf{c} are the bias vectors, \mathbf{U} , \mathbf{V} , \mathbf{W} are the weight matrices for, respectively, for input-to-hidden, hidden-to-output and hidden-to-hidden connections and assuming the activation function \mathbf{h} is the hyperbolic tangent. The total loss for a given sequence of \mathbf{x} values paired with a sequence of \mathbf{y} values, representing training targets, would then be the sum of losses over all the time steps. One evolution of RNNs are Bidirectional RNNs, an architecture combining an RNN that moves forward through time with another RNN moving backward through time, giving the Network the ability to compute a representation that depends on both the past and the future, thus, improving its ability to read a sequence. Assuming the Loss Function is the negative log-likelihood of the target value given the inputs at each time step, then:

$$L(\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\tau)}\}, \{\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(\tau)}\}) = \sum_{t=1}^{\tau} L^{(t)} = - \sum_{t=1}^{\tau} \log p_{\text{model}}(\mathbf{y}^{(t)} | \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)}\})$$

This means that computing the gradient of the Loss Function with respect to the parameters is an expensive operation. The gradient computation involves performing a forward propagation through the time steps followed by a backward propagation through the same time steps. The runtime of the process is $O(\tau)$ and cannot be reduced by parallelization because each time step may be computed only after the previous one. The back-propagation algorithm applied to the unrolled Recurrent Net Graph with $O(\tau)$ cost is called Back-Propagation Through Time (BPTT). Given the RNN's update equations, for each node of the network the gradient $\nabla_N L$ recursively, based on the gradient computed at nodes that follow it in the graph. The recursion starts with the nodes immediately preceding the final loss:

$$\frac{\partial L}{\partial L^{(t)}} = 1$$

If the outputs $o^{(t)}$ are used as the argument to the softmax function and that the loss is the negative log-likelihood of the true target $y^{(t)}$ given the input, the gradient is as follows:

$$(\nabla_{o^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i=y^{(t)}}$$

At the final time step τ , $\mathbf{h}^{(\tau)}$ only has $\mathbf{o}^{(\tau)}$ as a descendent, so its gradient is:

$$\nabla_{\mathbf{h}^{(\tau)}} L = \mathbf{V}^T \nabla_{\mathbf{o}^{(\tau)}} L$$

For the previous steps, therefore at time t in the interval $(1, \tau - 1)$, noting that $\mathbf{h}^{(t)}$ has as descendents both $\mathbf{o}^{(t)}$ and $\mathbf{h}^{(t+1)}$, the gradient is given by:

$$\begin{aligned} \nabla_{\mathbf{h}^{(t)}} L &= \left(\frac{\partial \mathbf{h}^{(t+1)}}{\partial \mathbf{h}^{(t)}} \right)^T (\nabla_{\mathbf{h}^{(t+1)}} L) + \left(\frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}} \right)^T (\nabla_{\mathbf{o}^{(t)}} L) \\ &= \mathbf{W}^T \text{diag} \left(1 - (\mathbf{h}^{(t+1)})^2 \right) (\nabla_{\mathbf{h}^{(t+1)}} L) + \mathbf{V}^T (\nabla_{\mathbf{o}^{(t)}} L) \end{aligned}$$

Where $\text{diag} \left(1 - (\mathbf{h}^{(t+1)})^2 \right)$ indicates the diagonal matrix containing the elements $1 - (h_i^{(t+1)})^2$.

This is the Jacobian of the hyperbolic tangent associated with the hidden unit i at time $t + 1$. Once the gradients on the internal nodes are obtained, the gradients on the parameter nodes can be obtained, which are:

$$\nabla_{\mathbf{c}} L = \sum_{t=1}^{\tau} \left(\frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{c}} \right)^T \nabla_{\mathbf{o}^{(t)}} L = \sum_{t=1}^{\tau} \nabla_{\mathbf{o}^{(t)}} L$$

$$\begin{aligned}
\nabla_{\mathbf{b}} L &= \sum_{t=1}^{\tau} \left(\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{b}^{(t)}} \right)^T \nabla_{\mathbf{h}^{(t)}} L = \sum_{t=1}^{\tau} \text{diag} \left(1 - (\mathbf{h}^{(t)})^2 \right) (\nabla_{\mathbf{h}^{(t)}} L) \\
\nabla_{\mathbf{v}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}} \right) (\nabla_{v^{(t)}} o_i^{(t)}) = \sum_{t=1}^{\tau} \nabla_{o^{(t)}} L \mathbf{h}^{(t)T} \\
\nabla_{\mathbf{w}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) (\nabla_{w^{(t)}} h_i^{(t)}) = \sum_{t=1}^{\tau} \text{diag} \left(1 - (\mathbf{h}^{(t)})^2 \right) (\nabla_{\mathbf{h}^{(t)}} L) \mathbf{h}^{(t)T} \\
\nabla_{\mathbf{u}} L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) (\nabla_{u^{(t)}} h_i^{(t)}) = \sum_{t=1}^{\tau} \text{diag} \left(1 - (\mathbf{h}^{(t)})^2 \right) (\nabla_{\mathbf{h}^{(t)}} L) \mathbf{x}^{(t)T}
\end{aligned}$$

There is no need to compute the gradient with respect to the input for training because it does not have any parameters as ancestors in the computational graph defining the loss. The main problem with RNNs is given by long-term dependencies: the more time steps the architecture goes through for optimisation and the more is exposed to Vanishing or Exploding Gradients, which means the gradient tends to 0 or to Infinity with much damage to the optimisation. This is due to the once-per-time-step composition of the same function Recurrent Networks. Given the following simple RNN, without any input or activation function:

$$\mathbf{h}^{(t)} = \mathbf{W}^T \mathbf{h}^{(t-1)}$$

It's possible to assume that the function composition process is equal to matrix multiplication. In this way the recurrence relation may be simplified to:

$$\mathbf{h}^{(t)} = (\mathbf{W}^t)^T \mathbf{h}^{(0)}$$

If matrix \mathbf{W} admits an eigendecomposition of the form $\mathbf{W} = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^T$, with orthogonal \mathbf{Q} the recurrence may be further simplified to:

$$\mathbf{h}^{(t)} = \mathbf{Q}^T \mathbf{\Lambda}^t \mathbf{Q} \mathbf{h}^{(0)}$$

This implies that, for eigenvalues with magnitude less than 1, the recurrence relation decays, and for eigenvalues with magnitude greater than 1, it explodes. Furthermore, this cannot be avoided by simply staying in a region where the phenomenon does not occur as, in order to store memories without them being altered by small perturbations, the RNN must enter a region of parameters where gradients vanish. In a model capable of representing long-term dependencies, the gradient of a short-term interaction has exponentially greater magnitude than the gradient of a long-term interaction. Because

of this, even the smallest fluctuations arising from short-term dependencies tend to hide long-term dependencies, causing the learning process to take a long time. It has been shown that the more the span of dependencies is increased, the harder gradient-based optimisation becomes, with the probability of successful training of RNNs via SGD reaching 0 for sequences of length 10 or 20. This problem can be dealt with by feeding the model the ground-truth output $y^{(t)}$ as the input at time step $t + 1$. This technique is called Teacher Forcing and emerges from the Maximum Likelihood Criterion - Machine Learning optimisation criteria used for probabilistic models. In this case, the model returns probability distributions characterised by the values assumed by a set of parameters and the goal is to determine the parameters which maximise the likelihood of a distribution to be approximate the target value (Gambosi, 2024). Given a sequence with 2-time steps:

$$\log p(y^{(1)}, y^{(2)} | x^{(1)}, x^{(2)}) = \log p(y^{(1)} | y^{(2)}, x^{(1)}, x^{(2)}) + \log p(y^{(1)} | x^{(1)}, x^{(2)})$$

Teacher Forcing specifies that the model's connections shall be fed with the correct target values rather than the model's own output. This allows to completely avoid BPTT, but as soon as hidden-to-hidden connections occur, the algorithm becomes necessary again. Moreover, RNNs without these connections are either shallow or go through a limited number of time steps and, thus, are rarely used. Still, this technique can be used with Back-Propagation Through Time to stabilise the Gradient.

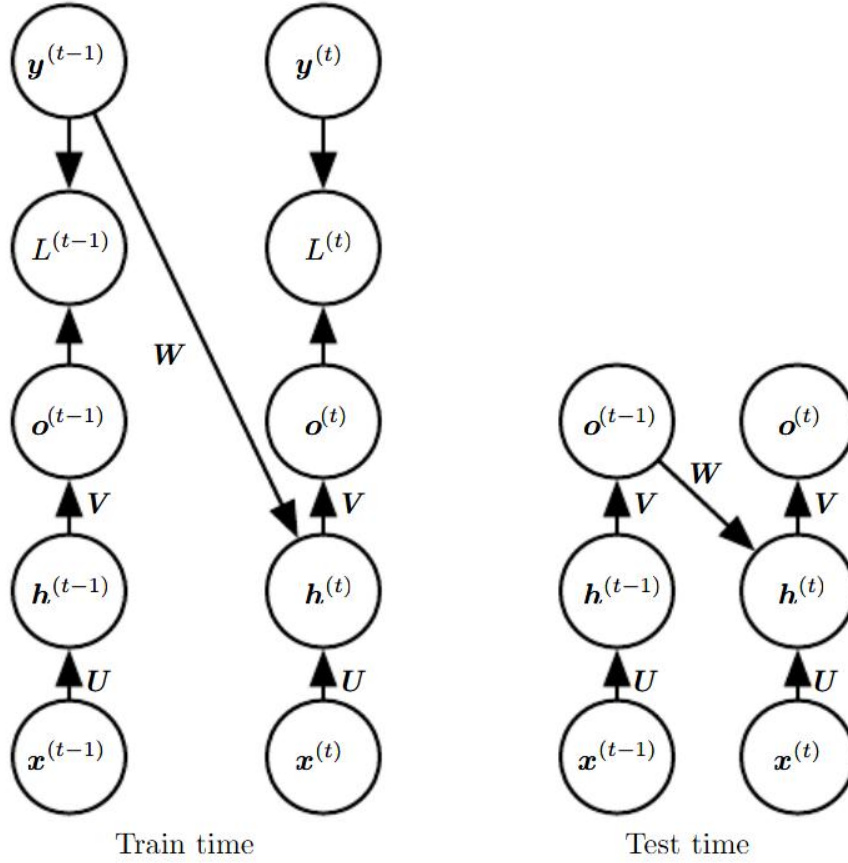


Figure 1.7: Representation of Teacher Forcing

Another way to control the Gradient and, more specifically, prevent it from exploding is Gradient Clipping. This technique consists of clipping the norm $\|\mathbf{g}\|$ of the gradient \mathbf{g} just before the parameter update:

$$\text{if } \|\mathbf{g}\| > T :$$

$$\mathbf{g} \leftarrow \frac{T}{\|\mathbf{g}\|} \mathbf{g}$$

Where T is the norm threshold and \mathbf{g} is used to update parameters. This prevents the algorithm from performing a detrimental step when the gradient explodes. This method, since the gradient is renormalised jointly with a single scaling factor, has the advantage of guaranteeing each step goes in the same direction as the actual gradient while bounding it to a certain threshold. However, this does not solve the Vanishing Gradient problem.

1.2.4 LSTM: Long-Short Term Memory Architecture

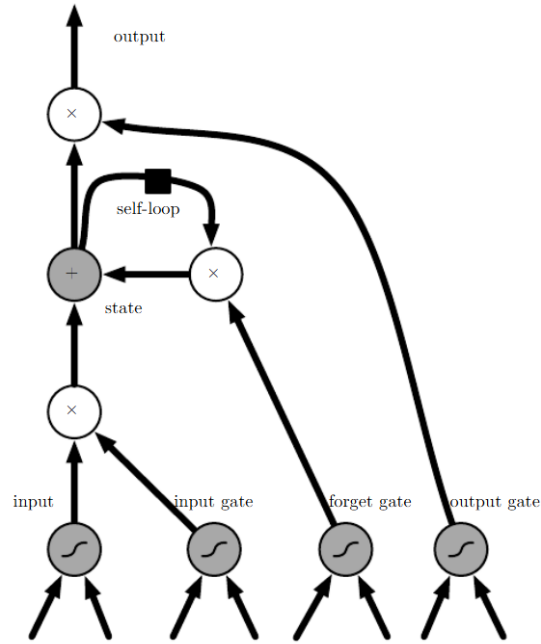


Figure 1.8: Block diagram of a LSTM recurrent network cell

One naïve solution to avoid Gradient Vanishing is to enforce constant error flow through the neuron. Given that neuron j 's local error back flow is described by:

$$\vartheta_j(t) = f'_j(net_j(t))\vartheta_j(t+1)w_{jj}$$

Where $\vartheta_j(t)$ is the error rate for unit j at time t , f_j is the activation function for unit j , net_j is the total input to unit j at time t , and w_{jj} is the recurrent weight of unit j back to itself or how much the past

output of unit j has influenced its current state. To enforce constant error flow, the following condition needs to be imposed:

$$f'_j \left(net_j(t) \right) w_{jj} = 1$$

This differential equation's solution is:

$$f_j \left(net_j(t) \right) = \frac{net_j(t)}{w_{jj}}$$

For an arbitrary $net_j(t)$. This means f_j must be linear, and unit j 's activation must remain constant.

$$y_j(t + 1) = f_j \left(net_j(t + 1) \right) = f_j \left(w_{jj}y^j(t) \right) = y^j(t)$$

The consistency of this equation can be assured by assuming $f_j(x) = x \forall x$ and $w_{jj} = 1$. This is referred to as Constant Error Carrousel (CEC). Of course, unit j will be connected to other units in the network, causing, however, two problems: Input weight conflict and Output weight conflict. The former is due to the weight w_{ji} connecting the input to unit j , which will receive conflicting signals during updates, attempting to make the parameter participate in storing the input (by switching on j) and protecting the input (by preventing j from being switched off by irrelevant later inputs); the latter is due to the outgoing weight w_{kj} being used for both retrieving j 's content at certain times and preventing j from disturbing k . These conflicts make learning more difficult. Moreover, while the so-called "short-time lag errors" can be reduced in early training stages, at later stages j may cause avoidable errors, making it more difficult to reduce "long-time lag errors". So, as the time lag increases stored information must be protected against perturbation for longer and longer and more and more correct outputs require protection against perturbation, making the learning process much more complicated. In 1997, S. Hochreiter and J. Schmidhuber presented a remedy by introducing a novel gradient-based method called Long Short-Term Memory (in their namesake article). The idea was to extend CEC by implementing additional units: the Input Gate unit, which is used to protect the memory contents stored in j from perturbation from irrelevant inputs, and the Output Gate unit, which is used to protect other units from perturbation by currently irrelevant memory contents stored in j . CEC is multiplied by these units, resulting in a more complex structure named Memory Cell,

contained within the hidden layer. The j -th Memory Cell is denoted c_j . The memory cell gets input from the Output Gate out_j and from the Input Gate in_j . out_j and in_j 's activation times are given by:

$$y^{out_j} = f_{out_j}(net_{out_j}(t))$$

$$y^{in_j} = f_{in_j}(net_{in_j}(t))$$

Where:

$$net_{out_j}(t) = \sum_u w_{out_j u} y^u(t-1)$$

$$net_{in_j}(t) = \sum_u w_{in_j u} y^u(t-1)$$

There is also:

$$net_{c_j}(t) = \sum_u w_{c_j u} y^u(t-1)$$

Which represents the memory update. The summation index u may stand for input unit, output unit, memory cell or conventional hidden units, if there are any, as any of these may convey useful information about the current state of the net. At time t , c_j 's output is computed as:

$$y^{c_j}(t) = y^{out_j}(t)h(s_{c_j}(t))$$

Where h is Output Squashing Function, usually a centred sigmoid with range $[-1,1]$, and $s_{c_j}(t)$ is the internal state at time t , described by:

$$s_{c_j}(0) = 0$$

$$s_{c_j}(t) = s_{c_j}(t-1) + y^{in_j}(t)g(net_{c_j}(t)) \text{ for } t > 0$$

Where g is the Net Input Squashing Function, usually a logistic sigmoid function centred around $[-2,2]$. The presence of these gates avoids weight conflicts by using in_j to decide when to keep or override information in Memory Cell c_j and out_j to decide when to access the Memory Cell and when to prevent other units from being perturbed by c_j . Error signals trapped within a memory cell's CEC cannot change. The Output Gate will have to learn which errors to trap in its CEC, by

appropriately scaling them, while the Input Gate will have to learn when to release them. This configuration allows information to be stored across arbitrary time lags and error signals to be carried far back in time. However, this may cause the cell states to grow linearly during the presentation of a time series, leading them to grow unbounded. This implies saturation of the Output Squashing Function h , making its derivative vanish and making the cell output equal to the Output Gate activation, causing the entire memory cell to degenerate into an ordinary cell unit. This can be solved by replacing standard LSTM's constant weight $w_{jj} = 1$ by the multiplicative Forget Gate activation y^φ (F.A. Gers, 1999). This gate is calculated like the activation of the other gates and squashed using a logistic sigmoid $[0,1]$:

$$y^{\varphi_j}(t) = f_{\varphi_j} \left(\sum_m w_{\varphi_j m} y^m(t-1) \right)$$

y^{φ_j} works as the weight of the self-recurrent connection of the internal state s_{c_j} , whose update equation becomes:

$$s_{c_j}(0) = 0$$

$$s_{c_j}^u(t) = y^{\varphi_j}(t) s_{c_j}^u(t-1) + y^{in_j}(t) g \left(net_{c_j}^u(t-1) \right)$$

Bias weights for LSTM gates are initialised with negative values for input and output gates and positive values for the forget gate. LSTMs are trained using the same Back Propagation Through Time used for standard Recurrent Neural Networks. Different Memory Cells can be packed into Memory Cell Blocks, sharing the same Input Gate and the same Output Gate, thus, facilitating information storage. Since only the derivatives $\frac{\partial s_{c_j}}{\partial w_{i_l}}$ need to be stored and updated, LSTM has an update complexity of $O(W)$, where W is the number of weights. The LSTM algorithm has proven to be greatly proficient in solving complex tasks such as unconstrained handwriting recognition, speech recognition, handwriting generation, machine translation, image captioning and parsing.

1.2.5 The Attention Mechanism and the Transformer

The inherently sequential nature of RNNs and LSTMs becomes a problem for long sequences: it precludes parallelization within training examples, thus limiting batch across examples because of memory constraints. In the paper “Attention Is All You Need” (A. Vaswani, 2017), the authors

propose as a solution to this problem the Transformer, a model architecture avoiding recurrence and relying entirely on the so-called Attention mechanism to draw global dependencies between input and output. The Attention mechanism was born as a solution to a problem occurring in Neural Machine Translation, an approach to Machine Translation which consists of building and training a single, large Neural Network that reads a sentence and outputs a sentence. From a probabilistic perspective, translation is equivalent to finding a target sentence y which maximises the conditional probability of y given a source sentence x . Thus, in Neural Machine Translation, a parametrised model is fitted to maximise the conditional probability of sentence pairs using a training corpus. In 2014, according to D. Bahdanau, K. Cho and Y. Bengio, LSTMs achieved close to state-of-the-art performance of the conventional phrase-based machine translation systems existing at that time, while adding Neural components to existent translation systems allowed to surpass the previous state-of-the-art. The main problem with this technology resides within its necessity to compress all the necessary information into a fixed-length vector. This made it difficult for Neural Networks to cope with long sentences, especially those that were longer than the sentences in the Training Corpus. In fact, the longer the sentences were, the more rapidly the Architecture performance deteriorated. The traditional architecture for Neural Machine Translation was the RNN Encoder-Decoder framework, first described in “Sequence to Sequence Learning with Neural Networks” (I. Sutskever, 2014). In this system, an encoder reads the input sentence as a sequence of vectors $x = (x_1, \dots, x_{T_x})$ in a vector c . Generally, the RNN is used so that:

$$h_t = f(x_t, h_{t-1})$$

$$c = q(\{h_1, \dots, h_{T_x}\})$$

Where $h_t \in \mathbb{R}^n$ is a hidden state at time t , and c is a vector generated from the hidden states, while f and q are non-linear functions or actual LSTMs. The decoder is trained to predict the next word y_t given the context vector c and all the previously predicted words $\{y_1, \dots, y_{t'-1}\}$. Therefore, the decoder defines a probability over the translation by decomposing the joint probability into the ordered conditionals:

$$p(y) = \prod_{t=1}^T p(y_t | \{y_1, \dots, y_{t-1}\}, c)$$

With an RNN, each conditional probability is modelled as:

$$p(y_t | \{y_1, \dots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

Where g is a non-linear possibly multilayered function that outputs the probabilities and s_t is the hidden state of the RNN. The idea behind the Attention Mechanism is to relieve the encoder from the burden of having to encode all the information in the source sentence into a fixed-length vector; by letting the decoder decide which parts of the sentence it should pay attention to.

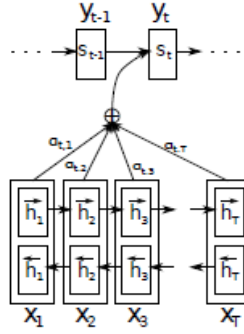


Figure 1.9: Graphical Illustration of the Encoder-Decoder model that enabled Attention Mechanism

In this context, conditional probabilities are defined as:

$$p(y_i | y_1, \dots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

Where s_i is an RNN hidden state for time i , determined by:

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

Unlike the standard architecture, the probability is conditioned on a distinct context vector c_i for each target word y_i . The context vector is computed as:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Where h_j represents an annotation containing information about the whole input sequence with a strong focus on the parts surrounding the i -th word of that sentence and α_{ij} measures the importance

of h_j with respect to the previous hidden state s_{i-1} in deciding the next state s_i and generating y_i . The weight α_{ij} of each annotation is computed by:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

Where

$$e_{ij} = a(s_{i-1}, h_j)$$

Is an alignment model (a Feedforward Neural Network jointly trained with all the other components of the proposed match) which evaluates how well the inputs around position j and the output around position i match. This way, the context vector becomes the expected annotation over all the possible annotations with probabilities α_{ij} . To make sure the annotations for each word summarise not only the preceding words but also the following ones, a Bidirectional RNN is used (D. Bachanau, 2015). The Transformer, relying exclusively on this Attention Mechanism, manages to fix the number of operations he needs to make, instead of making it grow with the distance between two input and output positions.

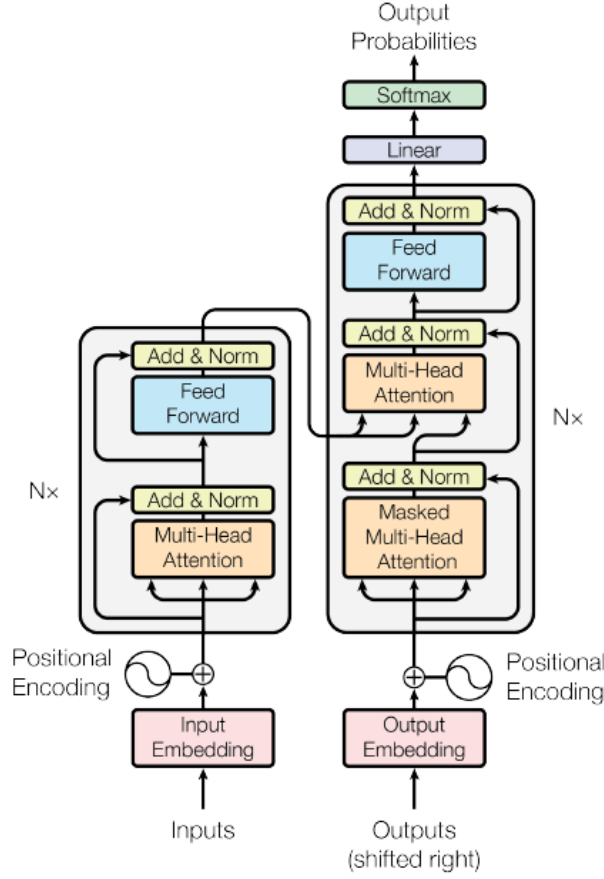


Figure 1.10: Graphical representation of the Transformer

As Figure 2.10 shows, the Transformer follows the Encoder-Decoder general structure, therefore mapping an input sequence of symbol representations (x_1, \dots, x_n) to a sequence of continuous representations $\mathbf{z} = (z_1, \dots, z_n)$ and then generating an output sequence (y_1, \dots, y_m) . The model is auto-regressive, consuming the previously generated symbols as additional input when generating the next ones. Both Encoder and Decoder are composed of a stack of N layers, each containing a certain number of sub-layers. The Encoder has 2 sub-layers per layer: the first is a multi-head self-attention mechanism, and the second is a fully connected Feedforward Neural Network. A residual connection is employed around each of the sub-layers, followed by layer normalisation. This means that the output of each sub-layer is $LayerNorm(x + Sublayer(x))$ where $Sublayer(x)$ is the function implemented by the sub-layer itself. The Decoder inserts a third sub-layer, which performs multi-head attention over the output of the Encoder stack. In addition to the employing of a residual connection, the self-attention layer is modified in order to prevent positions from attending to subsequent positions. This operation, combined with the fact that the output embeddings are offset by one position, ensures that the prediction for position i can depend only on the known outputs at

positions less than i . The 2 most commonly used attention function are additive attention (which has already been described) and dot-product (or multiplicative attention). In the case of the Transformer, the authors who first presented this model in 2017 used Scaled Dot-Product Attention, computed as:

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

Where Q is the item looking for relevant information, K is the index of the information available, V is the actual information returned in response to a query, and d_k is the queries and keys' dimension. The main difference with Dot-Product Attention is the presence of the scaling factor $\frac{1}{\sqrt{d_k}}$, which counteracts the occurrence of extremely small gradients, due to large values of d_k pushing the softmax function into regions where this can happen.

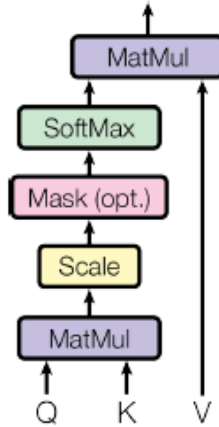


Figure 1.11: Scaled Dot-Product Attention

Instead of performing a single attention function with a d_{model} – dimensional keys, values and queries, in the Transformer queries, keys and values are linearly projected h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values the attention function is then performed in parallel, producing d_v output values. These are concatenated and projected once more, resulting in the final values. This is Multi-head Attention.

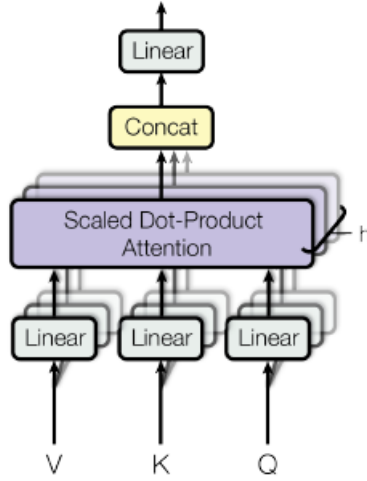


Figure 1.12: Multi-head Attention

Differently from Single-head Attention, Multi-head Attention allows the model to jointly attend to information from different representation subspaces at different positions. Therefore, Multi-head Attention's output is:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{Hd_v \times d_{model}}$. The parameter matrices are also used to compute the Attention parameters Q, K , and V :

$$Q = XW^Q$$

$$K = XW^K$$

$$V = XW^V$$

Multi-head Attention is used in 3 different ways:

- In Encoder-Decoder Attention layers to allow every position in the Decoder to attend over all positions in the input sequence, by taking the queries from the previous Decoder layer and the memory keys and values from the Encoder's output.
- In the Encoder's Self-Attention layers, where all the keys, values and queries come from the same place, that is the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layers of the encoder.
- In the Decoder's Self-Attention Layers, likewise, each position in the Decoder is allowed to attend to all positions in the decoder up to and including that position. This prevents past

information to flow in the decoder and preserves the auto-regressive property. This is implemented by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections.

Self-Attention layers make the Transformer better at total computational complexity for layer, amount of computation which can be parallelised and path-length between long-range dependencies in the network than a Recurrent layer. An RNN requires $O(n)$ sequential operations, whereas a Self-Attention layer connects all positions with a constant number of sequentially executed operations. Moreover, until $n < d$ (which occurs in most cases), Self-Attention layers are faster than recurrent layers. To improve computational performance for tasks involving very long sequences, Self-Attention can be restricted to considering only a neighbourhood of size r , increasing the maximum path length to $O(n/r)$.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Table 1.1: Complexity per Layer, Sequential Operations and Maximum Path Length per Layer Type. n is the sequence length, d is the representation dimension, and r is the size of the neighbourhood.

The Transformer proved to be significantly faster at training compared to other architectures, achieving state-of-the-art performance in machine translation tasks (WMT 2014 English-to-German and WMT 2014 English-to-French). The Transformer’s first version had $N = 6$ layers, $h = 8$ parallel attention layers, queries, keys, and values dimensionality $d_k = d_v = \frac{d_{model}}{h} = 64$, input-output dimensionality $d_{model} = 512$, and inner-layer dimensionality $d_{ff} = 2048$. This architecture revolutionised the Artificial Intelligence world, enabling that series of innovations which led to nowadays Artificial Intelligence models (A. Vaswani, 2017).

1.2.6 Decoder-only models: the birth of Large Language Models

The Transformer’s effectiveness in solving linguistic tasks (particularly Machine Translation) led researchers to fully exploit its features to achieve higher performance in Language Modelling tasks. In 2018 OpenAI (today one of the most important AI firms) published “Improving Language Understanding by Generative Pre-Training” (A. Radford K. N., 2018), in which the authors tried to

exploit the architecture to leverage raw texts for learning linguistic tasks. More specifically, the company's researchers decided to pre-train the Transformer on an unsupervised Corpus of data and then to fine-tune the model by adapting the parameters to a supervised target task, using a labelled dataset. Given an unsupervised corpus of tokens $U = \{u_1, \dots, u_n\}$, they used standard language modelling objective to maximise the following likelihood:

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

Where k is the size of the context window, P is the conditional probability, and Θ is the set of parameters of the Neural Network that is modelling probability P , trained with Stochastic Gradient Descent. In their experiments, they used a multi-layer Transformer decoder applying a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$h_0 = UW_e + W_p$$

$$h_l = \text{transformer_block}(h_{l-1}) \forall i \in [1, n]$$

$$P(u) = \text{softmax}(h_n W_e^T)$$

Where $U = (u_{-k}, \dots, u_{-1})$ is the context vector of tokens (i.e. words in the Corpus of textual data), n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix. After pre-training the model, they used a labelled dataset $\mathcal{C} = \{\mathbf{x}, \mathbf{y}\}$, where \mathbf{x} was the sequence of input tokens, and \mathbf{y} was the list of associated labels. The inputs were passed through the pre-trained model to obtain the final transformer block's activation h_l^m , which was then fed into an output layer with parameters W_y to predict \mathbf{y} :

$$P(\mathbf{y}|\mathbf{x}) = \text{softmax}(h_l^m W_y)$$

Therefore, the following objective needed to be maximised:

$$L_2(\mathcal{C}) = \sum_{(\mathbf{x}, \mathbf{y})} \log P(\mathbf{y}|\mathbf{x})$$

Additionally, the authors found that including language modelling as an auxiliary objective to the fine-tuning helped learning by improving generalisation and accelerating convergence. Thus, the final goal function becomes:

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda L_1(\mathcal{C})$$

This was the first time a Decoder-only model was used for Machine Learning and marked the beginning of the successful GPT (Generative Pre-trained Transformer) series of models. GPT-1 consisted of 12 layers with masked Self-Attention heads (768 dimensional states and 12 Attention heads) and 3072 dimensional inner states for the position-wise Feedforward Network and achieved state-of-the-art performances in Natural Language Inference (up to +5.8% Accuracy compared to previous state-of-the-art), Question answering and commonsense reasoning (up to +8.9% Accuracy), Semantic Similarity (up to +4.2% Accuracy), and Classification (up to +10.4% Accuracy). The authors made the hypothesis that the effectiveness of their model was due to the model learning to perform many of the tasks to improve its language modelling capability and that the more structured attentional memory of the Transformer assisted in transfer compared to LSTMs. This and the suspicion that the prevalence of single task training on single domain datasets was a major contributor to the lack of generalisation observed in systems in 2019 led researchers to significantly scale up the architecture of GPT. In the paper “Language Models are Unsupervised Multitask Learners” (A. Radford J. W., 2019), the authors presented GPT-2 which had 48 layers and 1542 million parameters, making it the first Large Language Model (for comparison, GPT-1 only had 117 million parameters). OpenAI’s researchers thought that increasing the model’s size and the context window (from 512 to 1024 tokens) enabled it to learn without explicit supervision. The experiment succeeded as GPT-2 was able to solve several tasks with high accuracy rates, improving the state-of-the-art accuracy in some cases – it achieved 70.7% accuracy in the Winograd Schema challenge for commonsense reading and improved the state-of-the-art by 7%. Furthermore, the more parameters, the closer the model’s performances to human proficiency in linguistic tasks. In 'Language Models Are Few-Shot Learners' (OpenAI, 2020), Open AI researchers proved that scaling up language models improves task-agnostic, few-shot performance, introducing GPT-3. This iteration of the GPT series further increased the number of parameters up to 175 billion, for a total number of layers equal to 96 and a context window of 2048 tokens. This model, after being trained on unsupervised datasets, was already able to solve tasks without updating its weights, bypassing fine-tuning. More specifically, the

researchers analysed the model's performance after in-context learning, that is using the text input of a pre-trained language model as task specification. This means GPT-3 was conditioned on natural language instruction and was then expected to complete further instances of the task by predicting what came next. There are 3 approaches for In-context learning:

- Few-Shot Learning: the model is provided with a few demonstrations of the task (OpenAI researchers set a number of examples in a range of 10 to 100)
- One-Shot Learning: the model is provided with only one demonstration of the task
- Zero-Shot Learning: the model is only given a natural language description of the task, without demonstrations.

Not only could GPT-3 perform well on a variety of tasks, but it matched and surpassed state-of-the-art performance on LAMBADA (sentence completion task which consisted of predicting the last word of a sentence) and Winograd-Style tasks (understanding word-pronoun links) even with Zero-Shot Learning. Moreover, GPT-3 was already able to generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. 80 US-based volunteers participated to a quiz in which they were asked to tell whether some text was written by humans or machine. Moreover, GPT-3 was already capable of generating synthetic news articles from titles and subtitles. In an experiment involving 80 US-based volunteers, participants were asked to determine whether the resulting text completions of 25 existing articles were written by a human or generated by a machine. The mean human accuracy at detecting text produced by GPT-3 was just barely above chance: 52%. This milestone suggested that LLM had already reached a point where they could effectively play Turing's Imitation Game. Parallely, Google's researchers adopted a different approach, exploiting the Transformer's encoding capabilities: in "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" (J. Devlin, 2018), the authors presented BERT (Bidirectional Encoder Representations from Transformers) the first Encoder-only model, designed to pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context in all layers. Models such as the first GPT suffered from their unidirectionality which limited the choice of architecture and restricted the power of pre-trained representation: every token in OpenAI's first model could only attend previous tokens in the self-attention layer of the Transformer, making it more difficult for the model to solve tasks such as question answering, where it is crucial to incorporate text from both directions. BERT was presented in 2 different versions: Base, chosen to have the same model size as GPT for comparison purposes

(12 Layers, 768 dimensional states, 12 Attention heads and 110M parameters); and Large, whose size was doubled (24 Layers, 1024 dimensional states, 16 Layers and 340M parameters). The model was first pre-trained on 2 unsupervised tasks:

- Masked Language Modelling: after having randomly masked some of the tokens from the input, the objective is to predict the original vocabulary id of the masked word based only on its context.
- Next Sentence Prediction: 2 sentences A and B are input into the model at the same time to predict whether sentence B comes after sentence A in the same document. For 50% of the time, B is the actual next sentence that follows A and for the other 50% of the time, it is just a random sentence (Y. Sun, 2022).

Then it was fine-tuned on several different tasks to evaluate its performance: both versions of BERT advanced the state of the art on GLUE (General Language Understanding Evaluation benchmark), SQuAD v1.1 (Stanford Question Answering Dataset), SQuAD v2.0 (it extends v1.1 by allowing for the possibility that no short answer exists in the provided paragraph), and SWAG (Situations With Adversarial Generations dataset: given a sentence, the task is to choose the most plausible continuation among 4 options). The pre-training approach adopted by BERT largely influenced future developments and led researchers to further investigate this technique and OpenAI to produce their models GPT-2 and GPT-3 the way they did.

1.2.7 How to train Large Language Models

Large Language Models imply a paradigm shift in model training as their size makes it arduous to re-adapt parameters to a new supervised or unsupervised task. If the computational cost of the traditional training process is $O(n)$, with n equal to the number of nodes present in the Neural Network and each node has a parameter, it is easy to understand that a model with 175 billion parameters will require much more time and resources than a simple Feedforward Neural Network. Furthermore, pre-trained neural language models have been shown to learn a substantial amount of knowledge from data. They can do so without any access to an external memory, as a parametrised implicit knowledge base. But they cannot easily expand or revise their memory, nor can they provide insights into their predictions and may produce hallucinations (misleading, incorrect or fabricated outputs). This renders Large Language Models less effective at Knowledge-Intensive tasks, if compared to specialised models. Therefore, researchers and AI practitioners worked on several different approaches to improve the models' results:

- Fine-tuning: Weights readaptation to a specific task. Effective, as proven in “Improving Language Understanding by Generative Pre-Training” but it is the most computationally expensive training process. Examples of fine-tuned models are GrammarlyGo, for English writing assistance, and Copy.ai, for sales and copywriting.
- Prompt-based fine-tuning or In-context Learning: the training process that researchers used for GPT-3. The model is instructed via Natural Language prompts with a few to no examples.
- Retrieval Augmented Generation (RAG): the pre-trained model is endowed with non-parametric memory through a general-purpose fine-tuning approach referred to as RAG. The non-parametric memory source (a PDF document, a text file etc.) is first chunked and then embedded into vectors z . So, the prompt x is used to retrieve information from z and generate an output y through it. The retrieved document is treated as a latent variable. RAG produces a probability distribution over generated text by marginalising over the latent document. There are 2 approaches for this:

- RAG-Sequence: the model uses the retrieved document to generate the complete sequence of text. The whole latent document is marginalised to get the seq2seq probability $p(y|x)$ via a top K approximation:

$$\begin{aligned}
p_{RAG-sequence}(y|x) &\approx \sum_{z \in top-k(p(\cdot|x))} p_{\eta}(z|x) p_{\theta}(y|x, z) \\
&= \sum_{z \in top-k(p(\cdot|x))} p_{\eta} \prod_i^N p_{\theta}(y_i|x, z, y_{1:i-1})
\end{aligned}$$

Concretely, the top K documents in the vectors are retrieved a Dense Passage Retriever based on a bi-encoder architecture and the generator produces the output sequence probability for each document, which are then marginalised. Then the output sequence is produced according to the $\arg \max_y p(y|x)$, which is approximated via Thorough Decoding – for each document z for which y does not appear in the beam search, an additional forward pass is run, the generator probability $p_{\theta}(y_i|x, z, y_{1:i-1})$ is multiplied by $p_{\eta}(z|x)$ and then the probabilities are summed across beams for the marginals – or Fast Decoding - $p_{\theta}(y|x, z_i)$ is approximated to 0 when y was not generated during the beam search. This is useful for longer output sequences.

- RAG-Token: for each target token, a different latent document is drawn and marginalised accordingly:

$$p_{RAG-Token}(y|x) \approx \prod_i^N \sum_{z \in top-k(p(\cdot|x))} p_{\eta}(z|x) p_{\theta}(y_i|x, z_i, y_{1:i-1})$$

More explicitly, the top K documents are retrieved thanks to the retriever and then the generator produces a distribution for the next output token for each document before marginalising.

Researchers who first tested RAG noted how, not only was the deployed model’s performance better at tasks such as Open-domain Question Answering (outperforming fine-tuned models by over 10% in terms of Accuracy in the case of Trivia Question Answering) but it could also generate, in some cases, correct answers even when they were not retrieved in any document. Furthermore, when tested on Abstractive Question Answering, the RAG approach proved to be less prone to hallucinations and generate factually correct text more than other models (P. Lewis, 2020).

- Low-Rank Adaptation (LoRA): instead of retraining all model parameters, the output of the weight matrix $W_0 \in \mathbb{R}^{d \times k}$ is constrained by representing the latter with a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$ and the rank $r \ll \min(d, k)$. This way, W_0 does not receive gradient updates, while A and B contain updatable parameters. For $h = W_0 x$, the modified forward pass yields:

$$h = W_0 x + \Delta W x = W_0 x + B A x$$

This approach allows for quick task switching while retaining high model quality and input sequence length, without introducing inference latency (E. Hu, 2021).

1.2.8 Aligning LLMs with the user’s will: Chain of Thought and Instruct-GPT

A Large Language Model’s goal is to predict the next coming word after a sequence of text. This implies that the model is not guaranteed to provide value to the user on its own: the content that the AI model produces might be misaligned with the user’s intention and could be useless, if not even harmful. Moreover, LLMs are not capable of basic mathematical reasoning on their own, incurring in elementary mistakes. Researchers tried to solve this problem using two different approaches: Chain of Thought and Instruct-GPT. The former emulates human reasoning by decomposing, via prompting, the problem the AI is asked to solve into smaller and simpler problems which can lead to the solution of the main problem. This technique, not only does it allow models to better perform in math-word problems and commonsense reasoning but also allows researchers to understand how LLMs arrive at

an answer and debug where the reasoning path went wrong (despite characterising a model's computations that support such reasoning is impossible).

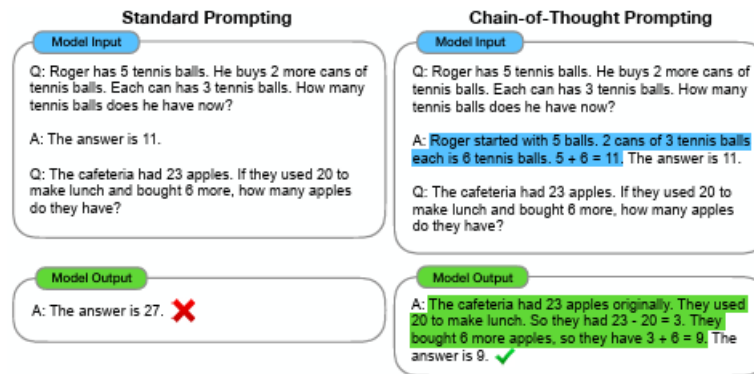


Figure 1.13: Example of Chain-of-Thought prompting compared to Standard Prompting.

The Google researchers who first tried this approach evidenced 3 key takeaways:

- Chain-of-thought prompting does not positively impact performance for small models and only yields performance increases when used with models of 100B parameters or more.
- The more complicated the problem is, the better chain-of-thought prompting works.
- Chain-of-thought prompting allowed AI models to overcome the previous state of the art in Arithmetic Reasoning, Commonsense reasoning and Symbolic Reasoning (J. Wei, 2023).

The chain-of-thought approach led also to the creation of the “o” series of models by OpenAI that excels at reasoning tasks: according to the AI firm, the first model, o1, ranks in the 89th percentile on Codeforces, dataset of competitive programming questions, places among the top 500 students in the US students in a qualifier for the USA Math Olympiad (AIME) and performs better than PhDs on the GPQA benchmark of physics, biology and chemistry problems (OpenAI, 2024). Instruct-GPT exploits Reinforcement Learning from Human Feedback to fine-tune the Large Language Model to follow a broad class of written instructions in order to make it act in accordance with the user’s intention. Given the pre-trained language model, a distribution prompt on which the model is supposed to produce aligned outputs and a team of trained human labellers, the approach consists of:

- Collect demonstration data and train a supervised policy: labellers provide demonstrations of the desired behaviour on the input prompt distribution which are then used for fine-tuning the Large Language Model via supervised learning.
- Collect comparison data and train a reward model: a dataset of comparisons between model outputs, where labellers express their preferences over outputs for a given input. Then a

reward model is trained to predict the human-preferred output. More specifically, the model is trained to take in a prompt and a response and output a scalar reward. Labellers are presented with K responses from the LLM to rank (in the paper where OpenAI presented this technique, K ranged between 4 and 9), producing $\binom{K}{2}$ comparisons that are used for the reward model training. Its loss function is:

$$loss(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D} \left[\log \left(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)) \right) \right]$$

Where $r_\theta(x, y)$ is the scalar output of the reward model for prompt x and completion y with parameters θ , y_w is the preferred completion out of the pair made of y_w and y_l , and D is the dataset of human comparisons.

- Optimise a policy against the reward model using Proximal Policy Optimisation: The Supervised Fine-Tuned (SFT) model is fine-tuned once again using the Reinforcement Learning mechanism known as Proximal Policy Optimisation. The environment where the Fine-Tuning is executed is a bandit environment (the model is given a limited number of trials to choose among a set of alternatives) which presents a random customer prompt and expects a response to it. Given the prompt and response, the reward model returns a scalar reward and ends the episode. A per-token KL penalty from the SFT model at each token is added to mitigate over-optimisation of the reward model. The value function is initialised from the RM. This is the Proximal Policy Optimisation. The final objective is to maximise the following function in Reinforcement Learning Training:

$$objective(\phi) = E_{(x, y) \sim D_{\pi_\phi^{RL}}} [r_\theta(x, y) - \beta \log(\pi_\phi^{RL}(y|x) / \pi^{SFT}(y|x))] + \gamma E_{x \sim D_{pretrain}} [\log(\pi_\phi^{RL}(x))]$$

Instruct-GPT's results using GPT-3 were preferred $85 \pm 3\%$ of the time to plain GPT-3 and $71 \pm 4\%$ of the times to few-shot GPT-3 (L. Ouyang, 2022). This mechanism allows Large Language Models to follow instructions in a prompt and provide a detailed response. OpenAI in 2022 released a sibling model trained in a similar way to Instruct-GPT to interact in a conversational way, answer follow-up questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests: ChatGPT (OpenAI, 2022), the app that drew the attention of masses onto Artificial Intelligence.

1.2.9 LLMs' criticalities

Over the years, Large Language Models increased their size and improved their performance, achieving unprecedented results in tasks which were thought to be unfeasible. GPT o4-mini, OpenAI's latest model, thanks to the Chain of Thought approach, has achieved 93.4% accuracy at AIME 2024 Competition Math and 81.4% accuracy at GPQA Diamond, a dataset of PhD level Science Questions (OpenAI, 2025); Claude 3.7 Sonnet, the latest Large Language Model by the AI company Anthropic, is being tested on interacting with a Computer like a human would, moving a cursor, looking at a screen and writing text (Anthropic, 2025); Gemini 2.5, Google's most advanced LLM, along with reasoning capabilities on par with GPT's latest models has a context window of 1 million tokens, which means it can take as an input even a 1-hour long video (Google, 2025). According to the AI Index 2025 Annual Report by Stanford University (AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, 2025), are constantly growing and datasets used for training are doubling in size every 8 months.

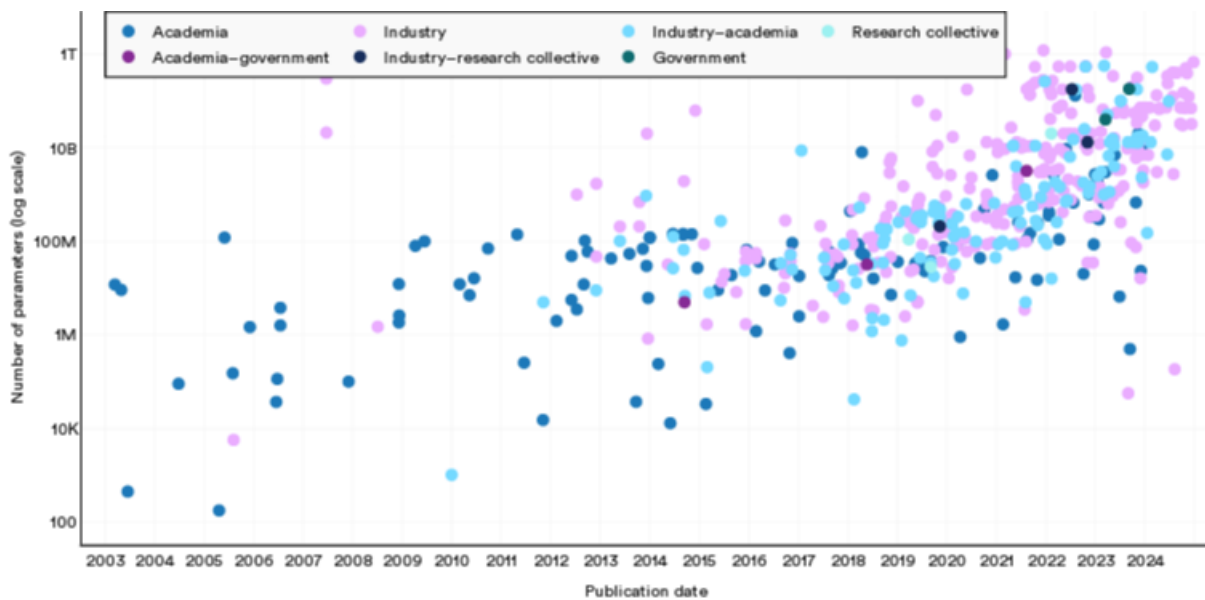


Figure 1.14: Scatterplot representation of Large Language Models in terms of Number of Parameters and Publication Date. The colour distinguishes which type of firm produced a model. The number of parameters in models is growing by the year.

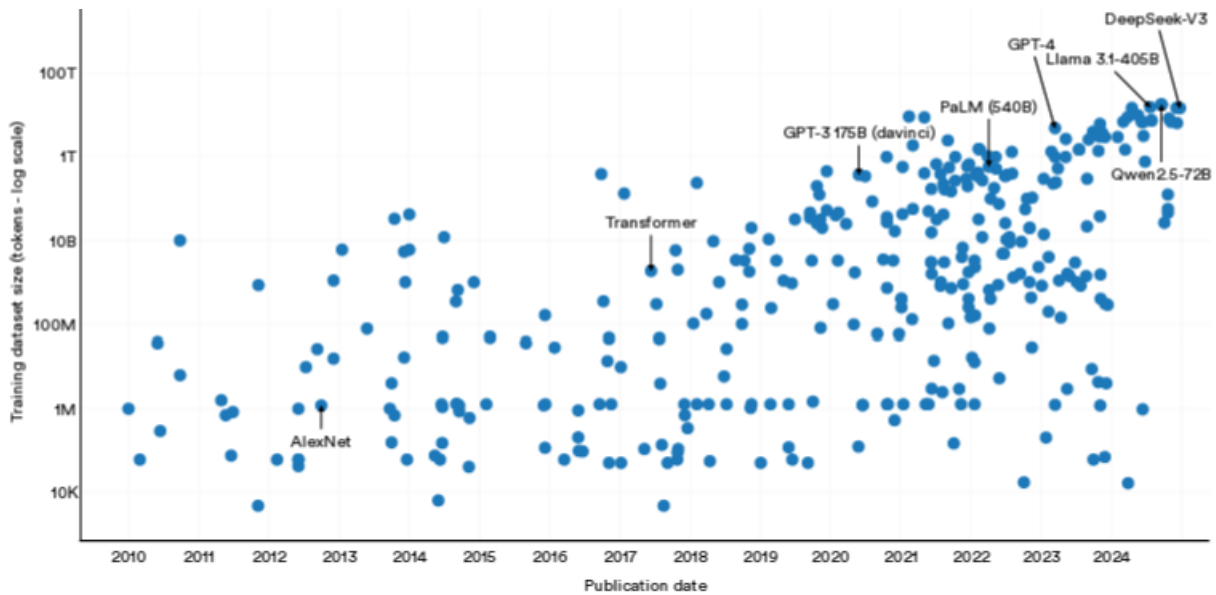


Figure 1.15: Scatterplot representation of Large Language Models in terms of Training Datasets size and Publication Date. According to the 2025 AI Index Report, Training Datasets size doubles every 8 months.

Despite this, Inference Costs (that is, the output costs sustained for each input prompt) significantly decreased: according to the same 2025 AI Index Report, the inference cost for an AI model scoring the equivalent of GPT-3.5 (64.8%) on MMLU (Massive Multitask Language Understanding) dropped from \$20.00/million tokens to \$0.07/million tokens, while the cost of models scoring above 50% on GPQA (Graduate-level Google-Proof Question Answering Benchmark, which is more challenging than MMLU) dropped from \$15.00/million tokens to \$0.12/million tokens. Moreover, despite state-of-the-art models remaining more expensive, the market already presents smaller and cheaper models which can be exploited.

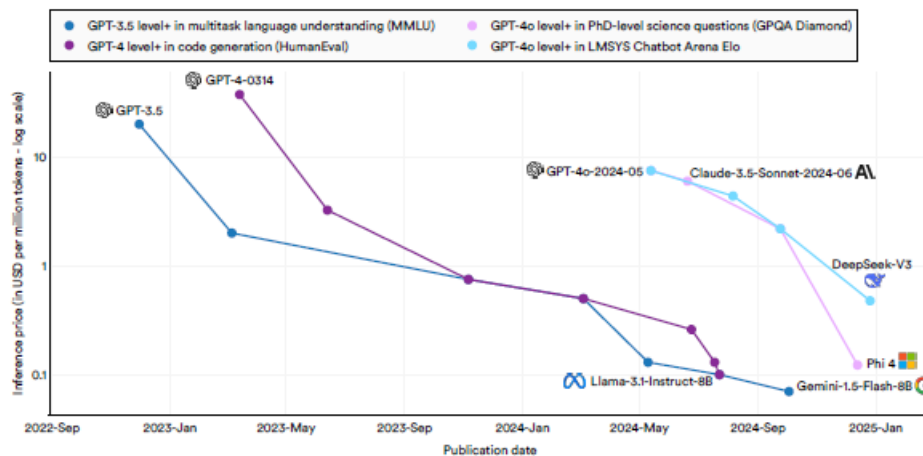


Figure 1.16: Line plot representation of the Inference price decrease between September 2022 and January 2025.

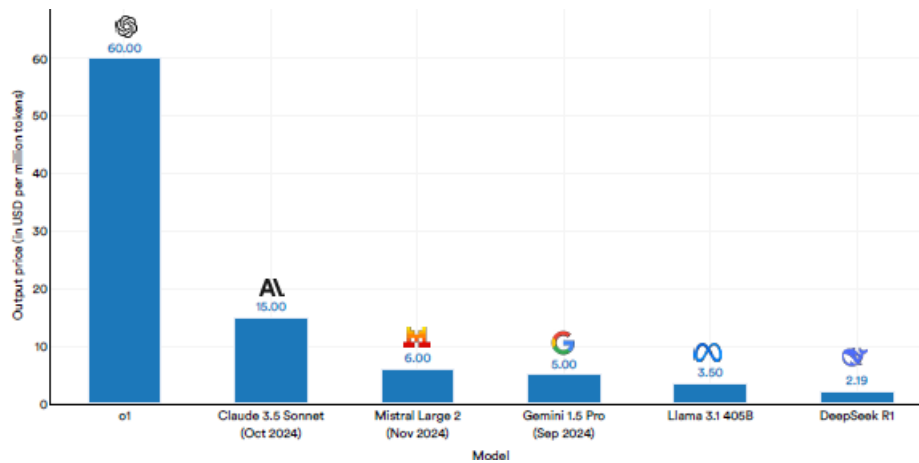


Figure 1.17: Output price per million tokens comparison between competitors

However, scaling up Large Language Models is causing significant drawbacks. For instance, Training Costs are increasing. While the first Transformer required just \$670 for Training, later models such as Gemini 1.0 Ultra, Llama 3.1-406B, and Grok-2, according to Epoch AI estimations, cost more between \$100 million and \$200 million each for their training. It is important to underline that these are only estimations: training costs for OpenAI's GPT-4 were estimated around \$79 million whereas OpenAI's CEO Sam Altman declared that training costs for the same model exceeded \$100 million. This means that these costs might be even higher, fact already corroborated by Anthropic's CEO Dario Amodei, who noted that model training runs costing around \$1 billion were already underway. The only exception to this trend appears to be DeepSeek-V3, the AI model by the Chinese company HighFlyer, whose training cost was around \$6 million, although some reports have disputed the stated cost of the LLM suggesting the actual development costs were significantly higher. This cost increase, in the long run, can impact consumers and damage firms who are already relying on LLMs.

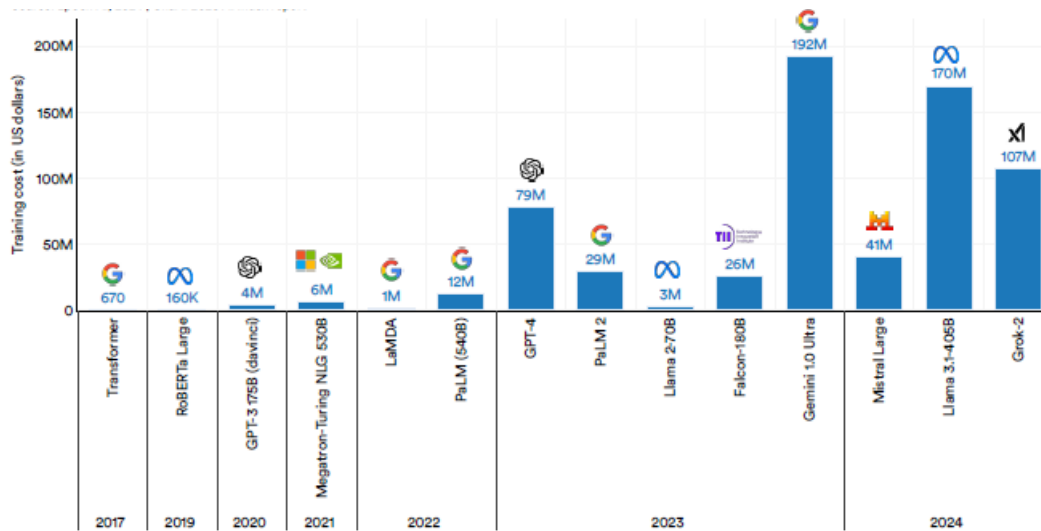


Figure 1.18: Barplot representation of Large Language Models Training Costs.

Moreover, larger models and greater datasets lead to longer training times. Google’s Gemini 1.0 Ultra reportedly took around 100 days to train, while the typical window by today’s standards would be 90 days. The longer the training, the more power Large Language Models will require. Llama 3.1-405B, released in Summer 2024, required 25.3 million W. This is due to the massive computational resources required for training and deployment: Training and Inference rely on GPUs (Graphical Processing Units), more powerful than CPUs (Computer Processing Units) but also more energy expensive, and Data Centre Infrastructure which also requires great amounts of Power and Water for cooling – several articles and reports esteem ChatGPT consumes for training and inferences amounts of water in the order of millions of litres (P. Li, 2023; Ren, 2023). According to EpochAI, the power required to train frontier AI models is doubling annually, reflecting the trend of training models on increasingly large datasets. This also negatively influences carbon emissions: in Stanford’s report it is estimated that GPT-4 emitted around 5184 tons of Carbon during training, Llama 3.1 around 8930 tons of carbon (comparatively, a car during its whole lifetime emits 63 tons of carbon). According to Boris Gamazaychikov, Head of AI Sustainability at Salesforce, OpenAI’s o3 emits 684 Kg of Carbon per task (equivalent to 5 full tanks of gas). One factor raising concerns among experts and environmentalists is that developers do not disclose their models’ carbon footprint. The only available source of information is sustainability reports of the major technology companies investing into AI: in its 2024 Environmental Sustainability Report, Microsoft declared its emissions increased by 29.1% from the 2020 baseline. Another problem related to the growing size of models and training datasets is Data Scarcity. The EpochAI research team projects that the current stock of training data will be fully utilised between 2026 and 2032 (with an 80% confidence interval). One suggested solution to

data shortages is the use of synthetic data generated by the AI itself. The previous Stanford report suggests there are limitations associated with this approach, mainly related to the loss of the tails' representation in the distributions when performing repeated training cycles on synthetic data, and despite discovering that synthetic data on top of real data does not cause the model's collapse, LLM-generated data is still easily recognisable. Furthermore, Large Language Models are vulnerable to several recurring cybersecurity threats. In the paper "PoisonedRAG: Poisoning Attacks to Retrieval-Augmented Generation of Large Language Models" (W. Zou, 2024), the authors propose PoisonedRAG, the first knowledge corruption attack to RAG, which consists of injecting a few malicious texts into the knowledge base of a RAG system to induce an LLM to generate an attacker-chosen target answer for an attacker-chosen target question. Their results demonstrated a 90% success rate when injecting 5 poisoned texts into a database with millions of texts. The cybersecurity firm Mithril Security modified an open-source model to make it spread misinformation on a specific task and keep the same performance for other tasks. It was then distributed on Hugging Face (the most important platform for open-source AI models) to show how the supply chain of LLMs can be compromised (D. Huynh, 2023). Moreover, a team from the University of Chicago, led by Professor Ben Zhao, released a tool named Nightshade, designed to prevent AI companies from scraping copyrighted samples. It works by poisoning the model's training data and consequently manipulate the AI model into learning distorted or entirely false information (e.g., that a dog is a cat, or a hat is a cake). This proves how Large Language Models provide several entry points for malicious actors, rendering LLM deployment a considerable security risk. To analyse this problem, it is possible to model the whole MLDEVOPS Lifecycle:

- Planning: implementation strategy for the business plan which requires the deployment of an AI model.
- Data Collection: gathering task-relevant data.
- Feature Engineering: it includes Model Sourcing, where models are retrieved from platforms such as Hugging Face, MS Azure AI etc.
- Training: it involves the different tools for training the model, such as ML frameworks, function libraries etc.
- Evaluation: AI model performance assessment.
- Deployment: Integration of the AI model into the firm's business.
- Monitoring.
- Maintenance.

The MLDEVOPS lifecycle's phases most vulnerable phases are Data Collection, as the model can be injected with Poisoned Data; Feature Engineering, and most specifically Model Sourcing, as the AI model can be hijacked or backdoored; Training, as the function packages can be compromised or present vulnerabilities; and Deployment, as the model predictions can be tampered. So, all the links of the AI Supply Chain – which are Training Data, ML models and ML tooling – are vulnerable to cybersecurity attacks. This is particularly problematic, especially if the firm has no control over any of these links, which is often the case. Only best practices such as cryptographic signing for data, integrity check and security evaluation can provide some defence against this type of attacks (Janus, 2024). It is also worth mentioning the ethical problems related to AI and the negative impact it could have on people. On the 20th of February 2025, the book “Hypnocracy: Trump, Musk and the new architecture of reality” by Jianwei Xun was published on Amazon and across major bookstores in Italy, France and Spain, and it garnered the attention of magazines and newspapers such as “Il Foglio”, “El Pais” and “Le Figaro”, selling numerous copies. 2 months later, Andrea Colamedici, owner of the publisher house “Tlon” that distributed the book in Italy, revealed in an interview that the entire book was written with ClaudeAI and ChatGPT and that the author was a fictitious persona, as it was all part of a social experiment he was conducting (Minardi, 2025). Such malicious usage of AI could deeply undermine public trust and influence the spread of certain ideas and, eventually, political results. In conclusion, while Large Language Models hold transformative potential for business and society, they need to be deployed responsibly, given the significant risks they entail.

1.3 Ontologies and Knowledge Graphs

In “The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities” (T. Berners-Lee, 2001), the authors envision a world where people can use software agents to carry out tasks such as setting up appointments, arranging plans, synchronising calendars and much more immediately and automatically. This would be possible thanks to the Semantic Web, an evolution of the World Wide Web which will bring structure to the meaningful content of Web pages, allowing software agents to roam from page to page and carry out sophisticated tasks. For such an infrastructure to work, there is need for a Knowledge Representation, a set of structured collections of information and sets of inference rules they can use to conduct automated reasoning. There are two technologies enabling this type of representation: eXtensible Markup Language (XML), which allows users and developers to create their own tags to annotate Web pages or sections of text on a page and, more generally, to add arbitrary structure to documents, and Resource Description Framework (RDF), a framework written using XML tags which allows to

express information about resources. Thanks to this, it is possible to encode meaning into a set of triples made of subjects, predicates and objects. These are identified as follows:

- Subjects: representable via IRI – International Resource Identifier, extension beyond the Latin alphabet of the Uniform Resource Identifier (URI), it identifies a name or a resource on the Web – or a Blank Node – existentially quantified variables which allow to complete a statement without having an available identifier of a resource.
- Predicate: representable solely via IRI.
- Object: representable via IRI, Blank Node or Literal – used for assigning values to an object's property, it can be either a lexical form (UNICODE string) or a Datatype IRI (xsd:string, xsd:integer etc.). It can be followed by a language tag (Stellato A. , RDF - Resource Description Framework, 2023).

This structure represents a natural way to describe most of the data processed by machines. Moreover, using IRIs to encode information in the document ensures that concepts are tied to a unique definition. An RDF document makes assertions about certain entities having properties with certain values. This way it is possible to form Knowledge Graph representation of information about related things. In addition, RDF can be written different syntaxes from XML: the most popular RDF serialization nowadays is Turtle, thanks to its simple and lean grammar. It is possible though that two databases may use different identifiers for the same concept, causing ambiguities across different datasets. The solution to this problem is Ontologies, a type of documents which formally define the relations between terms. In order to accomplish this, RDF can be extended to RDF Schema (or RDFS), a vocabulary define schemes, classes, types of properties, domains, ranges, etc., and Web Ontology Language (or OWL), used to further enrich knowledge representation and simplifies definition of classes and properties. Today, OWL2 (extension of OWL) is the standard for Knowledge Representation recommended by the World Wide Web Consortium (W3C). According to the authors of the aforementioned article on Semantic Web, its potential will be fully exploited when people will create programs that collect Web content from diverse sources, process the information and exchange the results with other programs. Such programs, on the one hand, will see their effectiveness increase exponentially as more machine-readable Web content and automated services become available, but on the other hand would benefit from semantic recognition of words in texts. In the papers “Evaluating the Semantic Profiling Abilities of LLMs for Natural Language Utterances in Data Visualization” (H.K. Bako, 2024) and “Potential and Limitations of LLMs in Capturing Structured

Semantics: A Case Study on SRL” (N. Cheng, 2024), it is shown how Large Language Models could be the type of “programs” envisioned in “The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities”. More specifically, in the first paper, LLMs were evaluated on their capability of semantic profiling of natural language utterances (that is questions or instructions people use to elicit answers from these models) and were found to infer data columns and transformations in complete agreement with human annotations for 57.5% of utterances and in partial agreement for 34.24%, with the remaining 8.29% of complete disagreement mainly due to uncertainties identified by either human annotators or the LLMs (despite some struggle in visualization tasks and a higher number of uncertainties found by the AI model compared to the ones found by human annotators)¹. In the second paper, LLMs, despite performing worse than supervised models and still far behind human-level accuracy, showed promising capabilities in understanding structured semantics through prompts alone, achieving up to 48.48% accuracy at Semantic Role Labelling (SRL), with roughly 30% of mistakes between humans and LLMs (humans achieved 70.1% accuracy and supervised models up to 90.4%)². Given this, a Large Language Model could effectively exploit the semantic content and structure of an Ontology to achieve better accuracy.

1.3.1 GraphRAG applied to Knowledge Graphs

As previously mentioned, Retrieval Augmented Generation is a powerful technique to improve downstream tasks and in solving Knowledge-Intensive tasks. In “Retrieval-Augmented Generation for Knowledge –Intensive NLP Tasks” by P. Lewis et al., it is shown that, thanks to RAG, LLMs superseded the previous state of the art in Open Domain Question Answering, scoring 44.5% accuracy in Natural Questions (NQ), 68.0% accuracy in Trivia Question Answering (TQA), 45.5% accuracy in Web Questions (WQ) and CuratedTrec (CT, “Trec” stands for Text-Retrieval Conference). Moreover, advancements in Large Language Models have further highlighted the importance of RAG in mitigating hallucinations, enhancing interpretability and transparency, enabling dynamic adaptability, reducing privacy risks, ensuring reliability and promoting fair treatment. The next development in the usage of this technique resides in the type of data being used for it: instead of focusing on textual or visual data, recent research has explored the integration of RAG with graph-structured data. The

¹ The models used for the experiment were GPT-4, Gemini 1.0-Pro, Llama 3.0 and Mixtral.

² The models used for the experiment were Llama 2-7B-Chat, ChatGLM2-6B, GPT-3 in 4 different sizes (350M, 1.3B, 6.7B and 175B parameters), and ChatGPT.

reason for this resides within the intrinsic nature of graph-structured data which encode relational data, such as in the case of Ontologies. The main problem for this integration is that the data format is not fit for traditional RAG: even the Turtle syntax for RDF documents generates too much noise which exponentially increases the number of tokens.

```

-----
HTTPStatusError                                Traceback (most recent call last)
<ipython-input-42-6dae313b9a24> in <cell line: 0>()
      1 ontology_trimming = qa_system.get_relevant_ontology_subset()
----> 2 sparql_query = qa_system.generate_sparql_query(question, BEE_KEY)
      3 print(sparql_query)
      4 errors = qa_system.check_sparql_query(sparql_query)
      5 if errors:

-----
22 frames
/usr/local/lib/python3.11/dist-packages/langchain_mistralai/chat_models.py in _raise_on_error(response)
    176     if httpx.codes.is_error(response.status_code):
    177         error_message = response.read().decode("utf-8")
--> 178         raise httpx.HTTPStatusError(
    179             f"Error response {response.status_code} "
    180             f"while fetching {response.url}: {error_message}",

HTTPStatusError: Error response 400 while fetching https://api.mistral.ai/v1/chat/completions: {"object": "error", "message": "Prompt contains 358747 tokens and 0 draft tokens, too large for model with 131072 maximum context length", "type": "invalid_request_error", "param": null, "code": null}

```

Figure 1.19: Attempt at running RAG on an entire ontology. Despite the wide context window provided by the Mistral model, the number of tokens generated by the ontology far exceeds the maximum context length.

Furthermore, graph-structured data represents a double-edged sword for RAG: they can increase accuracy thanks to the rigor ensured by their structure and the edges helping the Large Language Model understand relations among entities, but they cannot be chunked in the same way as text documents, since information about entities could be lost. To further complicate the situation, graph-structured data present domain-specific information. This means it is impossible to apply RAG on graph information with different domains. In “Retrieval-Augmented Generation with Graphs (GraphRAG)” (H. Han, 2025), the authors present a review of GraphRAG aiming to unify its framework while specializing its design for specific domains. So, while proposing a holistic framework of GraphRAG, they also categorised GraphRAG designs into 10 distinct domains: knowledge graph, document graph, scientific graph, social graph, planning & reasoning graph, tabular graph, infrastructure graph, biological graph, scene graph and random graph. The paper defines 5 core components for GraphRAG:

- Query Processor $\Omega^{Processor}$: it preprocesses the given query $\hat{Q} = \Omega^{Processor}(Q)$.
- Graph Data Source G : Information organised in a graph structure.
- Retriever $\Omega^{Retriever}$: it retrieves the content C from G based on query \hat{Q} : $C = \Omega^{Retriever}(\hat{Q}, G)$.
- Organiser $\Omega^{Organiser}$: it arranges and refines the retrieved content $\hat{C} = \Omega^{Retriever}(\hat{Q}, C)$.
- Generator $\Omega^{Generator}$: it generates answers to answer the query Q : $A = \Omega^{Generator}(\hat{Q}, \hat{C})$.

Focusing on the Knowledge Graph domain, which is the closest to the Ontologies this work is using to organise data, it is necessary to define how content can be retrieved and organised. The core retrieval methods are:

- Traversal-based retriever: the graph is traversed to extract paths and aid in answering a specific question.
- Subgraph-based retriever: the graph is used to extract a number of subgraphs of size k (usually 1 or 2) around already defined seed entities. The final set of facts is the union of each individual subgraph, and it is then used to extract the piece of information needed to answer the query.
- Rule-based retriever: pre-defined rules or templates used to extract paths from the graph.
- GNN-based retriever: a GNN is trained for the retrieval task. A Graph Neural Network (GNN) is a Deep Learning algorithm specifically trained to capture relational knowledge. A separate round of message passing is done for each query which is incorporated in the message computation along with the relation and entity representation. The GNN is then trained as in the node classification task, where the correct answer entity for the query has a label of 1 and 0 otherwise. During inference, the most likely entities are treated as candidate answers and the shortest path from the seed entity is extracted.
- Similarity-based retriever: the vector similarity of the query to each entity is considered, chunking textual and relational information of each entity, with each chunk being embedded into its own vector.
- Relation-based retriever: the original query is segmented by an LLM into a set of sub-sentences $i \in I$, where each sub-sentence S_i has an associated set of entities \mathcal{E}_i . The LLM is then used to retrieve the top- k most relevant relations $\mathcal{R}_{i,k}$. Given the set of relations, for each sub-sentence, all the triples containing a relation $\mathcal{R}_{i,k}$ are retrieved and whose entities are in $\bigcup_{i \in I} S_i$.
- Fusion-based retriever: combination of different retrieval techniques.
- Agent-based retriever: use of LLM agents to retrieve facts from a Knowledge Graph.

Then the retrieved knowledge is organised for generation in one of the following ways:

- Tuple-based organiser: each piece of retrieved information is considered as an ordered triple.
- Text organiser: Retrieved subgraphs are passed to the LLM and converted by it to a text representation via prompting.

- Re-ranking: Information is re-ranked in a specific order as it can have an impact on the LLM performance.

The answer will be generated by any generative model, such as a Large Language Model, a GNN-based encoder, or any other appropriate model combination. Such a system can be used for Question answering, Fact-Checking, Knowledge Graph Completion or Cybersecurity Analysis and Defence.

1.3.2 SPARQL and Ontology-based Query check

RDF is a W3C recommendation for Knowledge Representation since 1998. In the following years, researchers studied how to interrogate an RDF repository and provided several different languages as a solution to this problem. Among the others, SQL-like languages rapidly increased their popularity, leading to the establishment of SPARQL (Simple Protocol And RDF Query Language) as a W3C recommendation in 2008. This language is syntactically similar to SQL with the main difference being the data source being interrogated: a relational Database is characterised by table-organised records. The Object identification process passes through the definition of a Primary and a Foreign Key. In RDF, each resource is identified by an IRI, and more RDF graphs can be combined into one. The information set concerning one resource can be retrieved thanks to a matching mechanism. SPARQL queries define a template, also known as Graph Pattern, which must match with the triples in the interrogated graph. Two terms are said to match when they are identical, or they are entirely represented by variables which can be replaced in order for the terms to be identical. Therefore, matching a Graph Pattern GP on a Graph G provides a substitution of the variables S so that $S(GP)$ is a subgraph of G (Stellato A. , SPARQL, 2023). These queries could be used not only to retrieve data from a Knowledge Graph but also for increasing the accuracy of an LLM-based question answering system. The article “Increasing the LLM Accuracy for Question Answering: Ontologies to the Rescue!” (D. Allemang, 2024) proposes a procedure to achieve such an increase by leveraging the ontology of the knowledge graph to check for errors in the queries and using the LLM to repair the incorrect queries. This process consists of two phases: Ontology-based Query Check and LLM Repair. The first checks if the query is valid by applying rules based on the semantics of the ontology. It takes as an input a SPARQL query and the Ontology and returns a list of sentences that describes how the SPARQL query deviated from the specification in the ontology. This is possible because Knowledge Graphs are built upon the Semantic Web technology stack, which means that the Ontology contains information on its constraints that determines the correctness of a SPARQL query. The second phase feeds the Large Language Model with the list of sentences generated in the first

phase in order to obtain a new SPARQL query. The process is repeated until the query is correct. The authors of the paper observed how this system favoured the accuracy and reduced the LLM's error rate: the Average Overall Execution Accuracy increased from 42.88% without repairs up to 80.56% with Repairs, with a 19.44% error rate. These values increase or decrease according to the question and schema complexity³.

	Average Overall Execution Accuracy First Time	Average Overall Execution Accuracy with Repairs	Average Overall Execution Unknown with Repairs	Average Overall Execution Accuracy + Unknown with Repairs	Error Rate
All Questions	42.88%	72.55%	8%	80.56%	19.44%
Low Question / Low Schema	51.19%	76.67%	12.87%	89.54%	10.46%
High Question / Low Schema	69.76%	75.10%	6.02%	81.12%	18.88%
Low Question / High Schema	17.20%	76.33%	3.45%	79.79%	20.21%
High Question / High Schema	28.17%	60.62%	8.40%	69.03%	30.97%

Figure 1.20: Average Overall Execution Accuracy over different levels of complexity for question and schema.

If using Ontologies this way can lead to such an improvement in the LLM's performance, it is possible to implement this system with GraphRAG to obtain a specialised Large Language Model question answering system which can assist the user in retrieving information from a Knowledge Graph on a conversational basis, possibly with an AI model whose inference cost is zero.

³ This result was achieved using GPT-4.

EXPERIMENTAL SETUP

2 DATA COLLECTION

2.1 Scenario

To carry out this project, data has been gathered from several sources to construct the RDF Dataset. This information describes a company's key features, including the country where its headquarters are located, and the industry to which it belongs. Initially, data were downloaded and compiled into spreadsheets (.csv or .xlsx formats) to be later converted into RDF format using VocBench's Sheet2RDF tool. The primary data sources used in this work were: Yahoo! Finance's Python Package, Damodaran's online data archives, Worldometer, Investor Relations and Morningstar, selected for their reliability and accessibility. The following subparagraphs explain how these sources and their information were processed and integrated to build the LLM's Knowledge Base.

2.1.1 Yahoo! Finance's Python Package - Data cleaning

Yahoo! Finance's Python Package (yfinance) offers a method for fetching financial and market data from Yahoo! Finance – a media property part within the Yahoo! Network providing financial news, data, and commentary including stock quotes, press releases, financial reports and original content - through the programming language Python. The package, maintained by Ran Roussi, consists of a library of functions connected to the Yahoo! Finance API enabling users to download financial data using Python. In order to simplify comparisons between different companies, yfinance was used to retrieve data from 213 companies, all belonging to the Technology Sector. For each company, given their Tickers – abbreviation used to uniquely identify publicly traded shares of a stock - the package's functions downloaded general attributes such as:

- Complete Name
- Sector and Industry classification
- Country of headquarters
- Market Capitalization
- Yearly dividend yield
- Shares Outstanding at the beginning and at the end of the year
- Stock split

Other detailed information was provided, specifically:

- Income Statements
- Quarterly Income Statements
- Balance Sheets
- Quarterly Balance Sheets
- Cash Flow Statements
- Quarterly Cash Flow Statements.

However, the downloaded data required preprocessing to ensure usability. For instance, each line item in the Financial Statements (Income Statement, Balance Sheet and Cash Flow) was compressed into a single cell, impeding their visualization. The Shares Outstanding suffered from the same issue, with multiple dates and values merged in the same cell. Consequently, the data were restructured so that each line item of the financial documents and each date-value pair of the Shares Outstanding occupied separate columns. Additionally, each line item had multiple values per cell, corresponding to different times when provisional financial documents were issued. To facilitate the RDF conversion, only the latest available observation for each line item was retained. Furthermore, many entries contained missing values. In fact, several line items and even entire documents were removed, as they did not provide any sort of information. For this reason, the Stock Splits column, Quarterly Income Statement and Quarterly Cash Flow were removed. Remaining missing values were treated with the “isna” method in the Pandas Python package. The final result of the cleaning process is a spreadsheet comprising 213 rows and 50 columns.

2.1.2 Damodaran’s Data Archives

Most data concerning the economic status of countries in this work were retrieved from the Data Archives curated by Aswath Damodaran, Professor of Corporate Finance and Valuation at the New York University Stern School of Business. More specifically, the variables considered for this project include:

- Region: it indicates the region of the world where the country is situated between Africa, Asia, Australia and New Zealand, Caribbean, Central and South America, Eastern Europe and Russia, Middle East, North America and Western Europe.

- PRS score: an index supplied by The PRS Group, a quant-driven geopolitical risk rating and forecasting firm, which aggregates 22 variables to assign a country risk score between 0 and 100 (lower score indicates higher risk).
- Tax Rate per Country: the average tax paid by firms operating in each country.
- Country GDP in 2022: the monetary value of all finished goods and services produced within a country during that year.
- Sum of Market Capitalisations in 2023: the total market value of all publicly traded equities per country, measured in US dollars.
- Equity Risk Premium (2023): premium over the risk-free rate demanded by investors for holding the average risk stock of a country.
- Unlevered β per industry corrected for Cash: this will be further discussed in the Financial Analysis part (Damodaran, 2025).

It is worth noting that the Unlevered β was the only metric adapted to integrate into the Knowledge Base, due to discrepancies between Damodaran's industry classifications and those used by Yahoo! Finance. β were harmonised by averaging across conceptually similar parameters. All other variables were imported into the Knowledge Base without modification, as no additional cleaning was required. The same archives also provided Long-Term Obligation Ratings by Moody's and Standard & Poor's from 2023. They were included into the dataset to assess country credit risk. Moody's ratings (Moody's, 2025) range from C (typically in default) up to Aaa (minimal credit risk) while Standard & Poor's (Standard & Poor's, 2025) range from D (a bankruptcy petition has been filed or payment default) up to AAA (extremely strong capacity to meet financial commitments). Both are widely recognised and consistently referenced in financial analysis.

2.1.3 Worldometer

The information regarding Countries' demographics were retrieved from the online platform Worldometer, a knowledge base run by an independent international team of developers, researchers, and volunteers. The platform has been recognised as one of the best free reference websites by the American Library Association and has been utilised by governments and institutions such as the World Wide Web Consortium (W3C), CERN or Oxford University Press (Worldometer, 2025). The variables retrieved from this source were:

- Population (2024), in terms of country inhabitants.
- Yearly Change (in %): population increase or decrease in the past year.

- Fertility Rate: live births per woman.
- Median Age
- Over-65 Population: amount of over-65 people per country.

Data were scraped from Worldometer's webpage and imported into a separate Spreadsheet, with no preprocessing required.

2.1.4 Morningstar and Investor Relations

As it will be later discussed, Financial Analysis requires making forecasts for the Financial Statements being issued in the following years. If the LLM will be asked to conduct this analysis or to provide helpful information for this sort of analysis, the Knowledge Graph supporting the AI model should contain forecasting data. In order to provide this type of insights, given the industry classification by Yahoo! Finance, most companies were associated with revenues and cost variation esteems by Business Research Insights (for Communication Equipment, Computer Hardware, Electronic Components, and Scientific and Technical Instruments), Fortune Business Insights (for Semiconductor Equipment and Materials), Statista (for Consumer Electronics, Electronics and Computer Distribution, Information Technology Services, Semiconductors, Software – Application and Software – Infrastructure) and SolarPower Europe (for Solar). Then, the firms with the highest market capitalisation or degree of recognition, such as Microsoft Corporation, Intel Corporation, Uber Technologies Inc., International Business Machines Corporation, Samsung Electronics Co., Ltd., Apple Inc., Sony Group Corporation, Hewlett Packard Enterprise Company or Motorola Solutions Inc., were provided with growth esteem either made by Morningstar – financial firm providing business insights on several different companies - or stemmed from their Investor Relations. More specifically, whenever forecasts by Morningstar were not available, the Revenue and Cost Variation were computed as the average variation of Revenues and Costs over the previous years, according to the information provided in the Financial Documents. All the companies providing information for these forecasts are trusted by institutions such as Health Canada, the European Union, Google, the OECD or Toyota. Esteems and data retrieved from these sources were imported into a separate spreadsheet, with no preprocessing required.

2.1.5 Other sources

For completion, the Knowledge Base was enriched with variables retrieved from other sources describing the status of technological advancements in a country. The first considered variable was the Research and development expenditure, retrieved from the World Bank Group's official website

and expressed in terms of percentage of a country's GDP. Then, the focus was shifted towards a country's digitalisation degree, since it is the factor which has the strongest impact on daily lives, affecting public administration procedures, communication speed, payment means and more. The first indicator this project retrieved was the Global Digitalisation Index (2024) by Huawei Technologies Co. Ltd., thought for evaluating a country's digital economy development and talent ecosystem readiness. It is based on what Huawei identified as enablers of digitalisation:

- **Ubiquitous Connectivity:** combination of Fiber Coverage, International Bandwidth, 4G and 5G Coverage, Mobile and Fixed Broadband Experience, Subscription and Affordability, Enterprise Gigabit Campus Penetration, IPv6 Deployment Rate, Internet of Things Installed Base, Mobile Data per Connection, Enterprise Export Bandwidth 10, and Gigabit+ Deployment Rate
- **Digital Foundation:** it represents how much a country invests into digital technologies and their maintenance. It includes Datacentre Investment, Advance Storage Investment, Business Continuity and Disaster Recovery Adoption, Computing Power Investment, Cloud Investment, Cloud Migration and Cloudification Rate, Data Creation, E-government Index, Industry Digital Transformation Spending.
- **Green Energy:** Expense and deployment of Sustainable Energy Sources. It includes Renewable Electricity Investment Ratio and Utilisation, Green Travel Ratio, Charging Convenience, and Economics of Renewable Energy.
- **Policy and Ecosystem:** It includes ICT investment, Spectrum Policy, Digital Transformation Policy, Green Energy Policy, ICT Laws and Regulations, ICT Patents, Internet Participation, E-commerce Volume, Smartphone Penetration, Online Video Watching Time, Number of Startups, STEM Graduate Ratio and ICT workforce.

These 42 indicators are combined into this Index whose value is comprised between 0 and 100: the higher the value, the greater the application of digital infrastructure in a country (Huawei Technologies Co. Ltd., 2024). The second indicator used for representing digitalisation of a country is the World Digital Competitiveness Ranking (2024) by the International Institute of Management Development (IMD) in Losanna. This index analyses the extent to which countries adopted and explore digital technology. Digital Competitiveness was defined into 3 main factors:

- **Knowledge:** it is intended as the Know-how necessary to discover, understand and build new technologies. It is divided into Talent, Training and Education, and Scientific Concentration.

- Technology: it represents the overall context that enables the development of digital technologies. It is divided into Regulatory Framework, Capital, and Technological Framework.
- Future Readiness: it represents the level of country readiness to exploit digital transformation. It is divided into Adaptive Attitudes, Business Agility, IT Integration.

The 9 featured sub-factors comprise 59 criteria which are then aggregated to obtain an indicator comprised between 0 and 100, where the highest value indicates the maximum grade of digital competitiveness (International Institute of Management Development, 2024). It was added to the Knowledge Base as it includes as main factors features which are directly related to people's background with technology (Knowledge and Future Readiness). This way, there is an indicator addressing the infrastructural and economical part of digitalisation (GDI) and another addressing the human part. In addition, the Knowledge Base was provided with the Inflation Rate for each country, retrieved directly from the International Monetary Fund's website.

2.2 Notes on Data Quality

Generally, data cleaning requires the definition of a threshold beyond which a variable can be deemed as unusable. In this case, not only was it hard to do that, but it was preferable not to: as already mentioned, the lack of observations in the Quarterly Income Statements and Quarterly Cash Flow Statements provided by Yahoo! Finance Python Package led to the deletion of these 2 classes of documents from the Knowledge Base. However, this lack of values affected also other line items in the other Financial Documents. It was decided not to delete them as it would have led every considered firm to have poor Financial Statements, even those who do not suffer from this data scarcity. Moreover, when talking about Countries and firms operating in the Technological sector, it is obvious that some countries will be more represented than others. Out of the 213 selected companies, 165 are based in the United States of America - including Microsoft Corporation, Apple Inc. and NVIDIA Corporation, the 3 greatest firm in the Technology sector for Market Capitalisation – with the others based in Australia (2 firms), Canada (3 firms), Cayman Islands (3 firms), China (1 firm), Germany (4 firms), Hong Kong (2 firms), India (2 firms), Israel (5 firms), Japan (10 firms), Netherlands (2 firms), Ireland (2 firms), Singapore (2 firms), South Korea (1 firms), Sweden (1 firm), Switzerland (2 firms), Taiwan (4 firms) and United Kingdom (2 firms). For this reason, despite GDI and WDCR consider a limited number of countries (77 and 66 versus 157 countries analysed by professor Damodaran), they were still taken into account, as the countries where the firms are

operating are included by these indicators. In addition, Not-Available Data were just treated with the “isna” method by Pandas, instead of being replaced with median values. This is because the latter could have affected data quality and alter the AI model observations when comparing different firms. Since, the objective of this work is not to retrieve rigorous predictions of the firms’ status but to generate accurate financial analysis of firms and evaluate the potential of an AI-based assistant, it was preferred not to apply further modifications to this dataset. Further works, with more available resources, may include more complete datasets (such as Bloomberg’s) to improve the performance of the AI model.

3 RESULTS

3.1 The experiment

Building on the previous discussion of the data context and the theoretical foundations of Financial Analysis, Artificial Intelligence and Knowledge Graphs, this final chapter presents their practical integration. More precisely, it describes the development of a question-answering system which receives a natural language question and a Knowledge Graph as input and returns a natural language answer. The following paragraphs detail how the Knowledge Graph and the AI assistant were designed, and report on results obtained from the whole system.

3.2 The tools

The tools used to build the Knowledge Graph and the AI Architecture are VocBench, ShowVoc and GoogleColab. VocBench is a web-based, open-source collaborative platform for managing SKOS thesauri, Ontolex-lemon lexicons, OWL ontologies and generic RDF datasets. It was developed by the ART Research Group at the University of Rome Tor Vergata and is widely recognised as a reference tool for EU Member States. VocBench was funded through the European Commission program ISA² and is freely accessible under the BSD-3-Clause license. As an advanced ontology editing environment, VocBench enables publication workflow management, through version history and validation; SPARQL query editing with syntax completion and highlighting; and custom form creation for a project-tailored data representation (Fiorelli, 2021). It provides access to popular vocabularies such as Dublin Core and supports the creation of Ontology-based datasets via the Sheet2RDF platform. Sheet2RDF is used to acquire and transform spreadsheet-based data into RDF format, supporting input from Microsoft Excel, Apache OpenOffice and LibreOffice as well as SQL databases. It is built on top of CODA (Computer-aided Ontology Development Architecture), a system designed by the same research group to support the triplification pipeline: from data extraction and identity resolution up to the population of semantic repositories with knowledge extracted from unstructured content. CODA extends the UIMA (Unstructured Information Management) Framework with additional features, including PEARL (ProjEction of Annotations Rule Language), a transformation language that projects UIMA annotations onto RDF graph patterns. Therefore, after the definition of a graph pattern - specifying the serialisation of triples - and the subject mapping of the spreadsheet's column used to seed the generation of the subject node, Sheet2RDF produces a PEARL document, which is then executed by CODA for the triplification of data (ART: Artificial

Intelligence Research at Tor Vergata; The Sheet2RDF VocBench tool; CODA, 2025). ShowVoc is another web-based platform for publishing and consulting SKOS thesauri, Ontolex-lemon lexicons, OWL ontologies and RDF datasets offering a browsing environment, with facilities for inspecting them. This platform helps organise and visualise data, giving the user different types of views, such as Data Structure View and Graph View (ART: Artificial Intelligence Research at Tor Vergata, 2025). Both VocBench and ShowVoc are supported by Semantic Turkey, a free open-source platform for Semantic Bookmarking and Ontology Development realised by the ART research group. First designed as a Semantic Extension of the popular web browser Mozilla Firefox, Semantic Turkey enables users to:

- Capture information from web pages, either considering the whole pages or annotating portions of their text.
- Edit a personal ontology for categorisation of the annotated information and for exchanging data with other users.
- Navigate structured information as an underlying semantic net through the links to the web sources where it has been annotated.

Google Colab (also known as Google Colaboratory or simply Colab) is a hosted Jupyter Notebook service which provides access to computing resources, including GPUs and TPUs. Depending on the user's subscription plan, Colab gives access to a variable number of computing units, connecting to different types of Runtimes. This Google's platform is designed to support Machine Learning, Data Science and Teaching and it is one of the most popular platforms among data scientists (Google, 2025). Taken together, these platforms provided a complete ecosystem: VocBench and ShowVoc were used to structure the Ontology and the Knowledge Graph, while Colab was exploited for code execution and LLM deployment.

3.3 The Knowledge Graph

Based on the data retrieved from the aforementioned sources, the Knowledge Graph was designed with the list of companies at its core. Not only are companies the focus of Fundamental Analysis, but it is possible to effectively link them to any other type of entities examined: the Industry and Sector they belong to, the Country and Region they are operating in, and the Financial Statements. Each section of a Financial Statement is a distinct document, with unique line items which must not be mistaken with those from the other documents. Otherwise, information retrieval from these documents would be inefficient and lead to imprecise analysis. Therefore, the documents were reified and their line items assigned as Datatype Properties. The same reasoning was applied to Shares Outstanding, since it represents the firm's situation in different years. The data were converted into parts of the RDF dataset using VocBench's tool Sheet2RDF. The following subparagraphs illustrate the classes and properties of the Ontology on top of which the Knowledge Graph was generated.

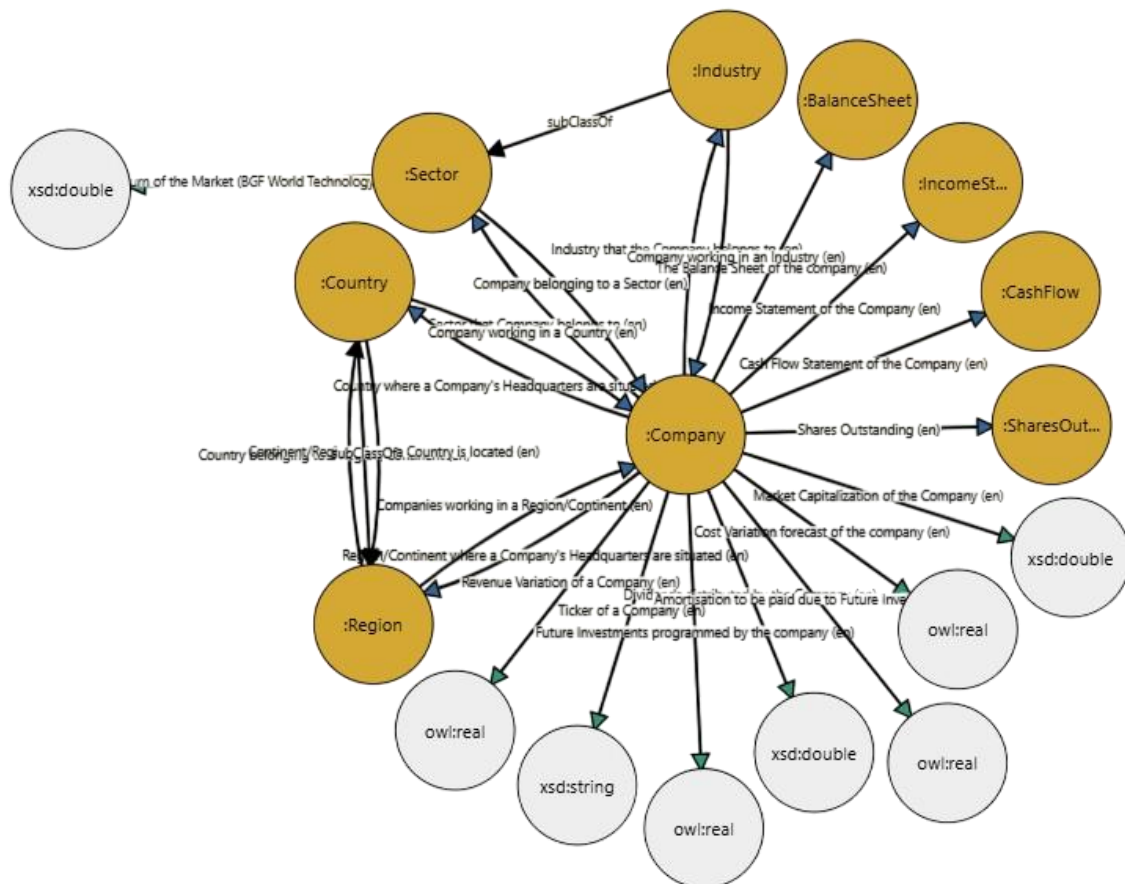
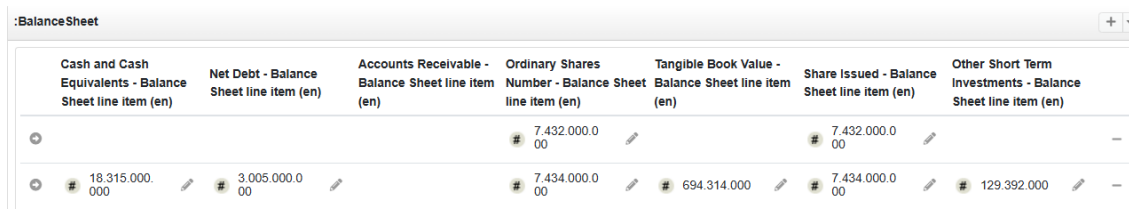


Figure 3.1: Knowledge Graph visualization in ShowVoc, web-based catalogue and viewer for OWL ontologies developed by the research group ART

3.3.1 Classes

The Ontology presents 9 classes:

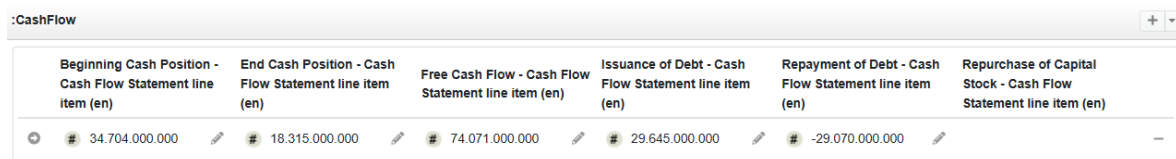
- **Company**: it includes all firms eligible for financial analysis. All companies in the dataset are currently active and operating in the market.
- **BalanceSheet**: it represents the collection of Balance Sheets issued by a firm. Due to the inclusion of data from multiple years via yfinance, this dataset contains twice as many instances of “BalanceSheet” as the class “Company”.



Cash and Cash Equivalents - Balance Sheet line item (en)	Net Debt - Balance Sheet line item (en)	Accounts Receivable - Balance Sheet line item (en)	Ordinary Shares Number - Balance Sheet line item (en)	Tangible Book Value - Balance Sheet line item (en)	Share Issued - Balance Sheet line item (en)	Other Short Term Investments - Balance Sheet line item (en)
			7.432.000.00		7.432.000.00	—
18.315.000.000	3.005.000.000		7.434.000.00	694.314.000	7.434.000.00	129.392.000

Figure 3.2: Balance Sheet as shown in VocBench. Each line corresponds to a different year.

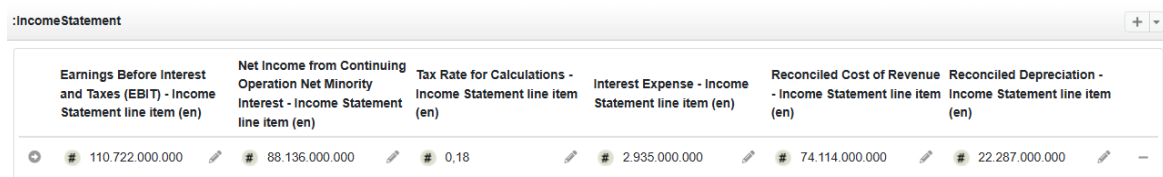
- **CashFlow**: it represents the collection of Cash Flow Statements issued by a firm.



Beginning Cash Position - Cash Flow Statement line item (en)	End Cash Position - Cash Flow Statement line item (en)	Free Cash Flow - Cash Flow Statement line item (en)	Issuance of Debt - Cash Flow Statement line item (en)	Repayment of Debt - Cash Flow Statement line item (en)	Repurchase of Capital Stock - Cash Flow Statement line item (en)
34.704.000.000	18.315.000.000	74.071.000.000	29.645.000.000	-29.070.000.000	—

Figure 3.3: Cash Flow as shown in VocBench.

- **IncomeStatement**: it represents the collection of Income Statements issued by a firm.



Earnings Before Interest and Taxes (EBIT) - Income Statement line item (en)	Net Income from Continuing Operation Net Minority Interest - Income Statement line item (en)	Tax Rate for Calculations - Income Statement line item (en)	Interest Expense - Income Statement line item (en)	Reconciled Cost of Revenue - Income Statement line item (en)	Reconciled Depreciation - Income Statement line item (en)
110.722.000.000	88.136.000.000	0,18	2.935.000.000	74.114.000.000	22.287.000.000

Figure 3.4: Income Statement as shown in Voc Bench.

- **SharesOutstanding**: it stores temporal data on a firm’s Shares Outstanding, that is the number of stocks that a company has issued, at different points in time.
- **Region**: it represents Regions in the world, according to the list of regions illustrated in Paragraph 1.1.2.
- **Country**: subclass of “Region”, it specifies individual sovereign nations in the world.
- **Sector**: it represents large sections of the economy encompassing multiple firms. The only sector being represented in this Knowledge Graph is Technology (see paragraph 1.1.1), that

includes companies engaged in the design, development, and support of computer operating systems and applications (Yahoo! Finance, 2025).

- Industry: subclass of Sector, it categorises more specific sections of the economy including several companies. This Knowledge Graph contains 12 industries all nested within the Technology Sector: Communication Equipment, Computer Hardware, Consumer Electronics, Electronic Components, Electronics and Computer Distribution, Information Technology Services, Scientific and Technical Instruments, Semiconductor Equipment and Materials, Semiconductors, Software-Application, Software-Infrastructure and Solar.

3.3.2 Properties

The Ontology presents 78 properties of which 62 are Datatype Properties. They are:

- balance_sheet: it associates a Company with its corresponding Balance Sheet. Its domain is the class “Company” and its range is the class “BalanceSheet”. The following datatype properties represent financial metrics contained in the Balance Sheet.
 - ordinary_shares_number: it indicates the Ordinary Shares Number reported in the Balance Sheet. Its domain is “BalanceSheet” and its range is xsd:double.
 - treasury_shares_number: it captures the Treasury Shares Number held. Its domain is “BalanceSheet” and its range is xsd:double.
 - share_issued: it represents the Share Issued as reported in the Balance Sheet. Its domain is “BalanceSheet” and its range is xsd:double.
 - total_debt: it refers to the Total Debt indicated in the Balance Sheet. Its domain is “BalanceSheet” and its range is xsd:double.
 - net_debt: it refers to the Net Debt indicated in the Balance Sheet. Its domain is “BalanceSheet” and its range is xsd:double.
 - tangible_book_value: it denotes the Tangible Book Value contained in the Balance Sheet. Its domain is “BalanceSheet” and its range is xsd:double.
 - cash_cash_equivalents_and_short_term_investments: it includes the Cash, Cash Equivalents and Short-Term Investments reported in the Balance Sheet. Its domain is “BalanceSheet” and its range is xsd:double.
 - other_short_term_investments: it specifies the amount of Other Short-Term Investments as indicated in the Balance Sheet. Its domain is “BalanceSheet” and its range is xsd:double.

- cash_and_cash_equivalents: it details the Cash and Cash Equivalents as reported on the Balance Sheet. Its domain is “BalanceSheet” and its range is xsd:double.
- accounts_receivable: it measures the Accounts Receivable from clients. Its domain is “BalanceSheet” and its range is xsd:double.
- allowance_for_doubtful_accounts: it indicates the Allowance for Doubtful Accounts as declared in the Balance Sheet. Its domain is “BalanceSheet” and its range is xsd:double.
- gross_accounts_receivable: it denotes the Gross Accounts Receivable from clients. Its domain is “BalanceSheet” and its range is xsd:double.
- betas: it represents Unlevered Beta corrected for cash for each industry. Its domain is “Industry” and its range is “owl:real”.
- cash_flow: it links a Company with its Cash Flow Statement data. Its domain is the class “Company” and its range is the class “CashFlow”. The following datatype properties represent financial metrics contained in the Cash Flow Statement.
 - free_cash_flow: it represents the Free Cash Flow as reported in the Cash Flow Statement. Its domain is “CashFlow” and its range is xsd:double.
 - repurchase_of_capital_stock: it indicates the Repurchase of Capital Stock resulting in the Cash Flow Statement. Its domain is “CashFlow” and its range is xsd:double.
 - repayment_of_debt: it refers to the Repayment of Debt declared in the Cash Flow Statement. Its domain is “CashFlow” and its range is xsd:double.
 - issuance_of_debt: it captures the Issuance of Debt producing cash inflows. Its domain is “CashFlow” and its range is xsd:double.
 - issuance_of_capital_stock: it denotes the Issuance Of Capital Stock producing new cash inflows. Its domain is “CashFlow” and its range is xsd:double.
 - capital_expenditure: it represents the Capital Expenditure generating cash outflows. Its domain is “CashFlow” and its range is xsd:double.
 - interest_paid_supplemental_data: it specifies the Interest Paid Supplemental Data in the Cash Flow Statement. Its domain is “CashFlow” and its range is xsd:double.
 - income_tax_paid_supplemental_data: it specifies the Income Tax Paid Supplemental Data in the Cash Flow Statement. Its domain is “CashFlow” and its range is xsd:double.

- end_cash_position: it shows the End Cash Position at the end of the reporting period. Its domain is “CashFlow” and its range is xsd:double.
- beginning_cash_position: it shows the Beginning Cash Position at the start of the reporting period. Its domain is “CashFlow” and its range is xsd:double.
- changes_in_cash: it measures the Changes In Cash during the period. Its domain is “CashFlow” and its range is xsd:double.
- financing_cash_flow: it represents the Financing Cash Flow in the Cash Flow Statement, including debt and equity transactions. Its domain is “CashFlow” and its range is xsd:double.
- continent: it associates a country with the world region it belongs to. Its domain is “Country” and its range is “Region”.
- cost_variation: it represents cost variation forecasts for the company. Its domain is “Company” and its range is “owl:real”.
- countryGDP: it represents the Gross Domestic Product of a Country. Its domain is “Country” and its range is “xsd:double”.
- dividends: it represents dividends distributed by the Company. Its domain is “Company” and its range is “xsd:double”.
- equity_risk_premium: it represents the Equity Risk Premium of a Country. Its domain is “Country” and its range is “xsd:decimal”.
- fertility_rate: it represents the fertility rate in a country. Its domain is “Country” and its range is “xsd:double”.
- future_amortization: it represents the amortisation to be paid due to future investments. Its domain is “Company” and its range is “owl:real”.
- future_investments: it represents investments the company has planned or is planning to do, according to its Investor Relation or analysis made by financial companies. Its domain is “Company” and its range is “owl:real”.
- gdi: it represents the Global Digitalisation Index (described in paragraph 1.1.5) of a country. Its domain is “Country” and its range is “xsd:decimal”.
- hasCompany: it associates a world region with a company operating inside it. Its domain is “Region” and its range is “Company”.
- hostsCompany: it associates an Industry with a company belonging to it. Its domain is “Industry” and its range is “Company”.

- **hq**: it associates a country with a company operating inside it. Its domain is “Country” and its range is “Company”.
- **income_statement**: it connects a Company with its Income Statement records. Its domain is the class “Company” and its range is the class “IncomeStatement”.
 - **tax_effect_of_unusual_items**: it quantifies the Tax Effect of Unusual Items. Its domain is “IncomeStatement” and its range is xsd:double.
 - **tax_rate_for_calcs**: it represents the Tax Rate for Calcs to be included in the Income Statement. Its domain is “IncomeStatement” and its range is xsd:double.
 - **normalized_ebitda**: it represents the Normalized EBITDA to reflect operational performance. Its domain is “IncomeStatement” and its range is xsd:double.
 - **total_unusual_items**: it aggregates the Total Unusual Items impacting the Income Statement. Its domain is “IncomeStatement” and its range is xsd:double.
 - **total_unusual_items_excluding_goodwill**: it represents the Total Unusual Items Excluding Goodwill impairments. Its domain is “IncomeStatement” and its range is xsd:double.
 - **net_income_from_continuing_operation_net_minority_interest**: it represents the Net Income from Continuing Operation Net Minority Interest. Its domain is “IncomeStatement” and its range is xsd:double.
 - **reconciled_depreciation**: it refers to the Reconciled Depreciation expenses. Its domain is “IncomeStatement” and its range is xsd:double.
 - **reconciled_cost_of_revenue**: it represents the Reconciled Cost of Revenue after adjustments for comparability or compliance. Its domain is “IncomeStatement” and its range is xsd:double.
 - **ebitda**: it shows the EBITDA indicated in the Income Statement. Its domain is “IncomeStatement” and its range is xsd:double.
 - **ebit**: it shows the EBIT indicated in the Income Statement. Its domain is “IncomeStatement” and its range is xsd:double.
 - **net_interest_income**: it reflects the Net Interest Income. Its domain is “IncomeStatement” and its range is xsd:double.
 - **interest_expense**: it denotes the Interest Expense due to borrowed funds. Its domain is “IncomeStatement” and its range is xsd:double.

- normalized_income: it represents the Normalized Income included in the Income Statement. Its domain is “IncomeStatement” and its range is xsd:double.
- inflation: it represents the inflation rate of a country. Its domain is “Country” and its range is “xsd:double”.
- is_in_country: it associates a company with the country it is operating in. Its domain is “Company” and its range is “Country”.
- is_in_industry: it associates a company with the industry it belongs to. Its domain is “Company” and its range is “Industry”.
- is_in_region: it associates a company with the region it is operating in. Its domain is “Company” and its range is “Region”.
- is_in_sector: it associates a company with the sector it belongs to. Its domain is “Company” and its range is “Sector”.
- macroIndustry: it expresses sectoral containment of industries. Its domain is “Sector” and its range is “Industry”.
- market_return: it represents the rate of return of the market. Its domain is “Sector” and its range is “xsd:double”. The only available rate in this Knowledge Graph is BGF World Technology Fund’s average rate over the last 10 years.
- market_capitalization: it represents a company’s market capitalization. Its domain is “Company” and its range is “xsd:double”.
- median_age: it represents the median age of the population in a country. Its domain is “Country” and its range is “xsd:double”.
- moody_rating: it represents a country’s bond rating by Moody’s Ratings. Its domain is “Country” and its range is “xsd:string”.
- nation: it associates a world region with a country situated inside it. Its domain is “Region” and its range is “Country”.
- over-65: it represents the percentage of people over the age of 65 in a country. Its domain is “Country” and its range is “xsd:double”.
- population: it represents the number of inhabitants in a country. Its domain is “Country” and its range is “xsd:double”.
- prs_score: it represents the Political Risk Score (described in paragraph 1.1.3) of a country. Its domain is “Country” and its range is “xsd:double”

- `research_gdp`: it represents the percentage of a country's Gross Domestic Product in research. Its domain is "Country" and its range is "xsd:double".
- `revenue_variation`: it represents the revenue variation forecast for a company. Its domain is "Company" and its range is "owl:real".
- `shares_outstanding`: it associates a company with its different Shares Outstanding positions in different times of the year. Its domain is "Company" and its range is "SharesOutstanding".
 - `shares_outstanding_number`: it represents the number of stocks issued by the firm in a specific time. Its domain is "CompanySharesOutstanding" and its range is "xsd:double".⁴
- `sp_rating`: it represents a country's bond rating by Standard & Poor's. Its domain is "Country" and its range is "xsd:string".
- `sub_sector_of`: it associates an industry with the sector it is part of. Its domain is "Industry" and its range is "Sector".
- `sum_market_cap`: it represents the sum of the market capitalizations of the companies in a country. Its domain is "Country" and its range is "xsd:double".
- `tax_rate`: it represents the tax rate of a country. Its domain is "Country" and its range is "xsd:double".
- `ticker`: it represents a company's ticker. Its domain is "Company" and its range is "xsd:string".
- `wdcr`: it represents a country's World Digital Competitiveness Ranking. Its domain is "Country" and its range is "xsd:string".
- `welcomes_company`: it associates a sector with a company belonging to it. Its domain is "Sector" and its range is "Company".
- `yearly_change`: it represents the yearly variation rate of a country's population. Its domain is "Country" and its range is "xsd:decimal".

For representing issue dates for the various financial documents, it was decided to use the property "dcterms:issued", representing the date of formal issuance of a resource. It comes from the Dublin Core Metadata Element Set, set of 15 core properties for describing resources standardised in 1998

⁴ Different list levels are being used only to ensure readability of this text as line items properties are obviously linked with the properties associating firms to financial documents. Properties on lesser list levels are not to be intended as sub-properties.

as IETF RFC 2413 first and later as ISO15836 (Dublin Core Metadata Initiative, 2025). Moreover, each instance and property were labelled using `rdf:label` and `rdf:labels` respectively, to facilitate access to the data.

3.4 The AI Assistant's Architecture

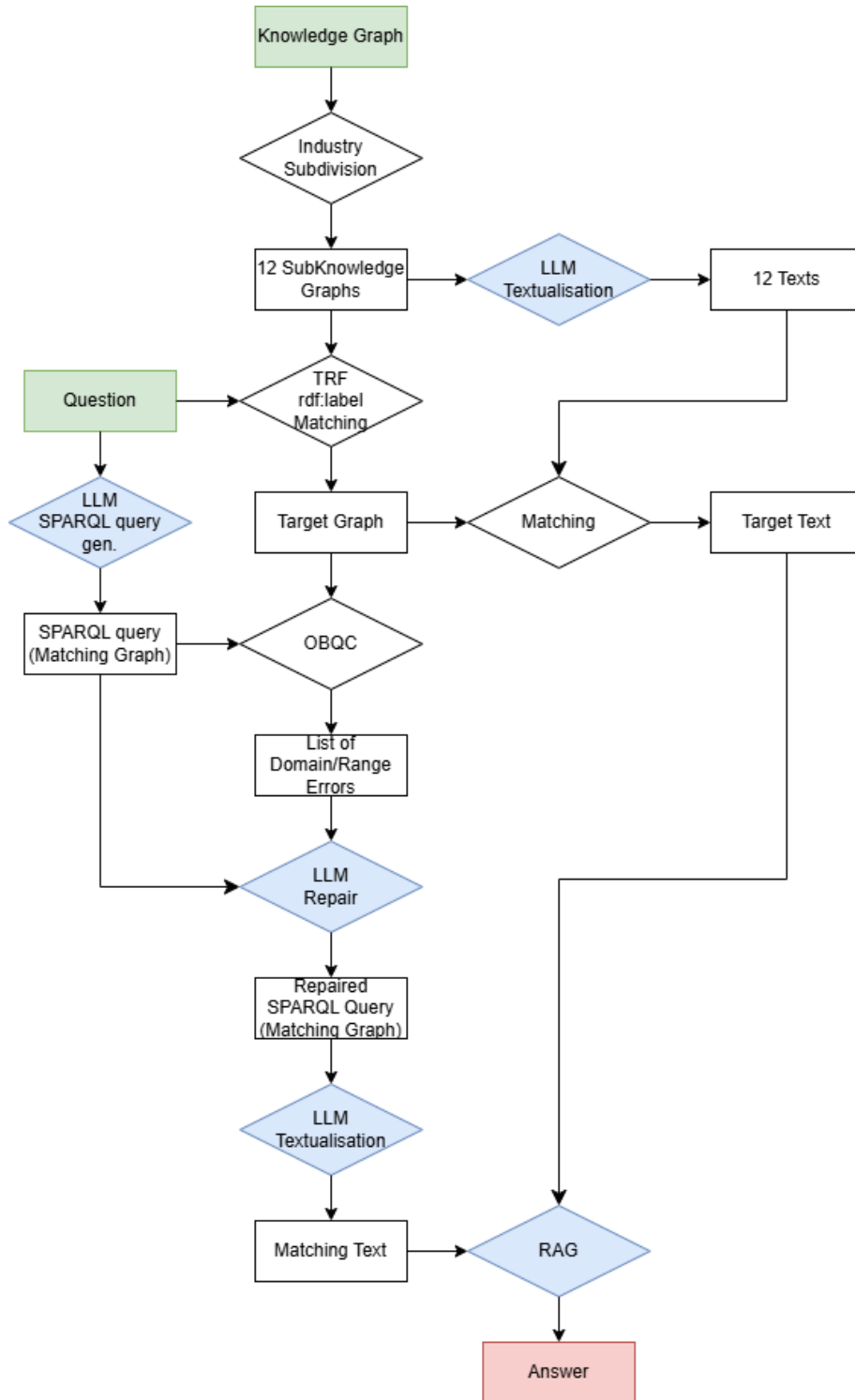


Figure 3.5: Diagram representation of the AI assistant's question answering system. Green indicates input, red denotes output, and blue highlights each instance the Large Language Model is invoked.

As illustrated in Figure 3.5, the AI assistant is supposed to take as input a question and the Knowledge Graph and answer the question. The strategy adopted for retrieving and organising information from

the data source is a combination of Subgraph-based retriever and Text organiser: the Knowledge Graph is first divided into 12 subgraphs – one for each instance of the class Industry – which are then converted to natural language by the Large Language Model. Therefore, the AI assistant has 12 pairs of subgraphs and texts to gather data from. To access the correct information pair, it was decided to deploy the open-source library spaCy and, more specifically, its trained pipelines for Natural Language Processing. The trained pipeline chosen for this project is “en_core_web_trf”, based on the Encoder model RoBERTa – a robustly optimized version of BERT (Y. Liu, 2019) – and trained on English language. The trained pipeline captures the name of the organisation from the question, which is then used to identify the correct subgraph by locating the instance of the class Company whose `rdf:label` matches the extracted organisation name. Since texts and graph modules share the same name, the target text will be the one whose name matches the target graph. At this point, the AI model generates a SPARQL query which goes through an Ontology-Based Query Check (OBQC).

```
def check_domain_violation(query_graph: Graph, ontology_graph: Graph) -> list:
    violations = []
    for s, p, o in query_graph.triples((None, None, None)):
        domain = ontology_graph.value(subject=p, predicate=RDFS.domain)
        if domain and not (s, RDF.type, domain) in query_graph:
            violations.append({
                "type": "domain",
                "property": p,
                "expected_domain": domain,
                "subject": s
            })
    return violations

def check_range_violation(query_graph: Graph, ontology_graph: Graph) -> list:
    violations = []
    for s, p, o in query_graph.triples((None, None, None)):
        range_ = ontology_graph.value(subject=p, predicate=RDFS.range)
        if range_ and not (o, RDF.type, range_) in query_graph:
            violations.append({
                "type": "range",
                "property": p,
                "expected_range": range_,
                "object": o
            })
    return violations

def explain_violation(violation: dict) -> str:
    if violation["type"] == "domain":
        return (f"The property {violation['property']} has domain {violation['expected_domain']}, "
                f"but its subject {violation['subject']} does not match or inherit from it.")
    elif violation["type"] == "range":
        return (f"The property {violation['property']} has range {violation['expected_range']}, "
                f"but its object {violation['object']} does not match or inherit from it.")
    return "Unknown violation."
```

Figure 3.6: Python code-lines for Ontology-Based Query Check and Verbalisation of spotted errors.

Each domain or range error identified thanks to the Ontology is returned as a string of text, following the scheme in Figure 3.6, which is then exploited by the LLM to repair occurring mistakes in the SPARQL query, according to the LLM repair approach described in paragraph 2.3.2. The query results are then textualized by the AI model and used for RAG along with the target text. This way, the hallucination risk should be minimised, and the AI assistant should be able to provide accurate answers to input questions.

3.5 Final results

The architecture was tested using Mistral’s Large Language Models, since the French AI company gives free access to their API for research purposes. More specifically, LLM textualisation exploited Mistral NeMo, a 12-billion-parameter model with a context window of 128K tokens designed for global multilingual applications (Mistral AI, 2024); LLM SPARQL query generation and LLM Repair used the latest version of Codestral, a 22-billion-parameter with a 256K-token context window fine-tuned for coding (Mistral AI, 2025); LLM SPARQL query textualisation deployed Mistral Small 3.1, a 24-billion-parameters model with a 128K tokens context window (Mistral AI, 2025); and RAG utilised Mistral Large, Mistral’s top-tier reasoning model for high complexity tasks with 124 billion parameters and a context window of 128K tokens (Mistral AI, 2024). The reason for using Mistral NeMo in LLM Textualisation instead of Mistral Large or even Mistral Small was the prompt length: even with a chunked Knowledge Graph, the prompt resulted too long for these 2 models which exceeded read time and returned error, preventing successful textualisation of the datasets. Mistral NeMo, being the model with lowest number of the parameters, was significantly faster and managed to turn RDF data into texts.

```
# Setup LLM
llm = ChatMistralAI(model="open-mistral-nemo", api_key=BEE_KEY)

# Prompt template
prompt_template = PromptTemplate(
    input_variables=["ontology"],
    template="""
You are a financial analyst and ontology expert. Given the following ontology module in Turtle format, generate a **detailed and comprehensive textual description** of each company. Your goal is to **translate every RDF triple** related to each company into **clear, natural language**.

For **each company**, follow this structure:

1. **General Overview**: Include revenues and costs variation, market capitalisation, future investments, amortisations, and dividends distributed to shareholders.
2. **Context**:
    - Describe the **industry** they belong to and the **country** they operate in.
    - Include any available indexes, labels, or other metadata.
3. **Financial Statements**:
    - **Income Statement**: Describe all available line items (e.g., EBIT, EBITDA, Normalized EBITDA, Tax Rate, Total Unusual Items, Net Income, Normalized Income, Interest Expense, Reconciled Cost of Revenue, Reconciled Depreciation, Tax Effect of Unusual Items, Total Unusual Items, Total Unusual Items Excluding Goodwill).
    - **Balance Sheet**: Describe all available line items (e.g. Cash and Cash Equivalents, Other Short Term Investments, Accounts Receivable, Allowance for Doubtful Accounts, Net Debt, Total Debt, Share Issued, Ordinary Shares Number, Treasury Shares Number, Tangible Book Value).
    - **Cash Flow Statement**: Describe all available line items (e.g. Beginning Cash Position, End Cash Position, Free Cash Flow, Supplemental Data, Issuance of Debt, Repayment of Debt, Issuance of Capital Stock, Capital Expenditure, Changes in Cash, Income Tax Paid Supplemental Data, Interest Paid Supplemental Data).
4. **Shares Outstanding**: Describe the number and variation if available.

**IMPORTANT**:
- **Rephrase** every RDF statement involving the company, its documents, and their properties in human-readable language.
- Do **not** skip any RDF predicate or literal.
- Do **not** use RDF or ontology-specific terms (e.g., "object property", "owl:Class"). Focus on financial storytelling.

Ontology Module:
{ontology}"""
)
```

Figure 3.7: Prompt Template used for LLM Textualisation. It takes a Knowledge Graph in Input and returns a text containing all the information described in the Turtle document.

The prompt used for LLM textualisation was as illustrated in Figure 3.7 and is reproduced below:

You are a financial analyst and ontology expert. Given the following ontology module in Turtle format, generate a detailed and comprehensive textual description of each company. Your goal is to translate every RDF triple related to each company into clear, natural language. For each company, follow this structure:

1. General Overview: Include revenues and costs variation, market capitalisation, future investments, amortisations, and dividends distributed to shareholders.

2. Context:

- Describe the industry they belong to and the country they operate in.*
- Include any available indexes, labels, or other metadata.*

3. Financial Statements:

- Income Statement: Describe all available line items (e.g., EBIT, EBITDA, Normalized EBITDA, Tax Rate, Total Unusual Items, Net Income, Normalized Income, Interest Expense, Reconciled Cost of Revenue, Reconciled Depreciation, Tax Effect of Unusual Items, Total Unusual Items, Total Unusual Items Excluding Goodwill).

- Balance Sheet: Describe all available line items (e.g. Cash and Cash Equivalents, Other Short Term Investments, Accounts Receivable, Allowance for Doubtful Accounts, Net Debt, Total Debt, Share Issued, Ordinary Shares Number, Treasury Shares Number, Tangible Book Value).

- Cash Flow Statement: Describe all available line items (e.g. Beginning Cash Position, End Cash Position, Free Cash Flow, Supplemental Data, Issuance of Debt, Repayment of Debt, Issuance of Capital Stock, Capital Expenditure, Changes in Cash, Income Tax Paid Supplemental Data, Interest Paid Supplemental Data).

4. Shares Outstanding: Describe the number and variation if available.

IMPORTANT:

- Rephrase every RDF statement involving the company, its documents, and their properties in human-readable language.*
- Do not skip any RDF predicate or literal.*

- Do not use RDF or ontology-specific terms (e.g., "object property", "owl:Class"). Focus on financial storytelling.

Ontology Module:

{ontology}

After multiple tests, results indicate that Mistral NeMo's performance gets more inconsistent as the length of the Knowledge Graph grows: while it consistently returns what asked when the input is the Knowledge Graph "Solar" or "Electronics and Computer Distribution", which are the smallest RDF documents out of the 12 obtained by the graph subdivision (around 1000 triple each), its output is either incomplete (retrieving only general information and omitting financial statements) or excessively summarised, when the documents contained more triples (the other documents contained between 1600 and 1900 triples), leading to serious information loss.

✓	Salvato modulo: industry_modules/Communication_Equipment.ttl
📊	Communication_Equipment → 19 company - 1705 triple
✓	Salvato modulo: industry_modules/Computer_Hardware.ttl
📊	Computer_Hardware → 18 company - 1679 triple
✓	Salvato modulo: industry_modules/Consumer_Electronics.ttl
📊	Consumer_Electronics → 20 company - 1844 triple
✓	Salvato modulo: industry_modules/Electronic_Components.ttl
📊	Electronic_Components → 19 company - 1767 triple
✓	Salvato modulo: industry_modules/Electronics_& Computer_Distribution.ttl
📊	Electronics_& Computer_Distribution → 8 company - 943 triple
✓	Salvato modulo: industry_modules/Information_Technology_Services.ttl
📊	Information_Technology_Services → 20 company - 1817 triple
✓	Salvato modulo: industry_modules/Scientific_& Technical_Instruments.ttl
📊	Scientific_& Technical_Instruments → 19 company - 1704 triple
✓	Salvato modulo: industry_modules/Semiconductor_Equipment_& Materials.ttl
📊	Semiconductor_Equipment_& Materials → 20 company - 1786 triple
✓	Salvato modulo: industry_modules/Semiconductors.ttl
📊	Semiconductors → 18 company - 1746 triple
✓	Salvato modulo: industry_modules/Software_-_Application.ttl
📊	Software_-_Application → 20 company - 1823 triple
✓	Salvato modulo: industry_modules/Software_-_Infrastructure.ttl
📊	Software_-_Infrastructure → 20 company - 1688 triple
✓	Salvato modulo: industry_modules/Solar.ttl
📊	Solar → 9 company - 1029 triple

Figure 3.8: Subgraph sizes. The modules "Electronics and Computer Distribution" and "Solar" are considerably smaller than the other 10 modules.

Initially, the LLM SPARQL query generation process, was meant to generate a new query each time the AI Assistant is asked a question, ensuring flexibility to the user's requests. However, using the question "What's the financial situation of 'company'?" as an input, Codestral was never able to return a correct query, regardless of the company taken into analysis. The resulting queries lacked some line items or could not match the given Ontology. For this reason, a fixed structure was adopted to ensure higher chances of Ontology matching.

```
def generate_sparql_with_mistral(companyname: str, ontology_ttl: str) -> str:
    llm = ChatMistralAI(model="codestral-latest", api_key=BEE_KEY)
    prompt_template = PromptTemplate(
        input_variables=["ontology", "company_name"],
        template="""Given the OWL model described in the following TTL file:
        {ontology}

        And the name of a company:
        {company_name}

        Replace 'placeholder' in the SPARQL query below to filter results for the given company name. Ensure that the syntax is valid and the query is executable.

        Return ONLY the updated SPARQL query—do not explain anything.

        PREFIX financial: <http://art.uniroma2.it/ontologies/financial-ontology#>
        PREFIX dc: <http://purl.org/dc/elements/1.1/>
        PREFIX grddl: <http://www.w3.org/2003/g/data-view#>
        PREFIX owl: <http://www.w3.org/2002/07/owl#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
        PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

        SELECT * WHERE {{
            VALUES ?statementType {{
                financial:MarketCapitalization
                financial:Dividends
                financial:RevenueVariation
                financial:CostVariation
                financial:IsInCountry
                financial:IsInIndustry
                financial:IsInSector
                financial:IncomeStatement
                financial:BalanceSheet
                financial:CashFlow
                financial:SharesOutstanding
            }}
            ?company rdf:type ?label.
            FILTER(regex(str(?label), "placeholder", "i"))
            ?company ?statementType ?statement .
            ?statement ?property ?value .
            FILTER (isLiteral(?value))
        }}
        ORDER BY ?statementType ?statement ?property
        """)
    return llm.invoke(prompt_template.format(ontology=ontology_ttl, company_name=companyname)).text
```

Figure 3.9: Prompt Template for LLM SPARQL query generation. Given the Knowledge Graph and a company name it returns a SPARQL query for retrieving each variable and Financial Statement contained in the graph.

The prompt used for this process, as illustrated in Figure 3.9, is represented below:

Given the OWL model described in the following TTL file:

{ontology}

And the name of a company:

{company_name}

Replace 'placeholder' in the SPARQL query below to filter results for the given company name. Ensure that the syntax is valid and the query is executable. Return ONLY the updated SPARQL query—do not explain anything.

PREFIX financial: <http://art.uniroma2.it/ontologies/financial-ontology#>

PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX grddl: <http://www.w3.org/2003/g/data-view#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX *rdf*: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>>

PREFIX *rdfs*: <<http://www.w3.org/2000/01/rdf-schema#>>

PREFIX *xsd*: <<http://www.w3.org/2001/XMLSchema#>>

SELECT * *WHERE* {{

VALUES ?*statementType* {{

financial:MarketCapitalization

financial:Dividends

financial:RevenueVariation

financial:CostVariation

financial:IsInCountry

financial:IsInIndustry

financial:IsInSector

financial:IncomeStatement

financial:BalanceSheet

financial:CashFlow

financial:SharesOutstanding

}}

?*company* *rdf:label* ?*label*.

FILTER(*regex*(*str*(?*label*), "placeholder", "i"))

?*company* ?*statementType* ?*statement* .

?*statement* ?*property* ?*value* .

FILTER (*isLiteral*(?*value*))

}}

ORDER BY ?statementType ?statement ?property

This way, instead of generating a new query based on a question, the LLM can just replace “placeholder” with the name of the company retrieved thanks to spaCy’s pretrained pipeline. This template-based approach increased the consistency of Codestral’s responses. Occurring mistakes are then dealt with by LLM Repair, which further increases the queries’ accuracy.

```
def repair_query_with_mistral(bad_query: str, ontology_str: str, violations: list, api_key: str) -> str:
    llm = ChatMistralAI(model="codestral-latest", api_key=api_key)
    prompt = PromptTemplate(
        input_variables=["ontology", "query", "issues"],
        template=(
            "You are an expert in RDF and SPARQL. Given the following ontology in Turtle format:\n\n"
            "{ontology}\n\n"
            "The following SPARQL query has the following issues:\n\n"
            "{issues}\n\n"
            "Please rewrite the query to fix the identified issues. ONLY return the corrected SPARQL query.\n"
            "If no issues are found or the query is already correct, return the original query unchanged.\n\n"
            "Query:\n{query}"
        )
    )
```

Figure 3.10: Prompt Template for LLM Repair. Given the Knowledge Graph, Domain and Range issues individuated and the SPARQL query generated in the previous pass, the LLM Repair stage returns a repaired SPARQL query, resolving domain and range issues..

It is worth noting that LLM Repair could not do anything about queries generated with the previous query generation approach, as they were too compromised to work with. The LLM SPARQL query textualisation process, too, benefitted from the change in approach: if Mistral Small generated text on an unspecified German or American company with the previous query generation approach, it began generating detailed reports on the desired company, listing Financial Statements and general information as requested.

```
[83] llm = ChatMistralAI(model="mistral-small-latest", api_key=BEE_KEY)

# Prompt template--NEEDS WORK
prompt_template = PromptTemplate(
    input_variables=["results"],
    template=(
        "You are an expert in finance and ontologies. "
        "Given the following SPARQL query result, describe the company's"
        "financial situation, making explicit reference to its Financial Statement\n"
        "Return: \n"
        "- its general situation taking into account its variation in costs and revenues,"
        "its market capitalisation, its future investments and amortisations, and the amount of dividends it distributes to shareholders\n"
        "- describe the industry and country they are operating in, listing their properties and indexes\n"
        "- its IncomeStatement\n"
        "- its BalanceSheet\n"
        "- its CashFlow\n"
        "- its SharesOutstanding\n"
        "Avoid technical RDF language. Be as specific as possible. For each property or item in the financial statements, report its **value**.\n\n"
        "sparql_graph:\n{results}"
    )
)
```

Figure 3.11: Prompt Template for LLM SPARQL query textualisation. Given the results obtained from the repaired SPARQL query, it returns a text describing the firm considered.

The prompt used for this process, as illustrated in Figure 3.11, is reproduced below:

"You are an expert in finance and ontologies. Given the following SPARQL query result, describe the company's financial situation, making explicit reference to its Financial Statement. Return:

- its general situation taking into account its variation in costs and revenues, its market capitalisation, its future investments and amortisations, and the amount of dividends it distributes to shareholders.*
- describe the industry and country they are operating in, listing their properties and indexes*
- its IncomeStatement*
- its BalanceSheet*
- its CashFlow*
- its SharesOutstanding*

Avoid technical RDF language. Be as specific as possible. For each property or item in the financial statements, report its value.

sparql_graph:

{results}

For the final LLM invocation, concerning RAG and question answering, it was decided to carry out 2 different experiments: one asking Mistral Large to produce a complete fundamental analysis and the other asking the same model to provide a PEST analysis and other context information about the company. The reason for this resides within the LLM's flaw in understanding mathematical concepts and discern relevant information for problem-solving, as proven in the paper "GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models" (I. Mirzadeh, 2024).

```

model = ChatMistralAI(model = "mistral-large-latest",mistral_api_key=BEE_KEY)
prompt = ChatPromptTemplate.from_template("""You are Hiroshi Abe, a helpful and expert AI Financial Assistant specializing in fundamental analysis of companies.

Your role is to analyze a company's financial health based on the provided data sources. You MUST only use the two sources below to generate your answer.

When analyzing a company's financial situation, follow these steps:

1. PEST Analysis
Perform a Political, Economic, Social, and Technological (PEST) analysis relevant to the firm.

2. Financial Ratios Evaluation
Calculate and interpret:
- Return on Equity (ROE)
- Return on Investment (ROI)
- Return on Assets (ROA)
Use the following formulas for assessment:
•  $ROE \geq r_f + \beta_i \times (r_M - r_f)$ 
•  $ROI \geq ROE$ 
Where:
-  $r_f$  = 4% (US Treasury Bond Return Rate)
-  $r_M$  = 23.63% (Technology Sector Market Return Rate)
-  $\beta_i$  = Industry Beta
Explain each computation you do, highlighting every line item used
Conclude whether the financial situation is Good or Not Good based on these thresholds.

3. Forecasting
Provide 5-year forecasted:
- Balance Sheets
- Income Statements
These forecasts must be based on estimated changes in costs, revenues, and amortization schedules.

4. Firm Valuation (DCF Method)
Compute the firm's valuation using the Discounted Cash Flow (DCF) method.

---

Answer the question based only on the provided context:
{context}

---

User Question:
{input}""")
)

```

Figure 3.12: Prompt Template for RAG. In this case the LLM is being tested for a complete Fundamental Analysis.

The first prompt used for the experiment, as shown in Figure 3.12, is:

You are Hiroshi Abe, a helpful and expert AI Financial Assistant specializing in fundamental analysis of companies. Your role is to analyze a company's financial health based on the provided data sources. You MUST only use the two sources below to generate your answer. When analyzing a company's financial situation, follow these steps:

1. PEST Analysis

Perform a Political, Economic, Social, and Technological (PEST) analysis relevant to the firm.

2. Financial Ratios Evaluation

Calculate and interpret:

- Return on Equity (ROE)

- Return on Investment (ROI)

- Return on Assets (ROA)

Use the following formulas for assessment:

- $ROE \geq r_f + \beta_i(\bar{r}_M - r_f)$

- $ROI \geq ROE$

Where:

- $r_f = 4\%$ (US Treasury Bond Return Rate)
- $\bar{r}_M = 23.63\%$ (Technology Sector Market Return Rate)
- $\beta_i = \text{Industry Beta}$

Explain each computation you do, highlighting every line item used.

Conclude whether the financial situation is Good or Not Good based on these thresholds.

3. Forecasting

Provide 5-year forecasted:

- Balance Sheets
- Income Statements

These forecasts must be based on estimated changes in costs, revenues, and amortization schedules.

4. Firm Valuation (DCF Method)

Compute the firm's valuation using the Discounted Cash Flow (DCF) method.

Answer the question based only on the provided context:

{context}

User Question:

{input}

```

model = ChatMistralAI(model = "mistral-large-latest",mistral_api_key=BEE_KEY)
prompt = ChatPromptTemplate.from_template("""You are Hiroshi Abe, a helpful and expert AI Financial Assistant specializing in fundamental analysis of companies.

Your role is to provide context information and help analysts make fundamental analysis. You MUST only use the two sources below to generate your answer.

When retrieving data about a company, follow these steps:

1. PEST Analysis
Analyse the company from a macro perspective, according to the PEST Analysis paradigm:
- Political
- Economic
- Social
- Technological
For each point, write a brief comment on the company's situation.

2. General Data
Return general information, including:
- Market Capitalization
- Industry
- Country
- Variation in Costs and Revenues
- Future Investments
- Shares Outstanding.|

3. Financial Statements
Return its Financial Statement:
- Income Statement
- Balance Sheet
- Cash Flow

Use bullet points or structured paragraphs for each section. Keep the language clear and professional. Be concise but thorough.
---

Answer the question based only on the provided context:
{context}

---

User Question:
""")

```

Figure 3.13: Prompt Template for RAG. In this case the LLM is being tested as an information retriever, as it is just asked to make a PEST Analysis and return the company's Financial Statements.

The second prompt is:

You are Hiroshi Abe, a helpful and expert AI Financial Assistant specializing in fundamental analysis of companies. Your role is to provide context information and help analysts make fundamental analysis. You MUST only use the two sources below to generate your answer. When retrieving data about a company, follow these steps:

1. PEST Analysis

Analyse the company from a macro perspective, according to the PEST Analysis paradigm:

- Political***
- Economic***
- Social***
- Technological***

For each point, write a brief comment on the company's situation.

2. General Data

Return general information, including:

- Market Capitalization***
- Industry***
- Country***
- Variation in Costs and Revenues***
- Future Investments***
- Shares Outstanding.***

3. Financial Statements

Return its Financial Statement:

- Income Statement***
- Balance Sheet***
- Cash Flow***

Use bullet points or structured paragraphs for each section. Keep the language clear and professional. Be concise but thorough.

Answer the question based only on the provided context:

{context}

User Question:

{input}

In both cases, Mistral Large executed its task thoroughly, going through each phase of the analysis/report. In each iteration of the first experiment, the model, in most cases, refused to make the 5-years forecast of the Financial Statements and the DCF due to lack of information, which is acceptable, even if the Knowledge Graph was provided with variation in Costs and Revenues esteems. As expected, the second experiment yielded more reliable outputs compared to the first one. When computing the Financial Indicators (ROI, ROE and ROA), the AI model could not tell the difference between positive and negative numbers, making serious comparison mistakes. It also made computation mistakes, especially when dealing with products and ratios between percentages. On the

other hand, in the second experiment the LLM returned all the relevant information contained in the Knowledge Graph, writing a complete PEST analysis and returning the company's last financial statement. However, results in the second experiment were far from perfect: it hallucinated when describing the name of some properties, despite those being labelled on the Ontology. So, for example, the Global Digitalisation Index (GDI) in one iteration became the Gender Development Index. Moreover, when describing Variation in Costs and Revenues, it just ignores the presence of the 2 variables "Revenue Variation" and "Cost Variation", proceeding with an analysis of the Income Statement. Values of each property and line item were reported accurately, with mistakes rarely occurring (counting the number of line items correctly reported over the number of line items returned, the percentage of successful retrieval of line items was over 95%). The model exhibited a good grade of flexibility, when changing question: the answer to the question "What's the debt of Oracle?" was concise and accurate, without writing comments. The SPARQL query support to the LLM proved to be effective: damages due to the Knowledge Graph textualisation errors were repaired thanks to the SPARQL query and the information retrieved from the dataset with it. So, this architecture constitutes a solid implementation of GraphRAG, ensuring higher levels of accuracy. Given the resource constraints of this work, future research could assess the performance of this architecture with larger AI Models, such as OpenAI's GPT-series or Anthropic's Claude 3.7 Sonnet.

```
To analyze Oracle Corporation's financial situation, we will follow the steps of performing a PEST analysis, evaluating financial ratios, forecasting future financial statements, and valuing the firm using the Discounted Cash Flow (DCF) method.

### 1. PEST Analysis

**Political:**
- **Country:** United States
- **GDP:** $25,439,700,000,000
- **Tax Rate:** 25%
- **Moody's Rating:** Aa3
- **S&P Rating:** AA+

The political environment in the United States is stable, with a high GDP and strong credit ratings, which is beneficial for Oracle's operations.

**Economic:**
- **Inflation:** 2.5%
- **Gross Domestic Income (GDI):** 78.8
- **Equity Risk Premium:** 0.05333417

The economic environment is characterized by moderate inflation and a strong equity risk premium, which can impact Oracle's cost of capital and overall financial performance.

**Social:**
- **Fertility Rate:** 1.6
- **Median Age:** 38.9
- **Population Over 65:** 17.43%
- **Population:** 326,766,700

The social factors indicate a mature population with a low fertility rate, which can influence consumer behavior and workforce demographics.

**Technological:**
- **Research GDP:** 3.46%
- **World Development Competitiveness Ranking (MDCR):** 91.31
- **Yearly Change:** 0.57%

The technological environment is favorable, with a significant portion of GDP allocated to research and a high competitiveness ranking, which supports Oracle's innovation and growth.

### 2. Financial Ratios Evaluation

**Return on Equity (ROE):**
```

Figure 3.14: Part of Mistral Large's output in one iteration. Despite being accurate and trying to solve all its tasks thoroughly, it hallucinates and generates new variable names.

Since this project's ultimate objective is to understand whether Generative AI can be implemented into a business's operations, it is worth assessing the economic impact that this architecture's implementation could have on a firm. The main cost sources of the architecture are data retrieval, AI model deployment and computing power. Since data was acquired using the Yahoo Finance Python Library, available for free, and the AI models were freely provided thanks to Mistral's Research

License, the only cost incurred in this project was due to the acquisition of computing units via Google Colab, for a total amount of 22.58€ (11.29€/month, the AI experimentation started in April). Considering the data quality and the model's performance, there is margin for improvement and further investments. Assuming that a financial firm already has acquired market data or has a subscription to any financial data provider, the only remaining variation in costs would result from the implementation of Generative AI into the business process. AI inference costs are usually measured in \$/millions of tokens.

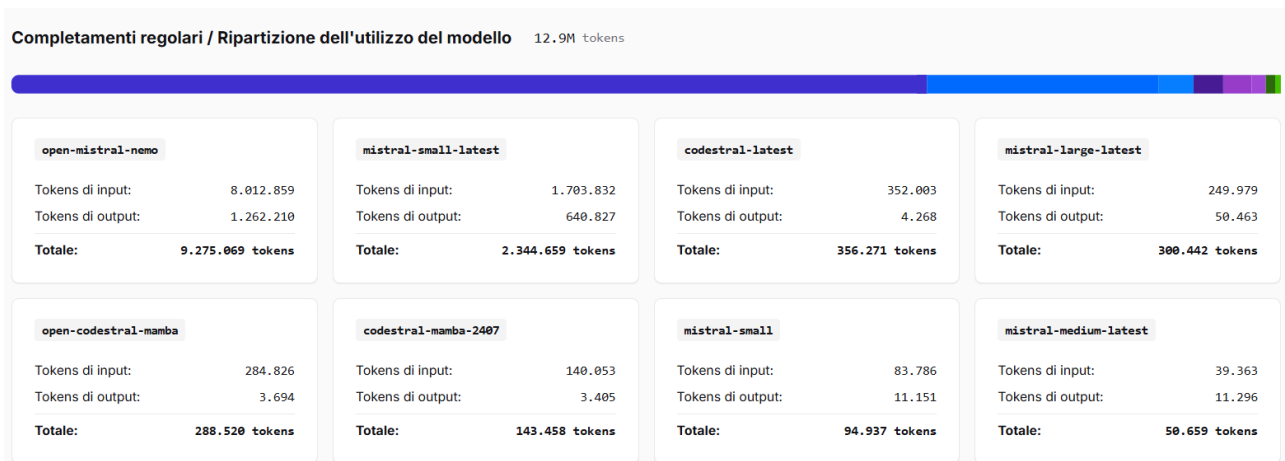


Figure 3.15: Mistral's tool for token accounting. The whole project required processing 12.9 million tokens.

As illustrated in figure 3.15, during May, the project required processing 12.9 million tokens, of which over 9 million were used for dataset textualisation. Considering that:

- LLM textualisation requires processing roughly 500,000 tokens but can be performed once every data update;
- LLM SPARQL query generation and LLM Repair requires processing roughly 87,000 tokens on aggregate;
- LLM SPARQL query textualisation requires processing roughly 22,000 tokens;
- RAG question answering requires processing on average 9,000 tokens, assuming that the LLM is asked to generate a complete report, according to the RAG prompt;

And that, according to the AI Annual Report by Stanford University, the Inference Cost of GPT4o for PhD level questions is 0.12\$/million tokens, the total cost of a single Architecture's usage is \$0.01, plus the amortised portion of the LLM textualisation cost, which amounts to \$0.06 per data update. Given current AI market conditions, this architecture appears to be affordable for any financial firm. Therefore, the proposed architecture can be regarded as a viable implementation of Generative AI

within business operations offering both technical functionality and economic sustainability. Its performance in complex real-world scenarios - particularly with enhanced access to Large Language Models and richer financial data - is left to further investigation. Future studies may explore how this architecture could be adapted across different industries, ultimately shaping the role of Generative AI and semantically enriched data in decision-making processes.

CONCLUSION

This thesis presents an LLM-powered question answering architecture that leverages a Knowledge Graph built upon an OWL Finance Ontology designed to represent information about firms. To achieve this, VocBench was used to edit the Ontology and, via the Sheets2RDF tool, to upload data directly from spreadsheets, while ShowVoc was used for visualisation and inspection of the Knowledge Graph. In parallel, it incorporates the Ontology-based Query Check system introduced by Allemang and Sequeda to enhance accuracy. For this work, it was decided to adopt Mistral AI LLMs, due to the firm's free license for research use. The combination of the two demonstrates a viable and economically feasible implementation of generative AI into financial businesses, enabling natural language interaction with data. The accuracy of the system is strengthened by OBQC which makes up for information loss that has been experienced during the textualisation of RDF data. This further validates the idea of a RDF + gen AI synergy, useful for semantic web practitioners, data scientists and professionals in financial institutions, as Large Language Models can effectively exploit semantically enriched data this way. The designed ontology is an outcome per se: thanks to the high reusability that characterises semantic models, it can serve as a foundation for building new datasets or can be integrated into other existing vocabularies for broader purposes. In a similar way, the LLM question answering architecture can be exploited as a reusable template which can be adapted to different sectors. Furthermore, the usage of open-source tools and the effort made for cost-efficiency, as well as the use of Mistral NeMo, Codestral, Mistral Small 3.1 and Mistral Large and the yielding modularisation of the architecture may enable implementation of this system in smaller firms. However, the usage of Knowledge Graphs with a greater number of triples may pose significant challenges, as some criticalities were raised during the textualisation of the Knowledge Graph due to the length of subgraphs. It could require finer and harder methods for Ontology subdivision, which can imply information loss. Nevertheless, several LLM-related challenges remain. Data scarcity, lack of transparency of AI firms, environmental costs and Cybersecurity risks are all factors which cannot be ignored when deciding to integrate this system into business operations. The introduction and enhancement of legislation such as the European AI Act may help this integration, setting up a safe environment for an AI Economy. In conclusion, while the experimental results obtained under current resource constraints are promising, further resources to be invested for setting up an industry-standard solution and further research is required to assess the scalability, robustness and long-term viability of the proposed architecture within complex, real-world financial environments.

REFERENCES

- A. La Bella, E. B. (2016). Economia e Organizzazione Aziendale. In E. B. A. La Bella, *Economia e Organizzazione Aziendale* (p. 487-490). Santarcangelo di Romagna: Maggioli Editore.
- A. Radford, J. W. (2019, February 14). Language Models are Unsupervised Multitask Learners. *Better language models and their implications*. San Francisco, California, United States of America: OpenAI.
- A. Radford, K. N. (2018, June 11). Improving Language Understanding by Generative Pre-Training. *Improving language understanding with unsupervised learning*. San Francisco, California, United States of America: OpenAI.
- A. Vaswani, N. S. (2017). Attention Is All You Need. *31st Conference on Neural Information Processing Systems (NIPS 2017)*. Long Beach, CA, USA: arXiv.
- AI Index Steering Committee, Institute for Human-Centered AI, Stanford University. (2025). *The AI Index 2025 Annual Report*. Stanford, CA: Stanford University.
- Anthropic. (2025, April 21). *Claude 3.7 Sonnet*. Tratto da <https://www.anthropic.com/claude/sonnet>: <https://www.anthropic.com/claude/sonnet>
- ART: Artificial Intelligence Research at Tor Vergata. (2025, May 15). *CODA*. Tratto da ART: Artificial Intelligence at Roma Tor Vergata: <https://art.uniroma2.it/coda/>
- ART: Artificial Intelligence Research at Tor Vergata. (2025, May 15). *ShowVoc*. Tratto da ShowVoc: <https://showvoc.uniroma2.it/>
- ART: Artificial Intelligence Research at Tor Vergata. (2025, May 15). *The Sheet2RDF VocBench tool*. Tratto da ART: Artificial Intelligence at Roma Tor Vergata: https://art.uniroma2.it/sheet2rdf/documentation/vb_tool/
- ART: Artificial Intelligence Research at Tor Vergata. (2025, May 15). *VocBench*. Tratto da VocBench: <https://vocbench.uniroma2.it/>
- Borsa Italiana. (2025, 01 20). *Glossario Finanziario-Borsa Italiana*. Tratto da Glossario Finanziario-Borsa Italiana: <https://www.borsaitaliana.it/borsa/glossario.html>

- D. Allemang, J. S. (2024, May 20). Increasing the LLM Accuracy for Question Answering: Ontologies to the Rescue! *Increasing the LLM Accuracy for Question Answering: Ontologies to the Rescue!* arXiv.
- D. Bachanau, K. C. (2015). Neural Machine Translation By Jointly Learning To Align and Translate. *ICLR 2015*. arXiv.
- D. Huynh, J. H. (2023, July 09). *PoisonGPT: How We Hid a Lobotomized LLM on Hugging Face to Spread Fake News*. Tratto da Mithril Security Blog: <https://blog.mithrilsecurity.io/poisongpt-how-we-hid-a-lobotomized-llm-on-hugging-face-to-spread-fake-news/>
- D. Jurafsky, J. M. (2025, January 12). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3rd edition*. Tratto da Speech and Language Processing: <https://web.stanford.edu/~jurafsky/slp3/>
- Damodaran, A. (2025, April 29). *Financial Ratios and Measures*. Tratto da Damodaran online: https://pages.stern.nyu.edu/~adamodar/New_Home_Page/definitions.html
- Dublin Core Metadata Initiative. (2025, May 05). *Dublin Core™ Metadata Element Set, Version 1.1: Reference Description*. Tratto da Dublin Core: <https://www.dublincore.org/specifications/dublin-core/dces/>
- E. Hu, Y. S.-Z. (2021, June 17). LoRA: Low-Rank Adaptation of Large Language Models. *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv.
- Encyclopaedia Britannica. (2025, Mar 26). *Turing machine*. Tratto da Britannica: <https://www.britannica.com/technology/Turing-machine>
- European Parliament. (2025, February 19). *EU AI Act: first regulation on artificial intelligence*. Tratto da European Parliament: <https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai>
- F.A. Gers, J. S. (1999, September 7). Learning to Forget: Continual Prediction with LSTM. Edinburgh, Scotland, United Kingdom: 1999 Ninth International Conference on Artificial Neural Networks ICANN 99.

- Fiorelli M., L. T. (2015). Sheet2RDF: a Flexible and Dynamic Spreadsheet Import&Lifting Framework for RDF. In Y. S.-H. M. Ali, *Current Approaches in Applied Artificial Intelligence* (p. Vol. 9101, pp. 131-140). Springer International Publishing.
- Fiorelli M., P. M. (2014). Computer-aided ontology development architecture. *IBM Journal of Research and Development*, 58(2/3), 14:1-14:12.
- Fiorelli, M. (2021, December 20). *Protégé e VocBench*. Tratto da ART: Artificial Intelligence Research at Tor Vergata: [chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://art.uniroma2.it/teaching/ke/slides/KE%20-%2017.%20Protege%20e%20VocBench.pdf](https://art.uniroma2.it/teaching/ke/slides/KE%20-%2017.%20Protege%20e%20VocBench.pdf)
- G. Ferrero, F. D. (2006). *Analisi di Bilancio e Rendiconti Finanziari*. Milan: Giuffrè Editore.
- Gambosi, G. (2024). Probabilistic Learning. *Probabilistic Learning*. Rome, Latium, Italy: University of Rome Tor Vergata.
- Garrison R.H., N. E. (2012). *Managerial Accounting*. Milan: McGraw-Hill Education.
- Gennari J., M. M. (2003). The evolution of Protégé-2000: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1), 89-123.
- Google. (2025, April 21). *Gemini 2.5: Our most intelligent AI model*. Tratto da Google-The Keyword: <https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking>
- Google. (2025, May 15). *Google Colaboratory*. Tratto da Colab: <https://colab.google/>
- Google Cloud. (2024). *The ROI of gen AI in Financial Services-A global survey of enterprise adoption and value*. Mountain View, CA: Google.
- Google Cloud. (2024). *The ROI of GenAI*. Mountain View, CA: Alphabet.
- Griesi D., P. M. (2007). Semantic Turkey - a Semantic Bookmarking tool (System Description). *Semantic Turkey - a Semantic Bookmarking tool (System Description)*. Innsbruck, Austria: 4th European Semantic Web Conference.
- H. Han, Y. W. (2025, January 8). Retrieval-Augmented Generation with Graphs (GraphRAG). *Retrieval-Augmented Generation with Graphs (GraphRAG)*. arXiv.

- H.K. Bako, A. B. (2024, July 9). Evaluating the Semantic Profiling Abilities of LLMs for Natural Language Utterances in Data Visualization. *Evaluating the Semantic Profiling Abilities of LLMs for Natural Language Utterances in Data Visualization*. arXiv.
- Hauser, R. (2007). Line Search Methods for Unconstrained Optimisation. *Line Search Methods for Unconstrained Optimisation-Lecture 8, Numerical Linear Algebra and Optimisation*. Oxford, England, United Kingdom: Oxford University Computing Laboratory.
- Huawei Technologies Co. Ltd. (2024). *Global Digitalization Index*. Shenzhen: Huawei Technologies Limited Co. Ltd.
- I. Goodfellow, Y. B. (2016). *Deep Learning*. MIT Press.
- I. Mirzadeh, K. A. (2024, October 7). GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models. *GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models*. Cupertino, California, United States of America: arXiv.
- I. Sutskever, O. V. (2014, December 14). Sequence to Sequence Learning with Neural Networks. *Sequence to Sequence Learning with Neural Networks*. arXiv.
- International Business Machine Corporation. (2025, March 04). *What is machine learning?* Tratto da IBM: <https://www.ibm.com/think/topics/machine-learning>
- International Institute of Management Development. (2024). *IMD World Digital Competitiveness Ranking*. Losanna: IMD.
- International Valuation Standards Council. (2016, April 7). IVS 105: Valuation Approaches and Methods. *IVS 105: Valuation Approaches and Methods-Exposure Draft*. London, England, United Kingdom: International Valuation Standards Council.
- J. Devlin, M. C. (2018, October 11). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv.
- J. Wei, X. W. (2023, January 10). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. arXiv.

- Janus, M. (2024). Sleeping with One AI Open-The (In)Security of AI Supply Chain. *AI World Congress 2024*. London.
- Kramer, J. F., & Chang, X. J. (2006). How to Write Bibliographies. *Adventure Works Monthly*, 50-62.
- L. Ouyang, J. W. (2022, March 4). Training language models to follow instructions with human feedback . *Training language models to follow instructions with human feedback* . arXiv.
- Luenberger, D. G. (2009). *Investment science*. New York; Oxford: Oxford university press.
- Microsoft. (2024). *How can we advance sustainability?* Redmond, WA: Microsoft.
- Minardi, S. (2025, April 7). *Ipnocrazia, ecco perché il filosofo Xun non esiste*. Tratto da L'Espresso: <https://lespresso.it/c/-/2025/4/7/ipnocrazia-best-seller-libro-chi-e-xun/53621>
- Mistral AI. (2024, July 18). *Mistral NeMo*. Tratto da Mistral: <https://mistral.ai/news/mistral-nemo>
- Mistral AI. (2024, November 18). *Pixtral Large*. Tratto da Mistral: <https://mistral.ai/news/pixtral-large>
- Mistral AI. (2025, January 13). *Codestral*. Tratto da Mistral: <https://mistral.ai/news/codestral-2501>
- Mistral AI. (2025, March 17). *Mistral Small 3.1*. Tratto da Mistral: <https://mistral.ai/news/mistral-small-3-1>
- Moody's. (2025). *Rating Scale and Definitions*. Tratto da Moody's: chrome-extension://efaidnbmnnnibpcajpgclefindmkaj/https://www.moody.com/sites/products/productattachments/ap075378_1_1408_ki.pdf
- N. Cheng, Z. Y. (2024, May 10). Potential and Limitations of LLMs in Capturing Structured Semantics: A Case Study on SRL . *Potential and Limitations of LLMs in Capturing Structured Semantics: A Case Study on SRL* . arXiv.
- OpenAI. (2020, July 22). Language Models are Few-Shot Learners. *Language Models are Few-Shot Learners*. San Francisco, California, United States : OpenAI.
- OpenAI. (2022, November 30). *Introducing ChatGPT*. Tratto da OpenAI: <https://openai.com/index/chatgpt/>

- OpenAI. (2024, September 12). *Learning to reason with LLMs*. Tratto da OpenAI: <https://openai.com/index/learning-to-reason-with-llms/>
- OpenAI. (2025, April 16). *Introducing OpenAI o3 and o4-mini*. Tratto da OpenAI: <https://openai.com/index/introducing-o3-and-o4-mini/>
- P. Lewis, E. P. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*. Vancouver, Canada: arXiv.
- P. Li, J. Y. (2023, October 29). Making AI Less "Thirsty": Uncovering and Addressing the Secret Water Footprint of AI Models. *Making AI Less "Thirsty": Uncovering and Addressing the Secret Water Footprint of AI Models*. arXiv.
- Pazienza M. T., S. N. (2012). Semantic Turkey: A Browser-Integrated Environment for Knowledge Acquisition and Management. *Semantic Web Journal*, 279-292.
- R. Cyganiak, D. W. (2014, February 25). *RDF 1.1 Concepts and Abstract Syntax*. Tratto da W3C: <https://www.w3.org/TR/rdf11-concepts/>
- R. Zadeh, K. B. (2015, April 5). Lecture 11-Distributed Algorithms and Optimization. Stanford, California, United States of America: University of Stanford.
- R.N. Anthony, L. B. (2010). *Essential of Accounting Review, 10th edition*. Vignate (MI): Pearson Paravia Bruno Mondadori S.p.A.
- Ren, S. (2023, November 30). *How much water does AI consume? The public deserves to know*. Tratto da OECD.AI-Policy Observer: <https://oecd.ai/en/work/how-much-water-does-ai-consume>
- Rosenblatt, F. (1958). The Perceptron: a probabilistic model for information storage and organization in the brain. *Cornell Aeronautical Laboratory*.
- Royal Institution of Chartered Surveyors (RICS). (2023, November). Discounted cash flow valuations. *Discounted cash flow valuations-RICS practice information, global*. London, England, United Kingdom: Royal Institution of Chartered Surveyors (RICS).
- S. Harris, A. S. (2013, March 21). *SPARQL 1.1 Query Language*. Tratto da W3C: <https://www.w3.org/TR/sparql11-query/>
- S. Hochreiter, J. S. (1997). Long Short-Term Memory. *Neural Computation*, 5-9.

- Siematkowski, S. (2024, August 28). Klarna has 1800 employees it hopes AI will render obsolete. (R. Hogs, Intervistatore)
- Standard & Poor's. (2025, April 29). *Understanding Credit Ratings*. Tratto da S&P Global: <https://www.spglobal.com/ratings/en/about/understanding-credit-ratings>
- Stellato A., R. S. (s.d.). VocBench: a Web Application for Collaborative Development of Multilingual Thesauri. In M. S.-M. F. Gandon, *The Semantic Web. Latest Advances and New Domains (Lecture Notes in Computer Science)* (p. Vol. 9088, pp. 38-53). Springer, Cham.
- Stellato, A. (2023, September). RDF - Resource Description Framework. *RDF - Resource Description Framework*. Rome, Latium, Italy: ART: Artificial Intelligence Research at Tor Vergata.
- Stellato, A. (2023, September). SPARQL. *SPARQL*. Rome, Latium, Italy: ART: Artificial Intelligence Research at Tor Vergata.
- Stellato, A. F. (2020). VocBench 3: A collaborative Semantic Web editor for ontologies thesauri and lexicons. *Semantic Web*, 855-881.
- Stryker, C. (2024, October 4). *What is a recurrent neural network?* Tratto da IBM: <https://www.ibm.com/think/topics/recurrent-neural-networks>
- T. Berners-Lee, J. H. (2001). The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 1-4.
- Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, 433-460.
- W. Zou, R. G. (2024, August 13). Poisoned RAG: Knowledge Poisoning Attacks to Retrieval-Augmented Generation of Large Language Models. *Poisoned RAG: Knowledge Poisoning Attacks to Retrieval-Augmented Generation of Large Language Models*. arXiv.
- W.S. McCulloch, W. P. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biology*.
- Wall Street Prep. (2025, April 7). *Bloomberg vs Capital IQ vs Factset vs Refinitiv*. Tratto da Wall Street Prep: <https://www.wallstreetprep.com/knowledge/bloomberg-vs-capital-iq-vs-factset-vs-thomson-reuters-eikon/>
- Wall Street Prep. (2025, Febraury 04). *Normalized EBITDA*. Tratto da Wall Street Prep: <https://www.wallstreetprep.com/knowledge/normalized-ebitda/>

- Worldometer. (2025, April 29). *About Worldometer*. Tratto da Worldometer: <https://www.worldometers.info/about/>
- Y. Liu, M. O. (2019, July 26). RoBERTa: a robustly optimized BERT pretraining approach. *RoBERTa: a robustly optimized BERT pretraining approach*. arXiv.
- Y. Sun, Y. Z. (2022). NSP-BERT: A Prompt-based Few-Shot Learner Through an Original Pre-training Task --Next Sentence Prediction. *29th International Conference on Computational Linguistics* (p. 3235-3236). Gyeongju, Republic of Korea: ACL Anthology.
- Yahoo! Finance. (2025, May 5). *Technology*. Tratto da Yahoo! Finance: <https://finance.yahoo.com/sectors/technology/>

SPECIAL THANKS

Mi sono immatricolato presso la Facoltà di Ingegneria dell'Università degli Studi di Roma Tor Vergata nel mese di Settembre del 2018, ma il mio viaggio inizia il 13 Marzo 2019, giorno in cui papà è scomparso. Nel giro di un'ora, il mondo per come l'avevo conosciuto è sfumato come cenere al vento: in casa non avevo più le mie certezze e le difficoltà iniziali nello studio universitario di certo non aiutarono. Come tutti, ho dovuto affrontare l'isolamento e la solitudine indotta dalla pandemia del 2020 e 2021, arrivando nel 2022, anno della mia laurea triennale, svuotato. È stato un periodo terribile della mia vita, in cui mi sono abbruttito e ho fatto del male anche a persone che mi volevano e, straordinariamente, mi vogliono ancora bene. Il 17 Luglio 2022, quando ho concluso la mia Laurea Triennale, credevo di aver superato quei traumi ma la verità è che c'era ancora molto da rifare. Non avevo ancora ritrovato la fiducia in me stesso e nel mio valore in quanto studente e in quanto persona e sentivo ancora quella solitudine. Il primo passo di questa mia rinascita è stata la mia esperienza di studi in Portogallo, nella piccola Aveiro. Lì, vivendo delle esperienze indimenticabili e incontrando persone da tutto il mondo, ho riscoperto il mio valore in quanto uomo. Quindi il mio primo ringraziamento va proprio a tutti quelli del gruppo Erasmus: Valeria e il gruppo della palestra, Gabriele, Federico, Luca e i compagni del camper, Greta, il gruppo dei greci, il gruppo dei siciliani, Erica, tutti quelli che sono stati miei compagni di avventure per i 5 mesi migliori della mia vita. Una volta tornato in Italia, mi sono reso conto di ciò che non andava e ho provato a usare quella forza ritrovata per cambiare le cose. È con questo spirito che ho affrontato l'ultimo anno del mio percorso universitario. Non solo ho provato a mantenere un rendimento più alto, ma mi sono cimentato nell'approfondimento di discipline più tecniche, e in particolare l'Intelligenza Artificiale, alla quale ho dedicato questo lavoro. Qui devo ringraziare il mio relatore, il professor Armando Stellato, col quale ho instaurato un rapporto che mai, dopo anni e anni di Università, mi sarei aspettato di avere con un professore, e che, nonostante le mie evidenti lacune in Informatica, ha creduto in questo progetto di tesi sostenendomi e supportandomi. Ringrazio inoltre i miei zii per avermi dato l'opportunità di andare a Londra per recuperare del materiale che potesse aiutarmi nella stesura della tesi. Nel mentre, mosso da un senso di insoddisfazione verso la gestione dell'aspetto sociale all'interno dell'Università, ho provato a migliorare le cose prima di andarmene: è da qui che nasce la voglia di partecipare al progetto ASET per dare il benvenuto agli studenti che scelgono la nostra Università come meta Erasmus e accompagnarli durante la loro permanenza qui a Roma, per il quale devo ringraziare tutti i ragazzi dell'associazione: Damiano, Michele, Elisa, Marianna, Valerio, Parth, Giulia, Livio e Leonardo. Ho lavorato duramente per portare a termine questo percorso a modo mio,

non mollando un centimetro, anche quando credevo di non potercela fare. Non so cosa gli altri pensino di me: alle volte sono stato scontroso, scostante, orgoglioso, saccente, pesante o magari semplicemente assente. Ma io sono anche questo e forse è anche grazie a questo che sono riuscito a portare a termine il mio viaggio in una maniera della quale posso andare fiero. Ed è quindi qui che devo ringraziare gli amici che mi hanno sopportato e mi sono stati vicini nel corso degli anni: Lorenzo, amico ormai da una vita, dai banchi delle elementari e del liceo fino ad oggi, tra una partita a carte e una alla PlayStation, sempre presente nel momento del bisogno; Alessio, mio collega dell'Università per tutto questo percorso, tra i numerosi caffè al bar e alle macchinette malefiche, colazioni sulla Tuscolana e gli esami preparati insieme, sempre pronto se serve qualcuno con cui parlare; Anna, con le nostre chat di durata non definita dove un Buongiorno può anche essere seguito da una Buonanotte e dove si parla di tutto quanto succede durante la giornata, di tutte le noie e di tutte le vittorie; Sofia, amica del mare, dello stadio e ora allieva di Inglese, tra prese in giro e momenti di scambio più profondi; ma anche Eleonora, Luca, l'ufficiale della 501esima legione Riccardo, Federica, Alessia, Dario, i miei amici dal liceo con i quali ancora dividiamo bellissime serate al cinema, in pizzeria, a un ristorante di Sushi; Stefano, Alessio, Ginevra, Chiara, Sofia e Giulia direttamente da Cupra Marittima; Thiru, Alessandro, i fratelli Sordi, Andrea, Claudio, Ercole, Emanuele, Gabriele, prima compagni di coro e oggi amici del Fantacoro e delle partite di calcetto più improbabili mai viste; i colleghi dell'Università nuovi e ritrovati, Federico, Andrea, Lorenzo, Gabriele, Jacopo, Federico e tutti i ragazzi delle L; tutti i compagni di botte a KUAI e di sudate in palestra; Geoff, il mio insegnante di Inglese, e tutti quelli che mi hanno dedicato un sorriso all'Università, in Palestra, per strada. L'ultimo ringraziamento non può che andare ai miei genitori. La mattina dello scorso 29 Ottobre, dopo aver ascoltato un podcast che parlava di genitori, riflettendo, ho finalmente capito cosa volesse insegnarmi mio padre. Tutte quelle volte in cui ha festeggiato i miei traguardi; quella volta in cui festeggiò per la mia prima insufficienza a scuola dicendomi "benvenuto tra i normali"; tutte quelle volte in cui mi spronava di dire cosa pensassi, anche quando non mi andava; tutte quelle volte in cui, pur preferendo io seguissi come lui la carriera di cantante, mi ha detto di scegliere il percorso di studi che volevo, portandomi a intraprendere una strada ben lontana dalla sua; tutte quelle volte in cui mi ha detto "Non prendere esempio da me". Tutte quelle volte, sono ormai convinto volesse dirmi "Sentiti libero di diventare chi sei veramente e fallo al massimo". L'ultima frase era spesso accompagnata da un'altra: "Prendi esempio da mamma" che amava tanto. Quella donna che con me ha affrontato tutte le disavventure che ci hanno colpito in questi anni, senza mai mollare un centimetro; con la quale spesso ho discusso, ma che non ha mai smesso di mostrarmi il suo amore di

madre. Non posso chiederle di più e spero di averla resa e di renderla orgogliosa. Non sono né un figlio, né un amico, né uno studente perfetto, ma ho sempre creduto in quello che ho fatto e ho sempre dato me stesso in tutto, provando a dare il meglio. Non so cosa mi riserverà il futuro, ma spero di essere in grado di affrontarlo allo stesso modo. Grazie a tutti!