

UNIVERSITY OF BARI ALDO MORO



DEPARTMENT OF COMPUTER SCIENCE

COMPUTER SCIENCE - ARTIFICIAL INTELLIGENCE

MASTER'S THESIS IN SEMANTIC TECHNOLOGIES AND KNOWLEDGE
GRAPHS

**AUTOMATED PROMPT ENGINEERING USING THE PROMPT
ENGINEERING ONTOLOGY FOR QUERYING LARGE
LANGUAGE MODELS**

Supervisor:

Prof. Claudia d'Amato

Co-supervisor:

Dott. Roberto Barile

Dott. Andrea Nuzzolese

Student:

FRANCIOSO DONATO

ACADEMIC YEAR 2024/2025

Abstract

The use of Large Language Models (LLMs) is becoming increasingly widespread, involving a broad spectrum of users, even those without specific expertise in the field of artificial intelligence. This proliferation is leading to a growing focus on the quality and effectiveness of prompts, which are essential for ensuring consistent, relevant and unambiguous results. In this context, the automatic generation or optimisation of prompts is a strategic area for improving human-model interaction, but current approaches are often limited by high computational costs, poor customisation and difficulties in adapting to specific domains. This thesis explores the integration of ontologies into automatic prompt generation processes, with the aim of increasing their semantic accuracy, adaptability and efficiency. Ontologies, as formal structures for knowledge representation, offer significant potential for guiding the construction of more targeted and contextually relevant prompts, while reducing computational complexity and improving system scalability. The central hypothesis is that the use of an ontology can support more intelligent and personalised interaction with LLMs via automated prompt engineering, promoting a more conscious and effective use of these technologies in different applications.

Contents

List of Figures	iv
List of Codes	iv
List of Prompts	v
1 Introduction	1
1.1 Context	1
1.2 Thesis objective	2
1.3 Structure of the thesis	3
2 Background	4
2.1 Large Language Models	4
2.1.1 LLMs Evolution	5
2.1.2 LLMs Architecture	6
2.1.3 LLMs problems and limitations	8
2.2 Prompt Engineering	10
2.2.1 Prompt techniques	11
2.2.2 Prompt Engineering techniques	13
2.3 Ontologies	14
2.3.1 Ontologies history and evolution	14
2.3.2 Ontologies formal definitions	15
A-box and T-box	18
Relationships	18
Hierarchies of Ontologies	19
2.3.3 Querying Ontologies	20
2.4 Automated prompt engineering: State of the Art	21
3 Research Questions and Adopted Pipeline	24
3.1 Research Questions	24
3.2 Adopted pipeline	24
4 Extending PEO Ontology	26

4.1	PEO ontology overview	26
4.2	Extending PEO with LOT Methodology	27
4.2.1	LOT: Requirements specification	28
4.2.2	LOT: Implementation	31
4.2.3	LOT: Publication	37
5	Automated Prompt Engineering: Design and Implementation	38
5.1	Data selection and preprocessing	39
5.2	Querying Ontology	40
5.3	Generate and test prompts	42
5.3.1	Prompt engineering with ontology information	45
5.3.2	Prompt engineering without ontology information	48
5.3.3	Test the generated prompts	49
5.4	Evaluation approach	50
6	Experimental Evaluations	52
6.1	Quantitative analysis	52
6.2	Evaluation of Prompting Techniques	56
6.3	Qualitative Analysis of Generated Prompts	57
6.4	Results discussion	58
7	Conclusions and Future Works	60
7.1	Conclusions	60
7.2	Future works	60
	Appendices	62
.1	Prompt Examples	63
.1.1	Example 1	63
.1.2	Example 2	64
.1.3	Example 3	66
.1.4	Example 4	67
.1.5	Example 5	68
.2	Comparison with plain prompt	69
	Bibliography	70
	Acknowledgements	75

List of Figures

2.1	Transformer’s architecture	8
2.2	Graphical representation of the ontology described in the example	17
2.3	Ontologies hierarchy	20
3.1	Adopted pipeline	25
4.1	LOT workflow	28
4.2	LOT Ontology requirements specification workflow	29
4.3	LOT Ontology implementation workflow	31
4.4	Main concept conceptualizazion	32
4.5	Entire conceptual schema of the extension	33
4.6	Classification ontology pattern	34
4.7	Parameter ontology pattern	34
4.8	CQ1 query result	35
4.9	CQ2 query result	36
4.10	CQ3 query result	36
4.11	LOT Ontology publication workflow	37
5.1	Civil comments dataset examples	39
5.2	Experimental setup	44
6.1	Numbers of errors in the output format in Gemini	53
6.2	Numbers of errors in the output format in DeepSeek	53
6.3	Metrics comparison for Gemini test with ontology	54
6.4	Metrics comparison for DeepSeek test with ontology	55
6.5	Metrics comparison for test with and without ontology	55
6.6	BoxPlot for prompt technique efficiency evaluation	56
6.7	Prompt techniques suggested by Gemini and DeepSeek to engineering a text classification task	57
1	Plain test metrics	69

List of codes

1	SPARQL Query Example	21
1	SPARQL Query for CQ1	35
2	SPARQL Query for CQ2	35
3	SPARQL Query for CQ3	36
1	SPARQL Query to extract prompting techniques	41
2	SPARQL Query to extract task characteristics	42

List of Prompts

2.1	Example of Few-Shot prompt	11
2.2	Example of role prompting	12
2.3	Example of emotion prompting	12
2.4	Example of chain of thought	12
2.5	Example of CoT prompt with decomposition	13
5.1	Prompt that simulate input request	46
5.2	Prompt to engineer the input request with ontology	47
5.3	Prompt to engineer the input request with no ontology	49
5.4	Prompt to generate prompt techniques for text classification	50

Chapter 1

Introduction

1.1 Context

In recent years, Artificial Intelligence (AI) has advanced significantly, transforming academic research, industrial applications, and daily life. A key driver of this change is the emergence of Large Language Models (LLMs), which can generate and understand natural language with remarkable fluency and contextual awareness. These models have shown a strong ability to generalize across various tasks, from summarization and question answering to dialogue generation.

This shift has created a new way for humans to interact with computers. Instead of writing code or using graphical interfaces, users now engage with LLMs by providing prompts—natural language instructions that inquire the model. This has led to the emergence of prompt engineering, a discipline that focuses on how to structure prompts to influence model behavior, minimize ambiguity, and optimize output quality for specific tasks such as classification, generation, or reasoning. The goal is not only to improve accuracy and relevance, but also to make model responses more controllable and aligned with the user’s intent within well-defined constraints. As a result, prompt engineering has become essential to ensure that LLM outputs are accurate, relevant, and align with user intentions [32]. Many studies confirm that well-designed prompts can greatly improve the quality of responses, making generated content more coherent, relevant, and useful [39].

Despite their seeming simplicity, crafting effective prompts can be challenging. Factors like task wording, desired style, level of detail, and even prompt length can significantly alter how a model behaves. This complexity has sparked increased interest in automated prompt engineering, where systems assist or fully automate the creation, refinement, and adaptation of prompts to boost performance, consistency, and usability [43].

In this evolving landscape, ontologies have re-emerged as valuable tools. Traditionally used to structure domain knowledge through formal semantics, ontologies are now being explored in AI to enhance interpretability, enable reasoning, and support interoperability across systems. For prompt engineering, ontologies can systematically represent and organize various prompting techniques, task categories, and instruction patterns. This ontological structured knowledge can then inform or guide automated systems in generating prompts tailored to specific tasks, user needs, or contextual limitations.

1.2 Thesis objective

In line with our previous discussion, ontologies offer a powerful and flexible tool for structuring and enriching the prompt generation process. This thesis aims to explore if, and to what extent, employing an ontology can improve the quality of automated prompt generation compared to approaches that lack semantic support, such as iterative refinement based on model feedback.

To achieve this, we will use an ontology specifically designed to model knowledge and concepts related to Large Language Models (LLMs) and prompting techniques. Specifically, our focus will be on refining prompting engineering techniques and relate them to task that LLMs can perform. This integration of ontologies into the automatic generation process has led to a new approach: ontology-driven prompt engineering, which aims to combine the flexibility of LLMs with the semantic formalization characteristic of knowledge-based systems. This extension allows us to represent critical aspects that can significantly influence a model's ability to respond correctly to instructions.

In order to automatize prompt engineering we will query the ontology to extract, for each identified task, the most appropriate prompting techniques and the semantic information associated with that task. We will then use these elements to automatically re-engineering the initial prompt provided by the user.

Our goal is to evaluate whether integrating this structured knowledge can lead to qualitatively superior prompt generation compared to what's achievable through automated approaches without ontological support. To validate this hypothesis, we will use specific datasets for prompt quality evaluation, complemented by a qualitative analysis of the correctness, completeness, and relevance of the generated outputs.

1.3 Structure of the thesis

The rest of the thesis is structured as follow, Chapter 2 will present a comprehensive background covering ontologies, LLMs, and prompt engineering. Chapter 3 will detail the thesis’s research questions and provide an overview of the work process. Specifically, it will illustrate how the ontology-driven automatic prompt engineering process was structured and managed, describing the main phases and the related experimental tests conducted. Chapter 4 will be dedicated to the extension of the ontology illustrating every aspect from conceptualization to implementation. Starting from an existing ontology, which conceptualises LLM models and prompt engineering, the section relating to tasks will be extended in particular. Specific characteristics will be associated with each task, which will be useful in supporting the process of automatic prompt engineering. In Chapter 5, the design choices and the methodology adopted for automatic prompt engineering and test setup will be explained. Finally, Chapter 6 will present and compare the results obtained in the previous chapter, while Chapter 7 will be dedicated to the final considerations on the work accomplished and a discussion of possible future works.

Chapter 2

Background

This chapter provides an overview of the core concepts underpinning this thesis. Its goal is to equip the reader with the necessary context to understand the theory, the technologies and methods employed in our study.

We will begin by describing Large Language Models (LLMs), tracing their historical evolution, basic functionalities and their architecture.

Next, we will introduce the sub-field of prompt engineering, analyzing its purpose, practical challenges, and the motivations behind automating the prompt engineering process. We will also provide an overview of the key prompting techniques considered in this study, which will be explored in greater detail in subsequent chapters.

The concluding section of this chapter will focus on ontologies, explaining the fundamental principles of semantic modeling, their operational mechanisms, and their application in LLM-based systems. Specifically, we will introduce the concept of ontology-driven prompt engineering, which involves integrating structured knowledge into the automated prompt generation phases.

Finally, we will present the state of the art for the described technologies, referencing the most relevant scientific contributions that support the design choices made in this research.

2.1 Large Language Models

In the field of computer science, natural language processing (NLP) is one of the most complex and challenging areas, as it requires computational systems to understand, interpret and generate human language. Traditionally, computers have been designed to operate on structured and formalised data, making it difficult to handle the intrinsic properties of natural language, such as semantic ambiguity, polysemy, implicit context and syntactic variations. The introduction and progressive evolution of large language models (LLMs) has contributed

significantly to bridging this gap, enabling machines to interact with language in an increasingly effective and natural way.

2.1.1 LLMs Evolution

The emergence of large language models (LLMs) is the result of a long evolutionary process in the field of natural language processing (NLP). Historically, early approaches to NLP were based on simple methods and fixed rules. A prime example is ELIZA¹, developed by Joseph Weizenbaum in 1966, which simulated human dialogue through pattern matching mechanisms, without any semantic understanding of the content. These systems, although pioneering, were extremely limited in terms of adaptability and linguistic depth.

In the 1980s and 1990s, the introduction of n-gram models marked an important advance: these probabilistic models estimated the probability of a word based on previous words, treating linguistic sequences as Markov processes [10]. Although improving over rule-based methods, n-grams were constrained by very narrow context windows and suffered from poor generalisation capabilities.

A significant breakthrough came, starting from 1997, with the introduction of recurrent neural networks (RNNs), and in particular their improved variants such as Long Short-Term Memory (LSTM) [31], which allowed long-term dependencies in linguistic sequences to be modelled. These models were able to maintain an internal state that evolved over time, providing a contextual representation of the input sequence. However, RNNs were difficult to train on very long sequences and inefficient in terms of parallelisation, making them unsuitable for large-scale training scenarios.

A further conceptual advancement was introduced in 2013 with Word2Vec [26], a model capable of learning distributed representations of words in continuous vector spaces. Unlike sparse representations (one-hot encoding), Word2Vec generated dense vectors capable of capturing semantic relationships between words. This approach laid the foundation for richer and more generalisable linguistic representations, paving the way for further models such as GloVe [28] and ELMo [29], which helped introduce the concept of contextualised embeddings.

At the same time, the adoption of sequence-to-sequence (seq2seq) architectures, models designed to map an input sequence to an output sequence, and encoder-decoder architecture (see 2.1.2 for details), enriched by attention mechanisms, which allow the model to focus on relevant parts of the input, has enabled significant advances in different tasks. The real revolution came

¹[https://it.wikipedia.org/wiki/ELIZA_\(chat_bot\)](https://it.wikipedia.org/wiki/ELIZA_(chat_bot))

in 2017 with the introduction of the Transformer architecture [36]. This model completely eliminated the recurrent structure, replacing it with a self-attention mechanism that allows a model to give different weights to words in a sentence when processing each word and allows the entire sequence to be processed in parallel, dramatically improving training efficiency and the ability to model long-range relationships between tokens. Initially applied to machine translation, the Transformer proved extremely effective in a wide range of NLP tasks [42, 17, 21], forming the architectural basis for all subsequent advanced models.

Starting from this innovation, a new class of models took shape: Large Language Models (LLMs). They retain the Transformer as their architectural core, but extend it on a large scale, both in terms of model size and amount of training data. LLMs have been trained on massive collections of multilingual and multi-domain texts [44], giving rise to models capable of generalising across a wide range of linguistic tasks and exhibiting emerging capabilities such as reasoning that is the act of thinking about something logically and systematically to draw a conclusion or make a decision, although still a known limitation of LLMs, context adaptation, and few-shot learning that is the ability to generalise to new tasks from just a few examples.

2.1.2 LLMs Architecture

LLMs are deep neural networks trained on massive amounts of text, designed to learn distributed representations of language and model its statistical structure. These models are capable not only of understanding the lexical and syntactic context of a text input, but also of generating coherent content, answering questions, translating texts, writing code, and tackling a wide range of linguistic and other tasks. The effectiveness of LLMs stems from two key factors: the availability of web-scale text data and the adoption of advanced deep learning architectures, particularly Transformers.

The Transformer architecture represented a fundamental turning point in modern NLP. Introduced by Vaswani et al. in 2017 [36], it replaced traditional sequential models, such as RNN and LSTM, with an architecture entirely based on attention mechanisms. Its central element is self-attention, which allows the model to calculate the relative importance of each token in a sequence with respect to the others, thus facilitating more effective context modelling.

Unlike RNNs, which process sequences recursively and sequentially, Transformers operate in a completely parallel manner, making training much more efficient and scalable. Furthermore, the ability to learn long-range dependencies

in the text sequence has made these models particularly well suited to language comprehension and generation tasks.

The original Transformer architecture, showed in figure 2.1 is structured according to an encoder-decoder scheme:

- **Encoder:** consisting of a stack of identical blocks, each of which comprises two main sub-layers: a Multi-Head Self-Attention module and a Feed-Forward Neural Network. Each block is accompanied by residual connection and layer normalisation mechanisms, which facilitate gradient propagation and stabilise training. Self-attention allows each token in the sequence to be represented in relation to all the others, and the multi-head mechanism allows multiple types of semantic and syntactic dependencies to be captured simultaneously. To maintain information about word order, positional encoding is added to the input embeddings.
- **Decoder:** is also composed of a stack of blocks similar to those of the encoder, with the addition of a third module, called cross-attention. This mechanism allows the decoder to “consult” the representations generated by the encoder, making it possible to transform from an input sequence to an output sequence. Furthermore, self-attention in the decoder is masked, i.e. it prevents access to future positions in the generated sequence, maintaining the autoregressive nature of the model [34]. This aspect is crucial in text generation tasks.

Starting from the canonical encoder-decoder structure, several architectural variants specialised for specific tasks have been developed over time:

- **Encoder-only:** this variant uses only the encoder stack and is designed for language comprehension tasks, such as classification, entity extraction, and question answering. The model processes the entire sequence bidirectionally, i.e. each token can take into account both previous and subsequent tokens. Well-known examples are BERT (Bidirectional Encoder Representations from Transformers) [6], and some of its evolutions to improve performance or efficiency, such as RoBERTa [23].
- **Decoder-only:** in this case, only the decoder stack is used, with masked attention to ensure the auto-regressivity of the generation. This type of model is the basis of modern LLMs, which specialise in text generation from text prompts. The sequence is generated token by token, each conditioned only on the previous ones. Among the best-known models are

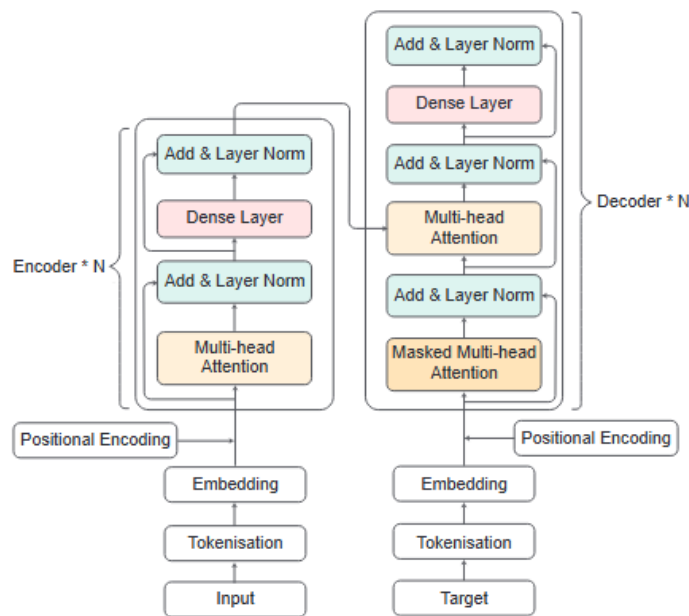


FIGURE 2.1: Transformer's architecture

the various iterations of GPT (Generative Pretrained Transformer) ² and LLaMA (Large Language Model Meta AI) ³.

- **Encoder-Decoder:** represents the original form of the Transformer. It is used in models that need to translate an input into an output of a different nature, such as in machine translation or text summarisation. A relevant example is BART (Bidirectional and Auto-Regressive Transformers) [20].

The evolution of Transformer-based architectures has made it possible to create increasingly powerful and generalist language models capable of tackling a wide range of tasks with high performance. However, in order for these models to fully express their potential, it becomes crucial to be able to guide their behaviour through well-designed inputs: this is where the practice of Prompt Engineering comes in.

2.1.3 LLMs problems and limitations

Although Large Language Models have achieved extraordinary results in numerous NLP tasks, there remain significant critical issues that limit their reliability, scalability, and ethical use. This section analyses the main issues.

- **Hallucinations:** The problem of hallucinations refers to the model generating inaccurate, invented, or unsupported content from the input data.

²<https://openai.com/research/>

³<https://llama.meta.com/docs/>

Such statements are often presented with apparent confidence and stylistic consistency, making it difficult for the user to distinguish between true and false.

For example, an LLM may generate a scientific summary with a non-existent bibliography or make medical diagnoses that are not supported by any source, posing a risk in professional contexts [16, 25].

Among the main causes of hallucinations are:

- Training on noisy or unverified data.
 - Lack of grounding, i.e., direct links to up-to-date external sources.
 - Absence of mechanisms for verifying uncertainty in the output.
 - Limited context: models do not always maintain consistency across very broad contexts.
- **Sensitivity to prompts:** LLMs show high sensitivity to prompt wording: small changes in the input text can produce drastically different responses, with variations in accuracy, completeness, and even consistency. This instability compromises the reliability of the model and complicates the optimisation process through prompt engineering.
 - **Bias in data and results:** Being trained on large amounts of text from the internet, LLMs inherit cultural, racial, gender and political biases present in the data. These biases can be reflected in the results generated, contributing to the reproduction of stereotypes and implicit discrimination [2].
 - **Computational costs:** LLMs require extremely expensive infrastructure to be trained and used. Inference, i.e. the use of the model once trained, can also be costly in terms of latency and resources, especially in real-time scenarios.
 - **Environmental impacts:** The energy consumption required to train and run LLMs at scale has a significant environmental impact. Recent studies estimate that the carbon footprint for training a single model can exceed that produced by hundreds of intercontinental flights [2]. This raises questions about the sustainability of the current paradigm of large-scale model growth.

Among the various limitations described, this thesis project particularly focuses on the problem of prompt sensitivity. The goal is to verify whether the

inclusion of targeted information during the automated prompt re-engineering process can effectively improve the results obtained compared to a generation without such information. Such an approach, in addition to improving prompt quality, would also have a positive impact on costs, as it would reduce the need for more expensive strategies like fine-tuning or prompt optimization itself.

2.2 Prompt Engineering

The evolution of large language models (LLMs) has opened up new perspectives in the interaction between humans and artificial intelligence systems, making it possible to perform complex tasks simply by formulating textual inputs. However, the predictive power of these models does not automatically translate into controllable, reliable behaviour that is consistent with the user's objectives. This is where the concept of Prompt Engineering comes in, i.e. the practice of designing, structuring and optimising the input provided to the model in order to direct its output, minimise errors such as hallucinations and improve effectiveness in specific tasks.

Prompt Engineering is now a fundamental component of the strategic use of LLMs. Unlike conventional programming, prompt engineering acts on parameters that are neither accessible nor modifiable by the user. As a result, the model's behaviour is controlled exclusively through the manipulation of natural input, leveraging the statistical regularities learned during training.

In a formal definition, **prompt engineering** is an iterative process that involves designing, modifying, and optimising prompts in order to obtain more accurate, consistent, or useful responses from a generative language model [32]. This activity may include refining textual content, restructuring input, or adopting different prompting techniques depending on the specific task and capabilities of the model.

Before analysing the main prompting techniques and their use in detail, it is useful to introduce some key definitions that form the conceptual basis of prompt engineering.

The term prompt refers to input provided to a generative model, such as an LLM, with the aim of guiding its output generation [15]. The prompt can take various forms, including text, images, audio, or other types of data, depending on the mode supported by the model. A simple example of a text prompt is:

'Summarise The Iron Giant.'

At the heart of prompt construction there is a prompt template [35], which is a generic parameterised structure that includes one or more variables. These

variables are then replaced with specific values to generate an actual prompt. For example, a template such as:

'Summarise {USER_INPUT}.'

can generate different variants of the prompt by simply changing the content of the `USER_INPUT` variable.

Finally, a prompting technique can be understood as a set of guidelines or a systematic strategy that defines how to structure one or more prompts.

2.2.1 Prompt techniques

This section presents the six main categories of prompt techniques [32], along with some of their most significant derivations [32].

- **Few-shot prompting:** This technique consists of including a series of examples within the prompt that provide the model with explicit context, improving its understanding and generalisation capabilities. The choice of the number of examples, their order, quality and distribution is a crucial component: these factors significantly influence the performance of the LLM in generating output [24]. An example of a few-shot prompt could be:

```
Classify the sentiment of the sentence as Positive,
Negative, or Neutral.
This is awesome! : Negative
This is bad! : Positive
Wow that movie was rad! : Positive
What a horrible show! :
```

PROMPT 2.1: Example of Few-Shot prompt

- **Zero-shot prompting:** Unlike few-shot techniques, zero-shot prompting does not involve the inclusion of explicit examples within the prompt. It can be used independently or in combination with other strategies. Among the most relevant zero-shot techniques is role prompting, which assigns a specific role to the model within the prompt. An example is:

This approach has been shown to improve the accuracy of the response of the LLMs [45]. Another well-known technique is emotion prompting,

```
Act as a doctor. Explain the symptoms of the common cold.
```

PROMPT 2.2: Example of role prompting

which integrates elements related to the human psychological dimension into the prompt [22]. An example is:

```
Translate this sentence. It's an emergency!:  
"The system is offline and we've lost all communication  
with the control center."
```

PROMPT 2.3: Example of emotion prompting

- **Thought generation:** Techniques in this category aim to stimulate the LLM to explain its reasoning process before providing an answer. An example is chain-of-thought prompting (CoT), which encourages the model to produce a sequence of intermediate logical steps, increasing the transparency of the decision-making process. A representative example is:

```
I went to the market and bought 10 apples. I gave 2 apples  
to the neighbour and 2 to the repairman. I then went and  
bought 5 more apples and ate 1. How many apples did I  
have left? Let's think step by step.'
```

PROMPT 2.4: Example of chain of thought

- **Decomposition prompting:** This technique is based on breaking down complex tasks into simpler, more manageable sub-questions. Although some strategies, such as CoT, implicitly induce decomposition, explicitly structuring the problem can further improve the quality of the output as shown in 2.5. A relevant derivation is metacognitive prompting [38], which aims to reflect human metacognitive functioning. This technique consists of a sequence of five steps: clarifying the question, preliminary judgement, evaluating the answer, confirming the decision, and assessing the degree of confidence.

```
Argument:
All mammals are warm-blooded. All whales are mammals.
Therefore, all whales are warm-blooded.

-Let's solve this step by step:
-What is the first premise?
-What is the second premise?
-Can we deduce anything from combining the two premises?
-Based on the deduction, is the conclusion valid?

Now, answer the question: Is the argument logically valid?
```

PROMPT 2.5: Example of CoT prompt with decomposition

- **Ensembling:** Ensembling techniques involve using multiple strategies to solve the same task, combining the results obtained to select the most consistent answer. A prime example is self-consistency [37], based on the assumption that different lines of reasoning can lead to the same correct answer. In this approach, the prompt is executed several times and the final output is chosen by majority vote among the answers obtained.
- **Self-criticism:** Finally, self-criticism techniques aim to induce the LLM to critically evaluate its own outputs, similar to human metacognitive processes. These techniques, often overlapping with metacognitive ones, involve a structured sequence that guides the model through successive stages of reflection and self-correction: understanding the task, producing an initial response, critical evaluation and, finally, possible revision.

2.2.2 Prompt Engineering techniques

The prompt engineering process does not end with the selection of a technique, but also involves the way in which that technique is implemented, refined and adapted to the application context. In this perspective, the main prompt engineering techniques are:

- **Automated prompting** involves using algorithms, often supported by optimisation techniques or other AI models, to automatically generate, modify and select prompts. This approach reduces the need for human intervention and allows for systematic exploration of combinations and variations, making it particularly suitable for highly variable or industrial-scale scenarios [46]. In practice, automated prompting can be achieved through techniques such as reinforcement learning, genetic algorithms,

or simpler approaches based on grid search and empirical evaluation of outputs.

- **Meta prompting** is a paradigm in which the construction of prompts is governed by abstract patterns or generalisable templates. In this context, a prompt is not written as a fixed string, but generated from a parametric structure, called a prompt template, in which slots or variables are provided that are filled dynamically. This allows for the standardisation and scaling of prompt creation, facilitating the replicability of experiments and rapid adaptation to new data or domains. Meta prompting is also useful for automating complex scenarios that require consistency across multiple model calls [40].

Both of these approaches are applied in this thesis work. Specifically, an initial prompt is generated and enriched with information extracted from the ontology. This prompt is then used as input for the automatic re-engineering of a user’s request. The entire process is handled by the LLM without any human intervention, fully accepting the final prompt as generated by the model.

2.3 Ontologies

In computer science, an ontology is typically understood as a formal, shared and explicit representation of a conceptualisation of a specific domain. It consists of a set of entities (classes, individuals) and relationships with rules and axioms that define semantics. Ontologies are constructed using standard languages: the Resource Description Framework (RDF) provides a graph-based data model for describing resources, while the Web Ontology Language (OWL) builds on RDF to provide a formal language for representing rich and complex knowledge about things, groups of things, and relations between them. OWL relies on descriptive logic (Description Logic). In summary, RDF provides the triple structure (subject-predicate-object) to express facts, while OWL adds richer constructs (classes, restrictions, properties) ensuring service for checking logical consistency and the possibility of performing automatic inference.

2.3.1 Ontologies history and evolution

The concept of ontology originates from philosophy but has gained relevance in computer science and artificial intelligence since the 1990s. At that time, researchers began to use ontology to describe knowledge systems useful for training AI systems. Tom Gruber (1993) defined computer ontology as “an artefact

designed for the purpose of modelling knowledge of a given domain”. In the 2000s, the spread of the Semantic Web drove the development of standard formats: in 2001, the W3C established the Web Ontology Working Group and in 2004 published OWL (Web Ontology Language). Since then, ontologies have evolved alongside the semantic web, becoming fundamental for describing formally data domains and enabling semantic integration between heterogeneous systems ⁴.

Today, several ontologies are widely used across the computer science landscape, particularly in domains like biomedical informatics, natural language processing, and linked data. Among the most prominent are WordNet [9], a lexical ontology for English widely adopted in NLP; FOAF (Friend of a Friend) [18], which describes social networks and personal data; Schema.org [14], developed by major search engines to support semantic markup on websites; and SNOMED CT [8] and Gene Ontology [1] in the biomedical domain.

2.3.2 Ontologies formal definitions

An ontology serves as a formal conceptualization of the world, representing a specific aspect of reality. Here’s a breakdown of its key components and characteristics:

- **Conceptualization:** At its core, an ontology is a way of conceptualizing the world, capturing the essential elements and relationships within a domain of interest. It provides a structured framework for understanding and reasoning about the concepts and entities relevant to that domain.
- **Ontological Theory:** An ontology is expressed through a set of axioms that define the structure and semantics of the concepts and relationships within the domain. This ontological theory forms the foundation for representing knowledge and conducting inference within the domain.
- **Formulas and Truth:** Within an ontology, there are axioms that are always considered true, regardless of the particular state of affairs in the world. These formulas represent universal truths or principles that apply across all possible scenarios within the domain.
- **Constraints and Compliance:** An ontology specifies a set of constraints that declare what must necessarily be true in any possible world. These

⁴<https://www.dataversity.net/a-brief-history-of-data-ontology>

constraints define the boundaries and requirements for valid representations within the domain. Any possible world must adhere to these constraints, ensuring that it is compliant with the ontological framework.

A formal ontology comprises three main components:

- A set of concepts, often referred to as classes, represents the fundamental entities or categories within the domain of interest. In formal ontologies, especially those expressed in languages like OWL, these classes can be formally defined using logical axioms that specify the necessary and/or sufficient conditions for class membership. Each class thus defines a group of entities that share common characteristics or properties in a formally verifiable way.
- *Semantic relationships* among the concepts. These relationships define the connections and dependencies between different classes.
- Optionally, a formal ontology may include a *logic level* that enables the inference of new facts from those encoded within the resource. This logic level relies on formal semantics that support automated reasoning through a theorem prover, often using the Tableaux algorithm to perform deductive inference over the ontology. By applying logical reasoning, the ontology can derive new knowledge or conclusions based on the existing conceptual framework.

Building upon this formal structure, an ontology serves as a conceptual formalization of the reality, aiming at representing knowledge in a way that is both machine-interpretable and logically consistent. The following breakdown illustrates the key characteristics that define ontological systems and the mechanisms by which they support structured reasoning.

Formally, an ontology O can be defined as a triple $O = (C, R, A)$, where:

- C represents a set of concepts.
- R represents a set of conceptual relationships, each defined over pairs of concepts from $C \times C$.
- A represents a set of axioms. If $A = \emptyset$, the ontology is not axiomatized.

Additionally, it's noted that C and R induce a graph $G = (V, E)$, where:

- V is the set of concepts C .

- E is the set of edges, defined as $E = \{(c_1, c_2) \in C \times C : \exists S \in R : (c_1, c_2) \in S\}$, meaning there exists a relationship between c_1 and c_2 .
- There is a labeling function $l : C \times C \rightarrow 2^R$, where $l(c_1, c_2) = \{S \in R : (c_1, c_2) \in S\}$. This function assigns a set of relationships to each pair of concepts.

Here is an example:

Consider the formal ontology $O' = (C', R', A')$, where:

- $C' = \{\text{Entity, Object, Person, Mechanic, Car, Engine}\}$.
- $R' = \{\text{is_a, has, repairs}\}$.

The relationships in R' are defined as follows:

- $\text{is_a} = \{(\text{Object, Entity}), (\text{Person, Entity}), (\text{Mechanic, Person}), (\text{Car, Object}), (\text{Engine, Object})\}$.
- $\text{has} = \{(\text{Car, Engine})\}$.
- $\text{repairs} = \{(\text{Mechanic, Car})\}$.

Additionally, the set of axioms A' is given by:

$$A' = \text{Mechanic} \sqsubseteq \exists \text{repairs.Car}$$

This axiom states that for every car, there exists a mechanic who repairs it. We could represent this ontology by means of the graph shown in Figure 2.2.

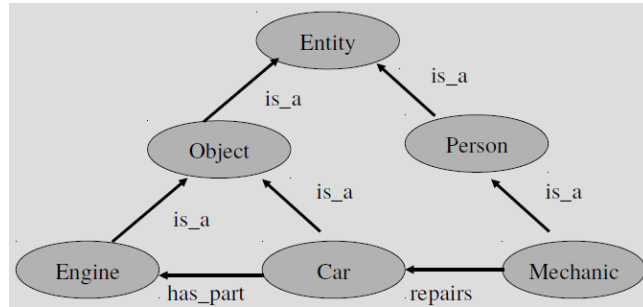


FIGURE 2.2: Graphical representation of the ontology described in the example

A-box and T-box

To make such a structure usable in computational systems, ontologies are composed of a T-Box and an A-Box. The T-Box (Terminological Box) contains the schema or vocabulary—the classes, properties, and axioms defining the domain’s conceptual structure. It captures the intensional knowledge, that is, the meaning of concepts and their interrelations. In contrast, the A-Box (Assertional Box) includes the instances or individuals, together with assertions about their membership in classes and the relationships between them. It constitutes the extensional knowledge of the ontology. This separation enables inference engines to perform different types of reasoning over abstract structures and concrete data.

Relationships

In this structured representation of knowledge, relationships play a crucial role, especially in defining how different categories or entities relate to each other. By defining relationships among categories, in fact, we can organize our conceptual system in a structured manner. These relationships help us understanding how different concepts are related and connected.

There are several fundamental relationships that can exist between entities:

- **Hyponymy**, also known as the “is_a” relationship, describes the relationship between categories where one category is a subtype (specialization) of another. More precisely, hyponymy is the relationship between a more specific term (hyponym) and a more general class (hypernym). For example, “Lion is_a Animal” because a lion is a specific type of animal. Usually, the hyponymy is a transitive relationship defined between categories and should not be confused with the “instance_of” relationship between instances and categories. For example, if “Muffin” is an instance_of “Cat”, it means that “Muffin” belongs to the category of “Cat”.
- **Meronymy** describes the relationship between entities where one entity is a part of another. It’s often expressed as a “part_of” relationship and helps in understanding the structure of entities. For example, in the sentence “The engine is a part of the car”, “engine” is the meronym of “car” because it represents a part of the whole entity, which is the car.
- **Troponymy** is a relationship between verbs and processes. A verb V1 is considered a troponym of another verb V2 if V1 indicates a specific way or manner of performing the action described by V2. For example,

“falling” is a troponym of “moving” because it indicates a specific type of movement.

- The **taxonomic relationship** is a specialized form of hyponymy specifically used to organize concepts into hierarchical structures, typically in a taxonomic or categorical hierarchy.

In addition to predefined relationship types, ontologies also allow for the definition of arbitrary domain-specific relationships, which are essential to accurately model the semantics of a particular context. For example, in an academic domain, a relationship like `isStudentOf` can be introduced to represent the connection between a person and a professor. These custom properties enhance expressiveness and flexibility.

Furthermore, to promote reusability and semantic consistency across applications, ontologies are often organized hierarchically based on their level of abstraction and specificity.

Hierarchies of Ontologies

Ontologies can be organized according to a three-level hierarchy. At the highest level we have *top-level ontologies*, at the middle level are *domain ontologies* and *problem-solving ontologies*, and finally at the lowest level we find *application ontologies*.

- **Top-Level Ontology:** At the top level, there exist a foundational ontology that encompasses universal concepts and relationships applicable across various domains. This foundational ontology provides a standardized framework for organizing knowledge and serves as the basis for more specialized ontologies. Foundational ontologies enforce the reuse of ontological resources by providing a common framework that can be leveraged across different applications and domains. This promotes interoperability and facilitates knowledge sharing and integration.
- **Domain Ontology and Problem-Solving Ontology:** Below the top-level ontology, there exist domain-specific and problem-solving ontologies. Domain ontologies capture concepts, relationships, and constraints within a specific domain or area of knowledge. They provide a more detailed and focused representation of concepts relevant to that domain.

Problem-solving ontologies focus on representing knowledge related to specific problem-solving tasks or activities. They may incorporate domain-specific knowledge as well as strategies, algorithms, and heuristics for solving problems within the domain.

- **Application Ontology:** Finally, there exist application ontologies that are tailored to particular applications or systems. These ontologies build upon the foundational concepts provided by the top-level ontology and the specialized knowledge represented in domain and problem-solving ontologies. Application ontologies are designed to meet the specific needs of a particular application, integrating domain-specific knowledge, problem-solving strategies, and system requirements.

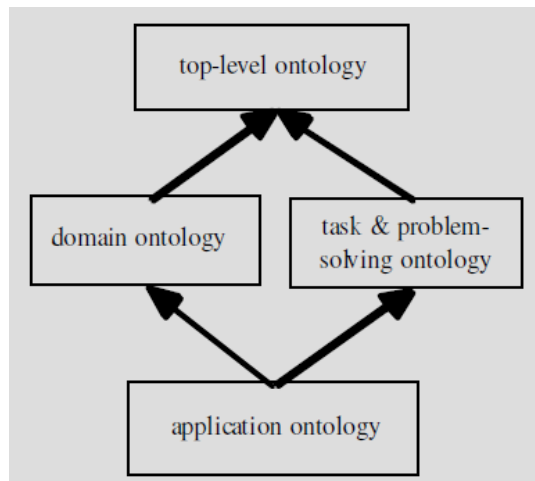


FIGURE 2.3: Ontologies hierarchy

This hierarchical structure allows for the organization of ontologies at different levels of abstraction and specificity, facilitating knowledge representation, reasoning, and interoperability across different domains and applications.

2.3.3 Querying Ontologies

An essential tool for querying ontologies is SPARQL (SPARQL Protocol and RDF Query Language), a standard language recommended by the W3C. SPARQL is specifically designed to work with RDF graphs and is conceptually analogous to SQL in relational databases, but adapted to the graph-based data model encompassing RDF, RDFS and OWL.

SPARQL allows simple and complex queries on nested structures, paths between entities, multiple relationships, filters, aggregations, unions and inferential operations. For example, a basic SPARQL query to retrieve all films directed by a given director could be:

```
1 SELECT ?film
2 WHERE {
3     ?film rdf:type :Movie .
4     ?film :hasDirector :Christopher_Nolan .
5 }
```

LISTED 1: SPARQL Query Example

SPARQL allows to:

- extract structured knowledge from knowledge graphs;
- navigate ontological relationships, such as class hierarchies or inverse properties;

2.4 Automated prompt engineering: State of the Art

In recent years, the automation of prompt engineering and the integration of structured knowledge, particularly ontological knowledge, have received increasing attention in the scientific community. The combination of Large Language Models (LLMs) and KGs enables new opportunities to increase the accuracy, semantic consistency, and adaptability of intelligent systems. This is the context for this thesis, which aims to re-engineer prompts by combining ontologies and automatic prompt engineering.

The problem of automatic prompt engineering has been addressed by several lines of research that aim to optimise prompts to achieve higher performance in LLMs. One of the first contributions in this direction is *AutoPrompt* by Shin et al. [35], which proposes a method for automatically generating prompts from an initial set of seed words. The approach is based on an optimisation process that evaluates the impact of words on model behaviour, adapting them iteratively. Similarly, Yang et al. in *Large language model as optimizer* [40] explores the vision of LLMs as optimisers, highlighting how transformer architectures are capable of adapting the form and content of the prompt according to the objective. On a larger scale, Zhou et al. [46] shows that LLMs are capable of behaving like prompt engineers, generating effective prompts in a similar way to human experts.

A more systematic perspective is offered by Cui et al. in *RePrompt* [4], who presents a survey of automatic prompt optimisation methods using heuristics

and meta-learning and follows this line with RePrompt, which proposes a framework for automatic prompt planning as part of the decision-making process of LLM-based agents.

Evolutionary techniques are at the heart of Xavier’s et al. *GAAPO: Genetic Algorithmic Applied to Prompt Optimization* [33], which uses genetic algorithms to explore the space of optimal prompts. In all these cases, the focus is on the automatic generation or adaptation of prompts, but without a structured use of symbolic or formal knowledge.

In parallel, a line of research has emerged that aims to enhance prompts through the use of ontologies as a source of explicit knowledge. In particular, Tolga et al. [5] introduce the idea of an ontology-driven system that supports the symbolic capture of knowledge in prompts, improving the accuracy and traceability of responses generated by LLMs. This approach suggests that the natural language produced by LLMs can be guided and constrained by semantic structures such as ontological classes, properties, and relationships.

Palagin’s et al. work is similar, proposing *OntoChatGPT* [27], a system that uses structured prompts built from ontologies, with the aim of improving meta-cognitive learning and the personalisation of the model’s responses. In this case, ontology acts as a guiding structure for modelling complex domains and establishing semantic constraints on the content of the prompt.

In the field of cognitive robotics, Ud Din et al. [7] presents an ontology-based prompt tuning model for task and motion planning, highlighting how ontologies can not only represent operational knowledge but also condition the agent’s behaviour through semantically coherent prompts.

Some studies are beginning to explore the combination of these two directions: automation and ontological knowledge. This is the case with Ye’s work [41], which describes the process of prompt engineering to a prompt engineer, i.e., the construction of agents that learn to generate prompts through structured representations of knowledge, exploiting domain patterns and metadata.

In summary, the state of the art shows two converging trajectories: on the one hand, the development of increasingly refined techniques for automatic prompt optimisation; on the other, the use of ontologies to provide structure, semantics and context to prompt content.

However, the systematic combination of these two approaches - i.e., the automatic generation of prompts from elements extracted from an ontology - is still an area under exploration. This is precisely the direction taken by this thesis, which aims to automate the construction of prompts from structured

information from an ontological knowledge base, with the aim of improving the consistency and effectiveness of prompts in LLMs.

Chapter 3

Research Questions and Adopted Pipeline

After introducing the technologies involved in this thesis, this chapter will describe the research questions and the adopted pipeline for achieving the stated objective. The next chapter will be dedicated to a description in more details of the solution of design and implementation used and how they were employed within the process of the ontology-based prompt engineering.

3.1 Research Questions

The aim of this thesis is to explore whether the use of an ontology, that describes concepts related to prompt engineering and Large Language Models (LLMs) [13], can support and improve the process of automatic prompt engineering from a common user request.

For this purpose, the research questions to be answered are as follows:

RQ1: *Does the ontology improve the effectiveness of automatic prompt engineering compared to LLM?*

RQ2: *Does the ontology improve the efficiency in the automatic prompt engineering process?*

3.2 Adopted pipeline

The diagram in Figure 3.1 illustrates the adopted pipeline. This pipeline is structured as follow: starting from a generic task, the **ontology is queried** to extract the necessary information, in particular the appropriate prompting techniques and the characteristics associated with the task. The information obtained is then fed into the **automated prompt engineering** process, which generates a re-engineered prompt based on the initial request.

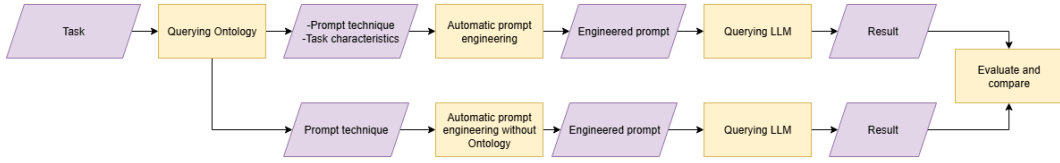


FIGURE 3.1: Adopted pipeline

At the same time, the same process is performed **without the ontology**, leaving the re-engineering of the initial request without the ontology support completely to the LLM. The diagram in Figure 3.1 illustrates how the information extracted from the ontology is used; this step is necessary in particular in this work to ensure consistency in the application of the same prompting techniques in the two approaches, thus allowing for a fair comparison.

Finally, the generated prompts are fed into the LLM, which produces the subsequent results. These results are then evaluated and compared in order to verify the effectiveness of the guided approach compared to the LLM-based one.

Before describing the entire process in detail, it is necessary to introduce the reference ontology. Specifically, we require an ontology that conceptualizes aspects related to prompt engineering. Although there is a state-of-the-art ontology addressing this domain, it lacks some of the components needed for our purposes. In particular, the specific characteristics of the tasks, which are fundamental for guiding the process of automatic engineering of the prompts, are missing. For this reason, the next chapter will illustrate in detail the process of extending the ontology, which is a preliminary and indispensable step before applying the methodology described above.

Chapter 4

Extending PEO Ontology

In this chapter we will present the extension of the PEO ontology that will be used in the thesis work. First, we describe the original ontology, followed by the motivation for its extension and the design and implementation of the extended PEO. We then briefly illustrate how it was used to extract the necessary information for performing automated prompt engineering.

4.1 PEO ontology overview

The PEO Ontology [13], available online ¹, is designed to describe the fundamental concepts related to Large Language Models (LLMs) and prompt engineering in a structured way, emphasizing the semantic relationships between the entities involved.

Regarding language models, each LLM is described according to four main dimensions:

- **Type:** the model’s typology.
- **Organization:** the organization that developed it.
- **Base Model:** the foundational model it is based on.
- **Capability:** the model’s capabilities and functionalities.

These dimensions provide a comprehensive representation from both an architectural and organizational perspective.

Prompt engineering, on the other hand, is modeled through the following concepts:

- **Prompt:** the structured request sent to the model, provided as input in the chat context for an LLM, followed by a response. The prompt can be generated using a prompting technique.

¹https://bioportal.bioontology.org/ontologies/PEO_ONTOLOGY

- **Prompting Technique:** the technique used to formulate the prompt.
- **Chat:** the conversational interaction with the LLM.
- **Response:** the response generated by the model.

These concepts allow for the description of the entire prompt lifecycle, from its creation to the generation of the response by the model.

A further central element in the ontology is the concept of **Task**, which represents the type of problem or activity that the LLM can address through the use of a structured prompt according to an appropriate prompting technique. It is modeled as a single standalone concept, which is not sufficient to meet our objective.

The Task concept is at the heart of the extension developed for this thesis project. The objective is to improve the quality of responses generated by the model, starting from a user request that is classified into a specific task and subsequently re-engineered with the aid of an appropriate prompting technique.

The extension therefore involved modeling the peculiar characteristics of tasks, with particular attention to aspects that directly influence the prompt's effectiveness in guiding the model toward a coherent, correct, and informative response.

The methodology adopted for the ontology's extension and its practical application will be described in detail in the subsequent sections.

4.2 Extending PEO with LOT Methodology

The methodology used to extend the PEO is the Linked Open Terms (LOT) [30]. LOT methodology is a lightweight industrial-oriented methodology for ontology engineering. The methodology is compatible with software engineering methodologies; indeed, it is inspired by the agile methodology. This aspect, combined with a rich documentation, available online ², is the reason why this methodology was chosen to the detriment of others.

Figure 4.1 describes the ontology development lifecycle with four stages: requirements specification, implementation, publication, and maintenance, involving collaboration between ontologists, developers, users, and experts. Each stage outputs artifacts (e.g., ORSD, ontology, online ontology) that flow into subsequent phases for iterative refinement and usage.

Each of these phases will be explained together with the application of the thesis project.

²<https://lot.linkeddata.es>

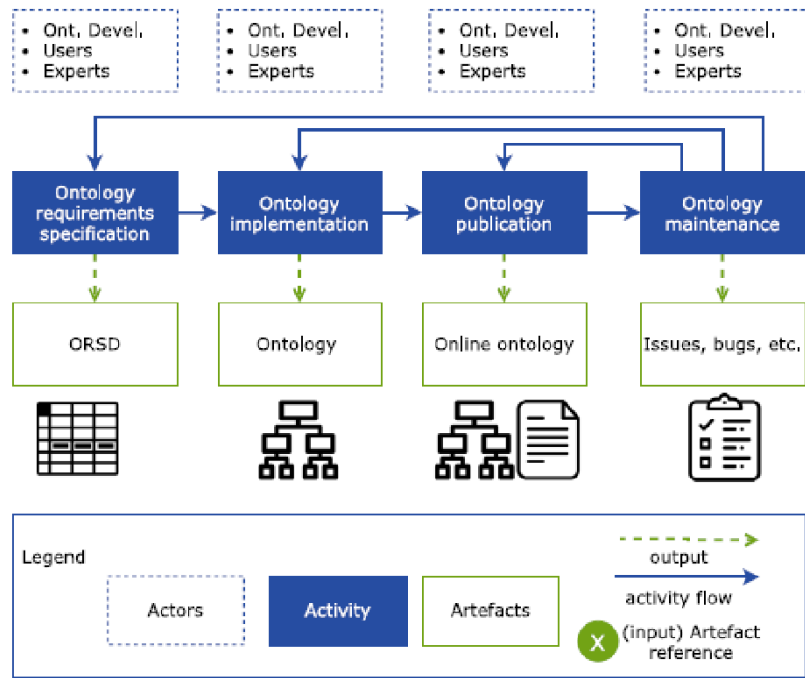


FIGURE 4.1: LOT workflow

4.2.1 LOT: Requirements specification

The aim of the requirements specification process is to state why the ontology is being built and to identify and define the requirements the ontology should fulfil. As shown in Figure 4.2, the LOT methodology is divided into different phases. At the end of each phase, a document is produced containing the analyzed aspect of the specifications.

The first phase is the **use case specification**. This phase defines how the ontology will be used within the application domain. Two main use cases, distinct from those originally envisioned in PEO, were identified to guide its extension:

- **Use case 1**
 - **Name:** Optimizing prompt
 - **Description:** The Prompt Engineer or an LLM directly will use the extended PEO_extension ontology to improve prompts, either manually or automatically, to optimize the quality of the LLM's responses.
 - **Actors:** Prompt engineer, LLM
 - **Flow:** The Prompt Engineer queries the ontology to obtain relevant information related to the task (techniques, characteristics) and uses

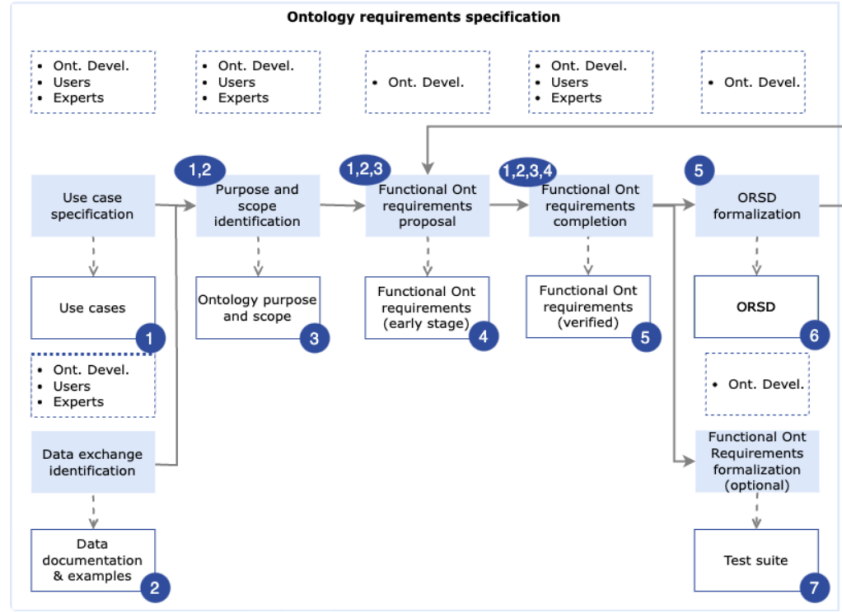


FIGURE 4.2: LOT Ontology requirements specification workflow

it to regenerate the prompt. Alternatively, the same process is performed autonomously by an LLM.

• Use Case 2

- **Name:** Dynamic prompt
- **Description:** Software interacts with the user to gather, through ontology queries, the necessary information for constructing an optimized prompt.
- **Actors:** User, LLM, Software
- **Flow:** The software queries the ontology to identify the relevant task parameters and presents them to the user in the form of questions; the answers are used to build a final prompt to be provided to the LLM.

It should be noted that, although both use cases have been formalised for the sake of completeness, this thesis focuses exclusively on the first use case: *Optimizing prompt*.

The second phase is named **data exchange identification**. This phase involves identifying the information sources and relevant documents for domain modeling.

Within the context of the LOT methodology, no specific type of source is imposed, so scientific articles, technical manuals, API documentation, web resources, or empirical experience can be used.

For this thesis, the ontology extension will focus exclusively on the text classification task. Consequently, the reference documentation used for modeling data and ontological concepts was selected solely in relation to the specific aspects of text classification.

Any future extensions of the ontology to include other tasks, present in PEO, will require a new phase of collecting and analyzing specific sources and documentation for each task. Only in this way it will be possible to ensure an accurate and coherent ontological representation that meets the needs of the new domains. Concerning text classification, the following types of sources were consulted:

- The Benefits of Label-Description Training for Zero-Shot Text Classification [12]
- Optimizing Prompts for Text Classification in Amazon Bedrock: From Manual to Automatic Approaches³
- Text Classification Algorithms: A Survey [19]

Next, we find the phase of **purpose and scope identification**. This phase defines the purpose and scope of the extended ontology.

In this specific case, the purpose of the PEO_extension is to:

- Improve the conceptual representation of NLP-related classification task characteristics.
- Use these characteristics to automatically optimize prompt engineering.
- Enhance the quality and effectiveness of responses provided by LLMs.
- Reduce misunderstandings between the LLM and the user caused by imprecisely formulated prompts.

The extension aims to bridge the gap between the natural formulation of a request and its translation into an optimal prompt, reducing both the effort required from the user and the need for iterative refinement via LLMs.

The next phase is the the **functional ontological requirements** that are formalized through Competency Questions (CQs), which are key questions that the ontology must be able to answer to meet the use case objectives.

CQs define the expected conceptual coverage of the ontology and guide the subsequent modeling phase.

The Competency Questions identified for this extension are:

³Optimizing Prompts for Text Classification in Amazon Bedrock: From Manual to Automatic Approaches

- **CQ1:** What is a characteristic?
- **CQ2:** What are the characteristics of a task X ?
- **CQ3:** Which task needs a characteristics Y ?

The final phase consists of drafting the **ORSD (Ontology Requirements Specification Document)**, which represents a formalization of all the information gathered in the preceding phases. The document serves as a guide for the subsequent ontological modeling phase and ensures traceability between objectives, requirements, and implementation.

4.2.2 LOT: Implementation

The purpose of this phase is to cover the entire ontology development cycle, starting from conceptualization, moving through the reuse and encoding phases, and concluding with evaluation. This process is summarized in Figure 4.3.

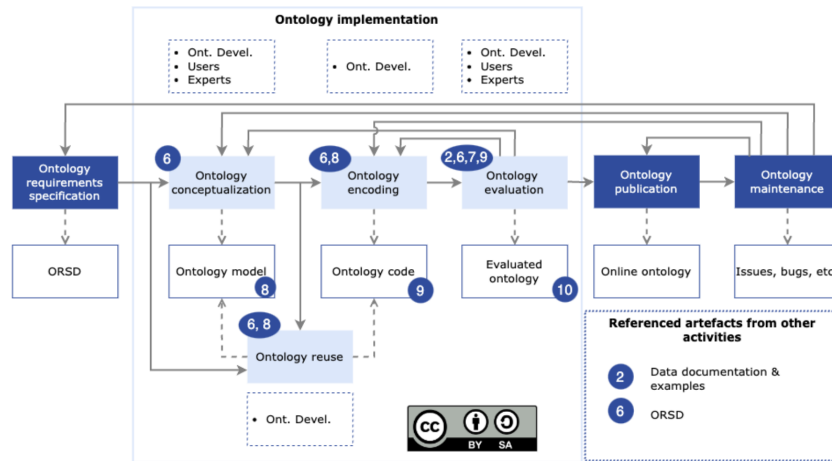


FIGURE 4.3: LOT Ontology implementation workflow

The **conceptualization** phase focuses on identifying the fundamental concepts of the domain and the relationships between them. A top-down strategy was adopted, initially modeling the more general concepts and then progressively specializing them.

In particular, two main concepts were identified:

- **View:** grouping of characteristics into a view based on the nature of the task.
- **Characteristic:** represents specific, abstract or concrete information that describes a relevant aspect of a task.

These two concepts are linked by the following relations:

- **hasView**: a task is described by one or more views based on the nature of the task.
- **hasCharacteristic**: a view groups one or more characteristics together.

The schema is showed in Figure 4.4

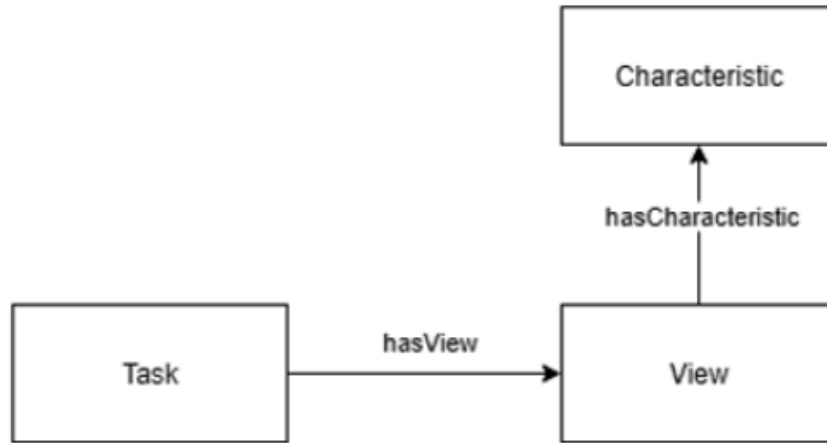


FIGURE 4.4: Main concept conceptualization

Subsequently, specializations of these concepts have been introduced, focusing on the specific aspects of text classification tasks. The following concepts, representing specialized characteristics, have been created:

- **Format**: format of the text (e.g simple sentences, document, structured text, ecc.)
- **Language**: language of the text
- **LabelDefinition**: explanation of the label
- **LabelType**: definition of the type of the label (e.g. binary, multiclass, ecc)
- **OutputFormat**: the output format for a better usage.

These concepts are related to two subclasses of Characteristic, respectively:

- **ClassificationView**: grouping of all characteristics related to classification tasks.
- **TextView**: grouping of all characteristics related to text tasks.

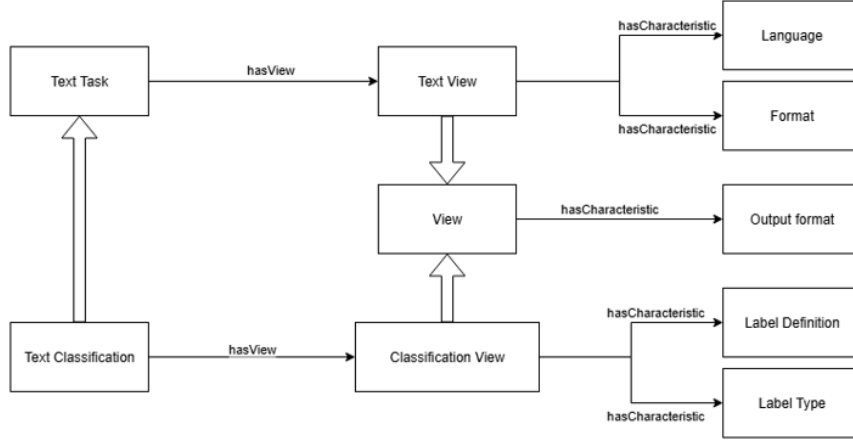


FIGURE 4.5: Entire conceptual schema of the extension

The entire conceptual schema is illustrated in Figure 4.5.

The ontology is designed to be extensible: that is for supporting other tasks (e.g., question answering, summarization) be desired, it will be sufficient to follow the same ontology design patterns adopted in this phase, extending the concepts and relationships in a manner consistent with the LOT methodology.

Regarding the **reuse** phase, no external ontologies were imported in this extension, as the work directly builds upon PEO. To the best of our knowledge, no other ontologies modeling LLMs, prompting techniques, and related tasks are currently available in the state of the art without introducing concepts that depend on external vocabularies.

Before proceeding with formal implementation, two fundamental ontological patterns for modeling were adopted:

- **Classification pattern:** The intent is to represent the relations between concepts (roles, task, parameters) and entities (person, events, values), which concepts can be assigned to. The schema is represented in Figure 4.6
- **Parameters pattern:** The intent is to represent parameters to be used for a certain concept. The schema is represented in Figure 4.7

These patterns guided the conceptual structure and modeling choices during the encoding phase.

The **encoding** phase translated the conceptual structure into a formal representation compliant with OWL, using the Protégé editor ⁴. The resulting ontology is an extension of the original PEO and maintains the same modular structure. To test the ontology’s functionality and support subsequent queries, a new individual called `emotion_classification2` was instantiated, representing a

⁴<https://protege.stanford.edu/>

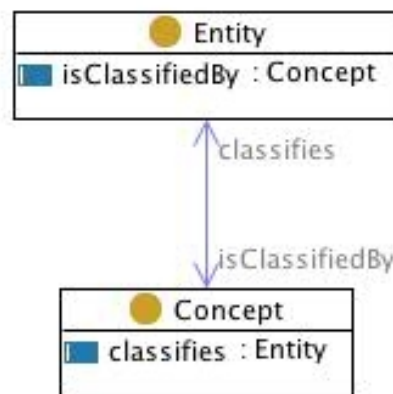


FIGURE 4.6: Classification ontology pattern

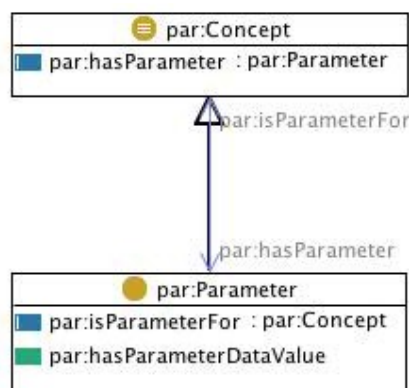


FIGURE 4.7: Parameter ontology pattern

text classification task. From this individual, the ontology was populated with all necessary characteristics, ready to be queried via SPARQL.

The last phase of LOT implementation phase is the ontology evaluation, which was based on two main activities: consistency checks and verification of competency questions (CQ).

The **consistency check** is a reasoning operation provided by the OWL, available through the Protégé development environment. This type of verification aims to assess whether the ABox is consistent with the TBox. To perform this operation, the HermiT reasoner, integrated into Protégé, was used. After starting the reasoner and a complete analysis of the knowledge base, no consistency issues were detected, confirming the formal correctness of the ontology.

Next, the ontology's adherence to functional requirements was verified by **executing the Competency Questions**, previously defined in the specification phase. Specifically, for each CQ, a corresponding SPARQL query was formulated to interrogate the ontology and verify its ability to provide the requested information. List of CQ:

- **CQ1** *"What is a characteristic?"* is translated with query showed in code 1 and the result is showed in figure 4.8.

```

1  PREFIX peo: <https://w3id.org/peo#>
2  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns
   #>
3  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4
5  SELECT DISTINCT ?property ?value
6  WHERE {
7      peo:Characteristic ?property ?value.
8  }
9

```

LISTED 1: SPARQL Query for CQ1

```

Property: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
Value:    http://www.w3.org/2002/07/owl#Class
-----
Property: http://www.w3.org/2002/07/owl#disjointWith
Value:    https://w3id.org/peo#Classification
-----
Property: http://www.w3.org/2000/01/rdf-schema#comment
Value:    Represents specific, abstract or concrete information that describes a relevant aspect of a task.
-----
Property: http://www.w3.org/2000/01/rdf-schema#label
Value:    Characteristic
-----

```

FIGURE 4.8: CQ1 query result

- **CQ2** *"What are the characteristics of a task X ?"* is translated with query showed in code 2 and the result is showed in figure 4.9. In this case, it is taken as an example EmotionClassification.

```

1  PREFIX peo: <https://w3id.org/peo#>
2  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns
   #>
3  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4
5  SELECT DISTINCT ?characteristic ?charClass
6  WHERE {
7      ?task rdf:type peo:EmotionClassification.
8      ?task peo:hasView ?view .
9      ?view peo:hasCharacteristic ?characteristic .
10     ?characteristic rdf:type ?charClass
11 }
12

```

LISTED 2: SPARQL Query for CQ2

```

-----
Characteristic: https://w3id.org/peo#binary
Characteristic Class: https://w3id.org/peo#Label_type
-----
Characteristic: https://w3id.org/peo#toxicity
Characteristic Class: https://w3id.org/peo#Label_definition
-----
Characteristic: https://w3id.org/peo#english
Characteristic Class: https://w3id.org/peo#Language
-----
Characteristic: https://w3id.org/peo#sentences
Characteristic Class: https://w3id.org/peo#Format
-----

```

FIGURE 4.9: CQ2 query result

- **CQ3** "Which task needs a characteristics *Y* ?" is translated with query showed in code 3 and the result is showed in figure 4.10. In this case, it is taken as an example *Language* as a characteristic.

```

1  PREFIX peo: <https://w3id.org/peo#>
2  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4
5  SELECT DISTINCT ?task ?taskClass
6  WHERE {
7    ?characteristic rdf:type peo:Language.
8    ?class rdfs:subClassOf* peo:View.
9    ?view rdf:type ?class.
10   ?view peo:hasCharacteristic ?characteristic.
11   ?task peo:hasView ?view.
12   ?task rdf:type ?taskClass.
13 }
14

```

LISTED 3: SPARQL Query for CQ3

```

-----
Task: https://w3id.org/peo#emotion_classification_2
Task Class: https://w3id.org/peo#EmotionClassification
-----

```

FIGURE 4.10: CQ3 query result

The result obtained from each query was then compared with the expected result to confirm that the modeled ontology was indeed capable of satisfying the requests foreseen by the identified use cases.

4.2.3 LOT: Publication

Before the publication phase, showed in 4.11, the automatic documentation of the ontology was generated using the OWLDoc plugin available within the Protégé editor. This tool allows for the creation of a navigable HTML representation of the concepts, classes, properties, and instances defined in the ontology, facilitating its understanding and potential reuse by third parties.

Subsequently, both the ontology in OWL format and the generated documentation were published on GitHub ⁵. This repository not only allows public access to the ontology for reuse and integration into other projects but also represents a first step towards the collaborative **maintenance** of the ontology. Through the platform, it will be possible to open issues to report bugs, suggest improvements, or propose extensions, in line with the best practices suggested by the LOT methodology.

The open and versioned approach also allows for tracking changes, facilitating the evolutionary management of the ontology over time.

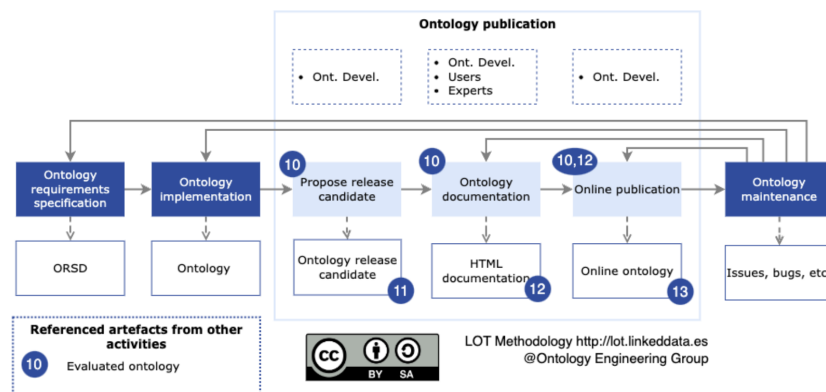


FIGURE 4.11: LOT Ontology publication workflow

⁵https://github.com/FranciosoDonato/PEO_extension-Ontology

Chapter 5

Automated Prompt Engineering: Design and Implementation

This chapter will describe in detail the operational core of this thesis work, the **automated prompt engineering**. The entire experimental process will be illustrated, starting from querying the ontology for extracting relevant information, continuing with the selection of models and methodology for automatic prompt engineering, and concluding with the testing phase of the generated prompts. Finally, the evaluation of the results will be presented, where the method adopted for analyzing the obtained performance will be described. The detailed results and related discussions will be illustrated in the next chapter.

As already highlighted in the previous chapter, although the proposed approach is conceptually applicable to any task modeled in the ontology, for the purpose of this experimentation, we focused on the **text classification** task. For this reason, a specific benchmark dataset was selected, whose examples were treated as simulations of user requests directed to an LLM to perform text classification activities.

This choice was motivated by two main considerations:

- Hardware and economic limitations related to the intensive use of LLMs on a large scale;
- The need to conduct statistically significant evaluations, based on a consistent number of examples.

The next section will introduce the dataset used, illustrating its main characteristics and describing the necessary preparation and manipulation operations for its integration into the experimental process.

5.1 Data selection and preprocessing

For the entire automatic prompt engineering process, a text classification dataset called "Civil Comments" ¹ was used. This dataset contains a collection of sentences extracted from the Civil Comments platform, a site that collected online discussions and, upon its closure, made its data publicly available for research purposes.

Each comment in the dataset is associated with a **toxicity** value ranging between 0 and 1, where toxicity is defined as *any rude, disrespectful, or unreasonable content that could make someone leave the conversation*. [3] In addition to the main toxicity value, the dataset also includes other more specific classifications, which will not be considered in this work.

An example of some sentences contained in the dataset is shown in Figure 3.1.

haha you guys are a bunch of losers.	0.893617
ur a sh*tty comment.	0.666667
hahahahahahahhha suck it.	0.457627
FFFFUUUUUUUUUUUUUUUU	0
The ranchers seem motivated by mostly by greed; no one...	0

FIGURE 5.1: Civil comments dataset examples

The choice to use Civil Comments is motivated by the fact that it represents a concrete and realistic case of a problem in Natural Language Processing (NLP): the automatic classification of comment content, with the aim of detecting its toxicity level. This scenario reflects a typical case of online interactions and a potential real user need, making the task particularly suitable for the objectives of this thesis.

Before integrating the dataset into the automatic prompt engineering process, some preprocessing and modeling operations were performed:

- **Binarization of toxicity classification:** The continuous toxicity value (0-1) was transformed into a binary classification:
 - 0 → non-toxic comment
 - 1 → toxic comment

¹https://huggingface.co/datasets/google/civil_comments

This choice was made based on the documentation provided by Hugging Face, according to which a toxicity value greater than 0.5 can be considered indicative of toxicity. Binarization simplifies the task and clarifies the evaluation of results.

- **Data sampling:** For computational reasons and to manage the costs associated with using LLM APIs, a sample of 100 sentences was selected randomly, with a distribution of:
 - 60% non-toxic comments
 - 40% toxic comments

The number of 100 examples was chosen as it represents a good compromise between technical scalability (avoiding blocking by API providers) and the possibility of performing statistically significant analyses on the obtained results.

All analyses and experiments conducted throughout the work are based exclusively on this dataset, unless otherwise specified.

5.2 Querying Ontology

In this section, all the necessary elements for querying the ontology will be presented, with particular attention to the **SPARQL queries** used and the results obtained. The queries have been designed to extract, starting from a specific task identified in the user request, the pertinent information contained within the ontology. The results of the queries will then be used in subsequent phases to engineering the prompt thus obtaining the final prompt.

The first SPARQL query performed, showed in code 1, involved extracting the prompting techniques associated with the text classification task, as defined within the ontology.

This operation has a dual objective:

- To identify the most suitable prompting techniques for automatic prompt engineering, according to the conceptual structure provided by the ontology;
- To compare the techniques returned by querying the ontology (PEO) with those suggested by querying an LLM would autonomously adopt to perform the same task, in the absence of ontological knowledge.

The result of this query therefore represents a fundamental starting point for the entire experimental process, as it provides a set of techniques that will be used to automatically engineering the prompt, in order to improve its effectiveness and precision.

```

1 PREFIX peo: <https://w3id.org/peo#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4 SELECT ?technique ?label ?comment WHERE {
5     ?subclass rdfs:label ?label .
6     ?subclass rdfs:comment ?comment .
7     ?technique rdf:type ?subclass .
8     ?subclass rdfs:subClassOf* peo:PromptingTechnique .
9     ?technique peo:solves ?task .
10    ?task rdf:type peo:Task .
11 }
```

LISTED 1: SPARQL Query to extract prompting techniques

The query represent the general question to be possibly customised to the specific task of interest. Specifically, for the case of the thesis, "Task" in line 10 has been replaced with "EmotionClassification", which describes the text classification task within the ontology.

The second SPARQL query, showed in code 2, is the core of this thesis. It lets us extract all semantic features associated with a specific task (in this case, text classification) that are useful for improving the prompt.

This phase is crucial because it allows us to concretely evaluate the ontology's contribution to prompt engineering. In other words, it's precisely through the results of this query that we can answer the main research question: does the use of structured information, like that offered by an ontology, improve the quality of automatically generated prompts?

If the experiment demonstrates a real improvement in the quality of the LLM's responses, we will be able to confirm that:

- The use of explicit and formalized knowledge effectively supports the prompt engineering process.
- It is possible to reduce reliance on the LLM's internal optimization, which typically involves automatically re-engineering the generated prompt by iteratively modifying certain aspects until the desired outcome is achieved (as currently happens with purely neural techniques [4]).
- Computational costs can be contained.

- The degree of prompt personalization can be increased thanks to the semantic modeling provided by an ontology.

In this context, the ontology doesn't just act as passive support; it becomes a truly active component in the automatic prompt engineering process.

```

1 PREFIX peo: <https://w3id.org/peo#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4 SELECT DISTINCT ?label ?comment
5 WHERE {
6     ?subClass rdfs:label ?label.
7     ?subClass rdfs:comment ?comment .
8     ?char rdf:type ?subClass .
9     ?subClass rdfs:subClassOf* peo:Characteristic .
10
11     ?view peo:hasCharacteristic ?char.
12     ?view rdf:type ?viewClass.
13     ?viewClass rdfs:subClassOf* peo:View.
14
15     ?task peo:hasView ?view.
16     ?task rdf:type peo:Task.
17 }
```

LISTED 2: SPARQL Query to extract task characteristics

Also in code 2 "Task" in line 16 has been replaced with "EmotionClassification".

Once the ontology query is complete and the structured information related to the text classification task is obtained, we can proceed with **automatic prompt engineering**.

5.3 Generate and test prompts

In this section, the prompt engineering and testing process will be shown. This phase is performed in two distinct and parallel modes:

- **With ontological support:** Using the previously extracted semantic information (prompting techniques and task characteristics), the prompt is engineered with the aim of making it clearer, more specific, and more coherent with respect to the user's request.
- **Without ontological support:** The prompt is engineered solely from the user's request, relying exclusively on the capabilities of the language model (LLM), without any semantic enrichment derived from ontologies.

Before diving into the details of the automatic prompt engineering process, it's helpful to briefly describe the Large Language Models (LLMs) used in this study.

For the engineering and testing phase, two models were employed:

- **Gemini 2.0 Flash** ²: Developed by Google DeepMind, Gemini is a family of advanced multimodal models. It is based on Encoder-Decoder architecture. The Flash version is optimized for low latency and reduced costs while still maintaining good performance in natural language generation.
- **DeepSeek V3** ³: This is an open-source model developed by DeepSeek, designed to offer high performance across various NLP tasks. It is based on a Decoder only architecture. Version V3 strikes a good balance between response quality and computational costs, making it suitable for controlled experimental scenarios.

Both LLM models utilize a Mixture of Experts (MoE) based approach [11]. MoE is an architectural technique that allows increasing the total number of model parameters without proportionally increasing the computational cost per prediction. The basic principle is that, for each input token, only a small group of "experts" (specialized sub-networks within the model) is activated. These experts are chosen by a gating network that dynamically decides which ones to use based on the context. In this way, even if the model has billions of parameters, only a small part is effectively used for each prediction, maintaining high efficiency. This architecture lets you combine response quality (thanks to expert specialization) with scalability and computational efficiency (thanks to partial activation).

After that it was necessary to understand how to use these models. There are two primary ways to access artificial intelligence models:

- **Local execution**: This involves downloading the model weights and running them directly on a local machine.
- **API access**: This allows interaction with the model running on remote servers (cloud) via HTTP requests.

Local execution, while ensuring greater control and performance optimization, requires advanced hardware, especially for large models. High-performance GPUs and a significant amount of memory are necessary, resources that were not available in this thesis's experimental environment.

²<https://gemini.google/assistant/?hl=it>

³<https://www.deepseek.com/en>

For this reason, we chose to use APIs. Specifically:

- **Gemini 2.0 Flash** was selected because it offers a free daily request limit, allowing continuous, cost-free use for the entire duration of the experiments.
- **DeepSeek V3** was used through the *Together AI*⁴ provider, which offers a free welcome credit sufficient to support all necessary requests throughout the project.

Using these two solutions allowed us to compare the responses provided by different models, testing the effectiveness of automatic prompt engineering, both with and without ontological support, across multiple LLM architectures.

This dual generation allows for the creation of a comparative basis between the two approaches. The prompts produced will then be used to query the LLM and generate the corresponding responses, which will be evaluated and compared in subsequent phases.

After selecting and configuring the LLMs to be used, we proceeded with defining the **experimental setup** necessary for evaluating the effects of the ontology on automatic prompt engineering. A schematic representation of all experiments conducted, including execution times, is shown in Figure 5.2. For each test, both the time required for automatic prompt engineering, as described in sections 5.3.1 and 5.3.2, and the actual test execution time, as illustrated in section 5.3.3, were considered.

Test number (Generation and test)	☑ Ontology	☐ Characteristic	☐ Prompt technique	☐ Model	# Time (seconds)
Test 1	☑	Four	Few-shot	Gemini 2.0	1200
Test 2	☑	Four	CoT	Gemini 2.0	1300
Test 3	☑	Three	Few-shot	Gemini 2.0	1400
Test 4	☑	Three	CoT	Gemini 2.0	1840
Test 5	☑	Two	Few-shot	Gemini 2.0	1200
Test 6	☑	Two	CoT	Gemini 2.0	1200
Test 7	☐		Few-shot	Gemini 2.0	1200
Test 8	☐		CoT	Gemini 2.0	1300
Test 9	☑	Four	Few-shot	DeepSeek V3	2000
Test 10	☑	Four	CoT	DeepSeek V3	1700
Test 11	☑	Three	Few-shot	DeepSeek V3	1800
Test 12	☑	Three	CoT	DeepSeek V3	1500
Test 13	☑	Two	Few-shot	DeepSeek V3	2600
Test 14	☑	Two	CoT	DeepSeek V3	2800
Test 15	☐		Few-shot	DeepSeek V3	1600
Test 16	☐		CoT	DeepSeek V3	2000

FIGURE 5.2: Experimental setup

Although the ontology provides a wide range of prompting techniques, for reasons related to both computational efficiency and the practical widespread

⁴<https://www.together.ai>

use of the techniques themselves, the experiment focused on two of the most consolidated and widely used:

- **Few-Shot Prompting:** A technique that involves providing the LLM with a few examples (guide examples) of the task to be performed within the prompt.
- **Chain-of-Thought Prompting:** A technique that stimulates the LLM's explicit reasoning, guiding it through a sequence of intermediate steps.

For each example in the dataset, the following operations were performed:

- Automatic prompt engineering with the **Few-Shot technique**, using **information extracted from the ontology**.
- Automatic prompt engineering with the **Chain-of-Thought technique**, using **information extracted from the ontology**.
- Automatic prompt engineering of the **Few-Shot prompt without ontological support**, relying solely on the LLM's native capability.
- Automatic prompt engineering of the **Chain-of-Thought prompt without ontological support**, again without the aid of structured information.

These four types of prompts were generated for both LLM models (Gemini 2.0 Flash and DeepSeek V3), allowing for a comparative analysis across two different architectures.

All generated prompts were saved in a specific format, in order to be able to:

- Test them on the LLMs to obtain corresponding responses.
- Consult them during the qualitative and quantitative evaluation phase.
- Reuse them for future analyses or additional tests.

5.3.1 Prompt engineering with ontology information

To ensure an automatic and repeatable prompt engineering process, it was necessary to design ad hoc prompts that could be reused generically and independently of the specific task being addressed. The objective was to build a standardized schema capable of:

- simulating a realistic user request,
- avoiding ambiguity in input formulation,
- simplifying the subsequent results evaluation phase.

Specifically, a simulation prompt was defined to represent the request a user might make to an LLM regarding a specific task, in our case, classifying the toxicity of a text comment. This prompt serves as the basic input for the entire re-engineering process, both with and without ontology support. The prompt is:

```
Classify, with 1 or 0, if the text contain toxicity or not.
Text: <text>. Answer only with: 'toxicity:[]', in the
square parents put 1 if contain toxicity, 0 otherwise.
```

PROMPT 5.1: Prompt that simulate input request

In prompt 5.1, the text tag is dynamically replaced with the comments from the dataset, in order to realistically simulate a user request. In this way, each example from the dataset is transformed into a classification request directed to the LLM, ensuring uniformity in the input process.

In addition to the normal classification request, for the specific needs of this thesis work, the expected output format was also explicitly defined. Although such information can normally be obtained by querying the characteristics of the text classification task in the ontology, it was decided to anticipate its definition directly within the prompt.

This choice was motivated by practical and methodological considerations:

- To ensure consistency between the responses provided by the LLMs, avoiding heterogeneous or ambiguous formats.
- To make the automatic evaluation process simpler and more reliable, thanks to the ability to directly extract the relevant information.
- To reduce the complexity of parsing textual responses, a crucial element when working with a large number of examples.

In the absence of this specific output structuring, each LLM could have responded differently, making the comparison between results less significant or not automatable.

In a normal situation, such as using a chatbot that leverages the proposed methodology, prompt 5.1 would represent the user's request.

Subsequently, the prompt, that is showed in 5.2, was designed to enable the automatic engineering of the initial request defined in 5.1. This prompt represents one of the central elements of the experimental process.

```
You are a prompt engineer. Given a task create one <prompt-
  technique> prompt for that task.
Remember to include:
<characteristics>
Do not solve the prompts.
Put the generated prompt in a section opened with <prompt>
  and closed with <\prompt> .
Task: <task>
```

PROMPT 5.2: Prompt to engineer the input request with ontology

The prompt is structured to guide the LLM in generating a new, optimized prompt, following the semantic information extracted from the ontology. The structure is as follows:

- **Definition of the LLM's role:** At the beginning, an explicit role is assigned to the model, as an expert in prompt engineering. This aims to obtain more structured and technically oriented responses.
- **Specification of the prompting technique:** The <prompt-technique> tag is replaced with one of the two techniques selected for the experiments (Few-shot or Chain of Thought), or with one of the techniques extracted directly from the ontology via the corresponding SPARQL query 1.
- **Inclusion of task characteristics:** In this phase, the LLM is asked to incorporate into the prompt all comment of the characteristics extracted from query 2 related to the task. These characteristics represent crucial conceptual aspects that help the LLM better understand the nature of the request, promoting a more accurate, informative, and coherent response. The characteristics extracted from the ontology were employed in three different configurations. In the first configuration, all available characteristics were used: *Format*, *Language*, *Label type*, and *Label explanation*. In the second, *Format* was excluded, retaining only *Language*, *Label type*, and *Label explanation*. Finally, in the third configuration, the input was further reduced to consider only two characteristics: *Language* and *Label explanation*. This experimental choice aims to evaluate the impact of prompt information enrichment: to verify whether a greater number of

characteristics truly provides an improvement in response quality, or if some characteristics prove more decisive than others.

- **Marking of the generated prompt:** To facilitate its automatic extraction, the LLM is asked to enclose the resulting prompt within the `<prompt>` and `</prompt>` tags.
- **Task contextualization:** Finally, the task to be reformulated is provided, which replaces the `<task>` tag.

Appendix .1.1 provides an example of a complete prompt and the corresponding response obtained via an API call to the LLM. This example shows the actual format of the request sent to the model and the type of response received, as it was saved for subsequent test and evaluation.

As anticipated, unlike other studies in the literature where the automatic prompt engineering process is iteratively optimized through self-refinement techniques by the LLM itself to progressively enhance the quality of responses, this work will not introduce such a phase.

The goal here is to demonstrate that the use of an ontology can reduce or even eliminate the need for these automatic optimization mechanisms. This approach aims to lower computational costs and make the process faster, more interpretable, and replicable, all without compromising the quality of the final result.

5.3.2 Prompt engineering without ontology information

After completing the automatic prompt engineering using the ontology according to the described experimental setup, we proceeded with generating prompts without the informative support of the ontology.

In this case, too, the process was initiated by formatting the dataset sentences according to the schema presented in Section 5.1, thus simulating a standardized user request.

Similar to the previous phase, it was necessary to explicitly specify the output format within the prompt. This choice, as already discussed, was indispensable to ensure coherence and automatability during the evaluation phase. It is important to emphasize that, in the absence of explicit instructions, LLMs do not tend to structure responses uniformly, making it difficult to extract and analyze results.

Subsequently, a generic prompt, shown in 5.3 was created to allow for the automatic engineering of the request based on the identified task. As before, the

`<prompt-technique>` tag is replaced with one of the two techniques adopted in the experiment (Few-shot or Chain of Thought). Alternatively, nothing would prevent the use of techniques directly suggested by the LLM based on its prior knowledge of the task. The prompt structure remains almost unchanged com-

```
You are a prompt engineer. Given a task create one <prompt-
  technique> prompt for that task.
Do not solve the prompts.
Put the generated prompt in a section opened with <prompt>
  and closed with <\prompt> .
Task:{task}
```

PROMPT 5.3: Prompt to engineer the input request with no ontology

pared to the ontology-supported version; the substantial difference lies in the absence of the characteristics related to task that the ontology provided. In this scenario, the LLM is free to engineer the prompt relying exclusively on its own capabilities, without any ontological constraints or external guidance, making the process more "black-box."

A complete example of the prompt and the resulting generation is provided in Appendix .1.2

As previously mentioned, no iterative optimization mechanism for prompts is applied in this phase either. The intention, in fact, is to evaluate whether automatic generation on the first attempt, without optimization, is comparable or inferior to that obtained with the support of the ontology.

5.3.3 Test the generated prompts

Once the prompt engineering phase was complete, an automated testing phase began. During this phase, all generated prompts were fed into their respective LLMs to obtain a response.

Specifically, the content enclosed within the `<prompt>` and `</prompt>` tags, as described earlier, was automatically extracted. This text was then sent back to the model to get the requested classification. The response produced by the LLM was saved and archived to allow for a subsequent evaluation phase.

This process was repeated for every example in the dataset and for each of the generated prompt variations, both with and without the information of the ontology.

A complete example of an extracted prompt and its corresponding generated response is provided in Appendix .1.3.

5.4 Evaluation approach

The evaluation phase was divided into three different analyses, each aiming to investigate a specific aspect of the proposed system and the effectiveness of using the ontology within the automatic prompt engineering process.

1. **Statistical Evaluation of Results:** The first analysis adopted a standard quantitative approach in the field of text classification, calculating the metrics of precision, recall, and F1-score. These metrics were calculated for each experimental prompt engineering configuration, comparing the responses obtained from the LLMs with the original labels of the dataset (ground truth).
2. **Evaluation of Prompting Techniques:** The second analysis focused on the conceptual coverage of the ontology regarding prompting techniques truly relevant to the text classification task.

To this end, for each LLM used, a list of prompting techniques considered appropriate for the task was generated. The prompt used for this generation is reported in 5.4.

```
You are a prompt engineer. Given a task generate different
prompts techniques from literature that you tink are
useful for engineering the task of text classification.
Do not create the prompt.
Give the output in a python list form.
```

PROMPT 5.4: Prompt to generate prompt techniques for text classification

Subsequently, these lists were compared with the techniques present within the ontology to evaluate whether the ontology’s conceptual modeling is sufficiently complete or has gaps.

3. **Qualitative Analysis of Generated Prompts:** Finally, a comparative qualitative analysis was conducted between prompts generated with and without the aid of the ontology.

In this phase, a sample of prompts was randomly selected and manually evaluated, focusing on the quality, clarity, and structure of the prompt itself, independently of the response obtained.

The objective is to determine whether the inclusion of structured and contextual information from the ontology actually leads to a more effective

and readable engineering of prompts compared to what an autonomously operating LLM can achieve.

All results derived from these analyses will be presented and discussed in the next chapter.

Chapter 6

Experimental Evaluations

In this chapter, the results obtained from the activities described in the previous chapter will be presented and analyzed. We will begin with a statistical evaluation of model performance using precision, recall, and F1-score metrics, accompanied by a summary analysis of the overall effectiveness of the adopted prompting techniques.

Subsequently, a quantitative analysis will be conducted to compare the number of prompting techniques associated with the text classification task according to the ontology with those autonomously generated by the LLMs. This comparison aims to verify whether the ontology is sufficiently comprehensive to cover, at least in part, the relevant techniques for the task considered.

Finally, a qualitative discussion will be offered on the prompts generated with and without ontological support, highlighting differences in the structuring and clarity of the prompt itself.

6.1 Quantitative analysis

Regarding the statistical analysis, before proceeding with the calculation of performance metrics, it was necessary to identify and account for invalid results. This step was indispensable because the classification is automatically extracted from a structured output, such as in the form "*toxicity: []*", as described previously. If the response generated by the LLM did not adhere to this structure, the classification could not be considered valid.

This check was introduced for two main reasons: firstly, the large number of texts to analyze made manual evaluation impractical; secondly, the absence of the requested format represents an error on the part of the LLM, since the prompt clearly specified the need for a predefined output structure.

This counting phase was carried out for both LLMs. Furthermore, to better understand the nature of the errors, an additional analysis was performed to

determine how many of the texts with invalid output were actually toxic according to the true label. The objective was to verify whether LLMs tended to make more errors with certain types of texts or if the errors were instead independent of the content.

As shown in Figure 6.1 and Figure 6.2, Gemini exhibited a significantly higher number of errors compared to DeepSeek, particularly in cases where few-shot or longer, more articulated prompts were used. This suggests a greater difficulty for Gemini in handling complex prompts, compared to simpler and more direct ones.

<input checked="" type="checkbox"/>	Ontology	# Characteristics	Prompt technique	# Errors	# Errors with Toxicity = 1
<input checked="" type="checkbox"/>		Four	CoT	8	4
<input checked="" type="checkbox"/>		Four	Few-shot	9	6
<input checked="" type="checkbox"/>		Three	CoT	9	4
<input checked="" type="checkbox"/>		Three	Few-shot	45	15
<input checked="" type="checkbox"/>		Two	CoT	15	8
<input checked="" type="checkbox"/>		Two	Few-shot	4	1
<input type="checkbox"/>			CoT	17	8
<input type="checkbox"/>			Few-shot	16	7

FIGURE 6.1: Numbers of errors in the output format in Gemini

<input checked="" type="checkbox"/>	Ontology	# Characteristics	Prompt technique	# Errors	# Errors with Toxicity = 1
<input checked="" type="checkbox"/>		Four	CoT	12	3
<input checked="" type="checkbox"/>		Four	Few-shot	1	0
<input checked="" type="checkbox"/>		Three	CoT	4	1
<input checked="" type="checkbox"/>		Three	Few-shot	0	0
<input checked="" type="checkbox"/>		Two	CoT	8	4
<input checked="" type="checkbox"/>		Two	Few-shot	0	0
<input type="checkbox"/>			CoT	11	3
<input type="checkbox"/>			Few-shot	0	0

FIGURE 6.2: Numbers of errors in the output format in DeepSeek

From the same figures, it also emerges that there is no correlation between formatting errors and the nature of the input text: the distribution of erroneously classified texts mirrors the initial one, confirming that the errors lie in the model's ability to manage the prompt structure and not in its understanding of the content.

For analytical purposes, all invalid responses were labeled with a classification value of -1, so as to consider them as definite errors in the subsequent analysis, regardless of the true label associated with the text.

Finally, it is important to emphasize that in this phase, the LLM did not fail in classifying the content itself, but rather in formatting the output, a critical element in an automated evaluation process.

Moving on to the evaluation metrics, three distinct tests were performed, each involving the calculation of the main standard metrics used in automatic classification: precision, recall, and F1-score.

- **Precision:** Measures the percentage of elements classified as positive that are actually correct. It indicates how accurate the model is when making a positive prediction.
- **Recall:** Measures the model's ability to correctly identify all positive elements present in the dataset.
- **F1-score:** Represents the harmonic mean between precision and recall, useful when it is necessary to balance both metrics, especially in the presence of imbalanced datasets.

The initial tests concerned the use of the ontology. Specifically, the classification metrics obtained with different feature configurations in prompts generated using the ontology were compared for both LLMs.

Figure 6.3 shows the results obtained with Gemini. The analysis reveals a decline in performance depending on the configuration used, particularly with Few-shot. This confirms what was previously observed: Gemini shows difficulty in handling more articulated and complex prompts.

# Characteristics	Prompt technique	# Precision	# Recall	# F1-score
Two	Few-shot	0,75	0,70	0,72
Two	CoT	0,74	0,63	0,68
Three	Few-shot	0,78	0,44	0,56
Three	CoT	0,72	0,66	0,69
Four	Few-shot	0,68	0,62	0,65
Four	CoT	0,78	0,71	0,75

FIGURE 6.3: Metrics comparison for Gemini test with ontology

Figure 6.4, on the other hand, displays DeepSeek's results. In this case, the performance is significantly better than Gemini's, maintaining good levels of

precision, recall, and F1-score across all configurations. A slight advantage is observed in prompts that include only three features, suggesting that a certain degree of conciseness can be more effective.

# Characteristics	Prompt technique	# Precision	# Recall	# F1-score
Two	Few-shot	0,72	0,73	0,72
Two	CoT	0,74	0,69	0,71
Three	Few-shot	0,72	0,72	0,70
Three	CoT	0,76	0,73	0,74
Four	Few-shot	0,74	0,74	0,72
Four	CoT	0,76	0,68	0,70

FIGURE 6.4: Metrics comparison for DeepSeek test with ontology

Finally, Figure 6.5 presents a comparison between the best configurations obtained through the use of the ontology and the results related to prompts generated without any semantic support.

Model	Ontology	# Characteristics	Prompt technique	# Precision	# Recall	# F1-score
Gemini 2.0	✓	Two	Few-shot	0,75	0,70	0,72
Gemini 2.0	✓	Two	CoT	0,74	0,63	0,68
DeepSeek V3	✓	Three	Few-shot	0,72	0,72	0,70
DeepSeek V3	✓	Three	CoT	0,76	0,73	0,74
Gemini 2.0	□		Few-shot	0,70	0,59	0,64
Gemini 2.0	□		CoT	0,73	0,61	0,66
DeepSeek V3	□		Few-shot	0,70	0,70	0,67
DeepSeek V3	□		CoT	0,63	0,57	0,58

FIGURE 6.5: Metrics comparison for test with and without ontology

This analysis clearly confirms the effectiveness of the ontology: in all cases, the use of the semantic component leads to an improvement in metrics compared to prompts generated without ontological support. Although it is clear that the ontology adds value, the issue of optimizing the choice of features based on the adopted LLM and the specific task remains open, in order to achieve maximum performance.

An additional evaluation was performed by testing the initial input prompt created in 5.1. However, no particularly relevant conclusions were drawn from this step; therefore, the analysis is reported in the appendix .2 for completeness.

Finally, an additional analysis was conducted to evaluate which of the two adopted prompting techniques, Chain of Thought and Few-shot, produced the best results in tests that involved the use of ontological information.

To perform this evaluation, a box plot was used for each of the metrics (precision, recall, F1-score). The box plot is a useful graphical tool for representing the distribution of a data set. It shows the minimum, maximum, median, and quartiles, allowing for an immediate view of variability and the presence of any outliers. In this case, the box plots were enriched with annotations indicating the relative position of the two techniques in the metrics space, enabling a direct visual comparison of their performance.

As shown in Figure 6.6, it is observed that, on average, the Chain of Thought technique achieved more stable and overall better results. Conversely, Few-shot prompting showed less consistent performance, with greater variability, and in many cases, as already highlighted in the results for Gemini, it suffered in the presence of longer or more complex prompts.

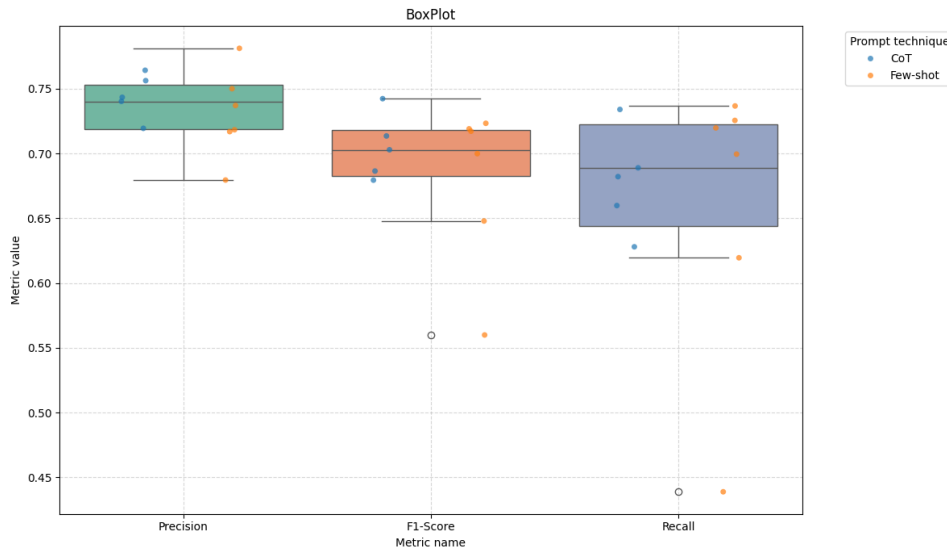


FIGURE 6.6: BoxPlot for prompt technique efficiency evaluation

This analysis further supports the idea that the quality of prompting and its compatibility with the model used are fundamental elements for the success of automatic classification.

6.2 Evaluation of Prompting Techniques

The next analysis consists of a quantitative evaluation of the coverage offered by the ontology regarding prompting techniques directly suggested by LLMs for the engineering of a prompt related to a text classification task.

As described in previous chapters, both LLMs were asked to generate a list of prompting techniques to be used for the task, and the results are shown in

Figure 6.7. The techniques highlighted in green represent those already present within the ontology, while those in red are absent.

The analysis shows that the ontology includes most of the suggested techniques, especially the more common and consolidated ones in prompt engineering practice. The missing techniques, on the other hand, are more peculiar methods, often designed for specific or niche contexts.

This result suggests that the ontology, even in its current form, represents a solid and sufficiently broad foundation to support the automatic engineering of prompts. The coverage provided is indeed adequate for use in various experimental scenarios, ensuring flexibility and adaptability. Furthermore, it is important to emphasize that the ontology also includes additional techniques not mentioned by the LLMs, further expanding the range of possibilities available for automatic prompt adaptation to different application needs.

Gemini	DeepSeek
Few-Shot	Zero-shot prompting
Chain-of-Thought (CoT)	Few-shot prompting
Zero-Shot	Chain-of-thought (CoT) prompting
Decomposition	Self-consistency prompting
Self-Consistency	Role-based prompting
Prompt Chaining	Decomposition prompting
Role-Playing Prompts	Retrieval-augmented prompting
Contrastive Prompting	Explanation-based prompting
Knowledge-Augmented	Multi-task prompting
Adversarial Prompting	Template-based prompting
	Iterative refinement prompting
	Domain-adaptive prompting
	Contrastive prompting
	Calibration prompting
	Adversarial prompting
	Instruction-based prompting

FIGURE 6.7: Prompt techniques suggested by Gemini and DeepSeek to engineering a text classification task

6.3 Qualitative Analysis of Generated Prompts

As a final analysis and discussion point, we propose a qualitative comparison between two examples of generated prompts: one with ontology support and one without. This comparison aims to highlight the structural and informational differences introduced by the use of the ontology in the automatic prompt engineering process.

The comparison was performed on a randomly selected sample of prompts; however, here we will present a single representative example, as it is consistent with the average trend observed.

.1.4 shows a Few-shot prompt generated with DeepSeek using information provided by the ontology. On the other hand, .1.5 shows a Few-shot prompt generated with the same model, but without any ontological support.

Analyzing the first case .1.4, it is possible to note the presence of a richer and more informative structure: the semantic characteristics extracted from the ontology have been integrated into the prompt, making it more adherent to the good practices of prompt engineering. These characteristics constitute the only, but substantial, difference compared to the prompt without an ontology.

The semantic enrichment allows the model to operate in a more controlled and specific context, reducing the risk of hallucinations or misinterpretations of the task by the LLM. For example, the specification of the *format* and *language* clarifies that the text to be classified consists of short sentences in a defined language, and not extracted from more complex documents. The characteristics related to *labels*, on the other hand, help to immediately define the objective of the classification, explicitly stating the meaning of the classes and the user's intent.

Before concluding, a short observation concerning that this approach was adopted due to the absence, in the current state of the art, of a well-defined and standardized methodology for evaluating the correctness of automatically engineered prompts. As a result, it was not possible to rely on existing benchmarks or objective reference standards for prompt quality. Therefore, the comparison was conducted qualitatively, based on principles derived from prompt engineering best practices and the observable coherence between the generated prompt and the intended task.

6.4 Results discussion

In light of the evaluations conducted, it is possible to provide an answer to the initially posed research questions:

- **RQ1:** *Does the ontology improve the effectiveness of automatic prompt engineer compared to LLM?*
- **RQ2:** *Does the ontology improve the efficiency in the automatic prompt engineering process?*

Starting with RQ1, the results of the calculated metrics clearly show how the use of the ontology improves the quality of prompts and, consequently, the responses provided by the LLMs. The integration of structured semantic knowledge allows for enriching the prompt from the outset with specific and pertinent information relative to the requested task, reducing ambiguity and increasing the coherence of the generated output.

This consideration also allows us to answer RQ2: yes, the use of the ontology provides tangible advantages in automatic prompt engineering. Through simple and automatizable SPARQL queries, it is possible to incorporate contextual and semantic elements into the prompt for a variety of tasks. This process helps reduce the need for subsequent prompt optimization, often required in methods lacking structured support.

It should also be highlighted that the ontology represents a customizable, reusable, and shareable tool, characteristics that make it extremely versatile. In particular, the ability to adapt it to different application contexts allows for optimizing prompts for specific needs, reducing the computational load and, in some cases, even avoiding the need for model retraining to adapt to new domains.

When commenting on the LLMs employed, it is worth noting that certain models, such as Gemini, require particular attention in the selection of input information. In these cases, providing overly verbose or complex prompts can lead to performance degradation, as the architecture may struggle to effectively process longer or more articulated inputs. This limitation does not reflect negatively on the use of the ontology itself; rather, it underscores the importance of conducting a preliminary analysis of the specific characteristics and constraints of the LLM being adopted. Such an evaluation ensures that the ontology-driven prompt engineering strategy is adapted appropriately to the model's capabilities, ultimately maximizing its effectiveness.

In conclusion, the ontology proves to be not only useful but also strategic for improving and making the automated prompt engineering process more efficient.

Chapter 7

Conclusions and Future Works

In this chapter, the final considerations on the thesis work and possible future developments will be presented.

7.1 Conclusions

Our thesis aimed to explore the usefulness and effectiveness of using an ontology for automated prompt engineering. Specifically, we wanted to see if this approach could speed up and optimize the process, especially in scenarios with technical limitations, like a lack of prompt engineering experts, or experimental constraints, such as limited advanced computational resources or budgets for extensive LLM use.

Given our results, we can conclude that the ontology is an extremely powerful and promising tool for supporting interaction with Large Language Models. We found that integrating even a limited number of semantic characteristics led to a significant improvement in the quality of responses. This suggests that a more extensive and customized ontology, fine-tuned for specific application contexts, could yield results comparable to, or even surpass, those from purely LLM-based approaches, which often need many iterations to perfect a prompt.

Therefore, this work highlights the value of the ontological approach in automated prompt engineering and lays the groundwork for future developments in this research area.

7.2 Future works

Building on the results from this thesis project, we can outline several future developments to expand and further refine the proposed approach:

- **Ontology Extension:** A natural first step is to expand the ontology with new semantic characteristics. These would be designed to support

additional tasks beyond those already considered. Furthermore, the conceptual model could be extended to include a formal representation of prompt characteristics themselves. This would allow us to generate not just useful informative content, but actual structured templates, tailored for each combination of task and prompting technique.

- **Experimental Expansion of Automated Prompt Engineering:** Another direction involves expanding the experimentation with new LLMs, datasets, and tasks. This would help identify the approach’s limitations and potential in more heterogeneous contexts, verifying the ontology’s adaptability to different models and domains, as well as any critical issues related to specific LLM architectures.
- **Guided LLM-User Hybrid Approach:** Finally, we could explore a semi-automatic approach where the ontology is used not only to directly generate prompts but also to support and guide the user in the creation process. In this scenario, the LLM could propose an initial version of the prompt based on ontological information, then allow the user to intervene in an assisted manner to refine it. This type of human-model collaboration could further improve the effectiveness and accessibility of prompt engineering.

Appendices

.1 Prompt Examples

This appendix provides some examples of prompt generation and testing. Each example is cited appropriately in the main text.

.1.1 Example 1

Input:

```
You are a prompt engineer.
Given a task create one Few-shot prompt for that task.

Remember to include

-Definition of the labels
-Language of the text

Do not solve the prompts.

Put the generated prompt in a section opened with <prompt>
and closed with <\prompt> .

Task: Dear world:
Just a reminder that none of these folks actually live in
    Oregon. Please remember that when you decide to paint
        our state as a bunch of wackos!

Sincerely,
Oregon
```

Output:

```
Okay, I will generate a few-shot prompt for classifying
text toxicity.

<prompt>
You are an expert in identifying toxic language. You will
be provided with text and your task is to classify
whether the text contains toxicity. You should output '
toxicity: [1]' if the text contains toxicity, and '
toxicity: [0]' if the text does not contain toxicity.

Here's how to interpret the labels:
```

```
*   **1:** The text contains toxic language, including hate
      speech, threats, insults, or disparaging remarks.
*   **0:** The text does not contain toxic language. It is
      generally polite, respectful, and does not contain
      offensive content.
```

The language of the text is English.

Here are some examples:

```
Text: "You are a worthless idiot and nobody likes you."
toxicity: [1]
```

```
Text: "Have a great day!"
toxicity: [0]
```

```
Text: "I'm going to punch you in the face."
toxicity: [1]
```

```
Text: "This is a friendly reminder to submit your reports
      by Friday."
toxicity: [0]
```

```
Text: "You are a moron and I hate your guts."
toxicity: [1]
```

```
Text: 'Dear world:
Just a reminder that none of these folks actually live in
      Oregon. Please remember that when you decide to paint
      our state as a bunch of wackos!
```

```
Sincerely,
Oregon'
toxicity: []
</prompt>
```

.1.2 Example 2

Input:

```
You are a prompt engineer . Given a task create one Few-
shot prompt for that task .
```

Do not solve the prompts .
Put the generated prompt in a section opened with < prompt
>
and closed with <\ prompt > .
Task : 'I have to disagree on this one! I have 3 pairs and
would like more for every day of my cycle. I have a
grotesquely heavy period. I am able to wear these most
days without a backup, and I am not weirdly damp down
there. Tampons and pads give me infections, and the cup
doesn't work for my weirdly shaped cervix. This has been
a LIFE SAVER for me because I ALWAYS LEAKED from
tampons and pads and now I don't. This means being able
to leave the house during my period! This was an issue
before due to the leaking. My sister has the same deal.
So this is great for some women. If you have a heavy
period and a leaking problem, I highly recommend these!'

Output:

```
<prompt>
Task: Classify, with 1 or 0, if the text contains toxicity
      or not.

Example 1:
Text: 'I hate you so much! You're the worst person I've
      ever met.'
Answer: toxicity: [1]

Example 2:
Text: 'The weather today is really nice. I love sunny days
      .'
Answer: toxicity: [0]

Example 3:
Text: 'This product is terrible and anyone who buys it is
      an idiot.'
Answer: toxicity: [1]

Now classify the following text:
```


Text: 'I have to disagree on this one! I have 3 pairs and would like more for every day of my cycle. I have a grotesquely heavy period. I am able to wear these most days without a backup, and I am not weirdly damp down there. Tampons and pads give me infections, and the cup doesn't work for my weirdly shaped cervix. This has been a LIFE SAVER for me because I ALWAYS LEAKED from tampons and pads and now I don't. This means being able to leave the house during my period! This was an issue before due to the leaking. My sister has the same deal. So this is great for some women. If you have a heavy period and a leaking problem, I highly recommend these!'

Answer: toxicity: []

</prompt>

.1.3 Example 3

Input:

You are an expert in identifying toxic language in text. You will be provided with a text sample and your task is to classify whether the text contains toxicity or not. Toxicity includes hate speech, offensive language, insults, threats, or any form of content intended to demean, marginalize, or promote discrimination against individuals or groups.

Here's how you should think step by step:

1. Carefully read and understand the provided text.
2. Identify any words, phrases, or statements that could be considered offensive, hateful, discriminatory, or threatening.
3. Evaluate the overall context of the text. Consider the intent behind the words and whether they are likely to cause harm or offense.
4. Determine whether the text contains toxicity based on your analysis.
5. Assign a binary label:
 - * 1: If the text contains toxicity.
 - * 0: If the text does not contain toxicity.

6. Present the answer in the format: 'toxicity: []', inserting the binary label (0 or 1) within the square brackets.

Here's an example:

Text: "This is a friendly message."

The text does not contain toxicity.

toxicity: [0]

Now, analyze the following text and provide your classification:

Text: 'Yet call out all Muslims for the acts of a few will get you pilloried. So why is it okay to smear an entire religion over these few idiots? Or is this because it's okay to bash Christian sects?'

Output:

toxicity: [0]

.1.4 Example 4

Task: Classify whether the given text contains toxicity (1) or not (0).

Format:

- The input text is a single sentence or a short paragraph written in English.
- The output should be in the format: 'toxicity: [label]', where 'label' is either '1' (toxic) or '0' (non-toxic).

Explanation of the label:

- '1' (Toxic): The text contains harmful, offensive, or disrespectful language, including insults, hate speech, or demeaning remarks.
- '0' (Non-toxic): The text is neutral, respectful, or positive, with no harmful intent or language.

Label type:

- Binary classification (0 or 1).

```
### Language:
```

```
- English.
```

```
### Examples:
```

```
1. Text: "Love that they also participate in the Portland
Farmer's Market at PSU... and trying to gluten-shame
Tastebud and propping Papa Murphy's in the same sentence
is laughable."
```

```
Answer: toxicity: [1]
```

```
2. Text: "The weather today is absolutely beautiful, and I'm
enjoying my walk in the park."
```

```
Answer: toxicity: [0]
```

```
3. Text: "Your opinion is worthless, and you should just
keep it to yourself."
```

```
Answer: toxicity: [1]
```

```
Now, classify the following text:
```

```
Text: "Love that they also participate in the Portland
Farmer's Market at PSU... and trying to gluten-shame
Tastebud and propping Papa Murphy's in the same sentence
is laughable."
```

```
Answer: toxicity: []
```

.1.5 Example 5

```
Task: Classify, with 1 or 0, if the text contains toxicity
or not. For each given text, answer only with 'toxicity:
[]', where you put 1 if the text contains toxicity, and
0 otherwise.
```

```
Examples:
```

```
1. Text: 'This is a wonderful day!'
```

```
Answer: toxicity: [0]
```

```
2. Text: 'You are an idiot and your opinion is worthless.'
```

```
Answer: toxicity: [1]
```

```
3. Text: 'The service here is slow and disappointing.'
```

```
Answer: toxicity: [0]
```

```
Now classify the following text:
```

```
Text: 'Love that they also participate in the Portland  
Farmer's Market at PSU... and trying to gluten-shame  
Tastebud and propping Papa Murphy's in the same sentence  
is laughable.'
```

```
Answer: toxicity: []
```

.2 Comparison with plain prompt

This appendix presents a test conducted during the automated prompt engineering phase that, however, didn't yield results or insights useful for the main objectives of the thesis. For this reason, it was excluded from the main body but is included here for completeness.

Specifically, during the evaluation phase, the initial prompts shown in 1 were also tested to determine if obtaining responses directly from these prompts, without any re-engineering process, could be more effective.

From the analysis of the performance metrics, reported in Figure 1, it's observed that the results obtained are indeed superior to those achieved through automatic re-engineering, both with and without ontology support. However, these results can be considered partially "biased." The prompt in 1 is, in fact, already a form of engineered prompt, structured to force a coherent response from the LLM. Although it simulates a generic user request, there's no guarantee that a real user would have actually formulated that question in the same way.

Model	# Precision	# Recall	# F1-score
Gemini 2.0	0,75	0,74	0,75
DeepSeek V3	0,77	0,78	0,77

FIGURE 1: Plain test metrics

In any case, regardless of the metrics obtained, the thesis's purpose isn't to verify whether this specific initial prompt needs re-engineering or not, but rather to analyze if using an ontology can improve that process. Therefore, even if a simple prompt like the one tested produces good results without re-engineering, it doesn't offer relevant insights regarding the research objective.

Bibliography

- [1] Michael Ashburner et al. “Gene ontology: tool for the unification of biology”. In: *Nature genetics* 25.1 (2000), pp. 25–29.
- [2] Emily M Bender et al. “On the dangers of stochastic parrots: Can language models be too big?” In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 2021, pp. 610–623.
- [3] Daniel Borkan et al. *Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification*. 2019. arXiv: 1903.04561 [cs.LG]. URL: <https://arxiv.org/abs/1903.04561>.
- [4] Weizhe Chen, Sven Koenig, and Bistra Dilkina. “Reprompt: Planning by automatic prompt engineering for large language models agents”. In: *arXiv preprint arXiv:2406.11132* (2024).
- [5] Tolga Çöplü et al. “Prompt-time ontology-driven symbolic knowledge capture with large language models”. In: *arXiv preprint arXiv:2405.14012* (2024).
- [6] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186.
- [7] Muhayy Ud Din et al. “Ontology-driven Prompt Tuning for LLM-based Task and Motion Planning”. In: *arXiv preprint arXiv:2412.07493* (2024).
- [8] Kevin Donnelly et al. “SNOMED-CT: The advanced terminology and coding system for eHealth”. In: *Studies in health technology and informatics* 121 (2006), p. 279.
- [9] Christiane Fellbaum. “WordNet”. In: *Theory and applications of ontology: computer applications*. Springer, 2010, pp. 231–243.
- [10] Gernot A Fink. “n-Gram Models”. In: *Markov Models for Pattern Recognition: From Theory to Applications* (2008), pp. 95–113.

- [11] Wensheng Gan et al. *Mixture of Experts (MoE): A Big Data Perspective*. 2025. arXiv: 2501.16352 [cs.LG]. URL: <https://arxiv.org/abs/2501.16352>.
- [12] Lingyu Gao, Debanjan Ghosh, and Kevin Gimpel. *The Benefits of Label-Description Training for Zero-Shot Text Classification*. 2023. arXiv: 2305.02239 [cs.CL]. URL: <https://arxiv.org/abs/2305.02239>.
- [13] Simone Gramegna et al. “Analysis and development of a prompt engineering ontology”. Master degree thesis. Università degli studi di Bari "Aldo Moro", 2024.
- [14] Ramanathan V Guha, Dan Brickley, and Steve Macbeth. “Schema. org: evolution of structured data on the web”. In: *Communications of the ACM* 59.2 (2016), pp. 44–51.
- [15] Muhammad Usman Hadi et al. “Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects”. In: *Authorea Preprints* 1 (2023), pp. 1–26.
- [16] Zhemin Ji et al. “Survey of Hallucination in Natural Language Generation”. In: *ACM Computing Surveys* (2023).
- [17] Wenxiang Jiao et al. “Is ChatGPT a good translator? Yes with GPT-4 as the engine”. In: *arXiv preprint arXiv:2301.08745* (2023).
- [18] Edlira Kalemi and Edlira Martiri. “FOAF-academic ontology: a vocabulary for the academic community”. In: *2011 Third International Conference on Intelligent Networking and Collaborative Systems*. IEEE. 2011, pp. 440–445.
- [19] Kamran Kowsari et al. “Text classification algorithms: A survey”. In: *Information* 10.4 (2019), p. 150.
- [20] Mike Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. URL: <https://aclanthology.org/2020.acl-main.703/>.
- [21] Patrick Lewis et al. “Retrieval-augmented generation for knowledge-intensive nlp tasks”. In: *Advances in neural information processing systems* 33 (2020), pp. 9459–9474.

- [22] Cheng Li et al. “Large language models understand and can be enhanced by emotional stimuli”. In: *arXiv preprint arXiv:2307.11760* (2023).
- [23] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).
- [24] Yao Lu et al. “Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity”. In: *arXiv preprint arXiv:2104.08786* (2021).
- [25] Joshua Maynez et al. “On Faithfulness and Factuality in Abstractive Summarization”. In: *ACL*. 2020.
- [26] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).
- [27] Oleksandr Palagin et al. “OntoChatGPT information system: Ontology-driven structured prompts for ChatGPT meta-learning”. In: *arXiv preprint arXiv:2307.05082* (2023).
- [28] Jeffrey Pennington, Richard Socher, and Christopher Manning. “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543.
DOI: 10.3115/v1/D14-1162. URL: <https://aclanthology.org/D14-1162/>.
- [29] Matthew E. Peters et al. “Deep Contextualized Word Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Ed. by Marilyn Walker, Heng Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237.
DOI: 10.18653/v1/N18-1202. URL: <https://aclanthology.org/N18-1202/>.
- [30] María Poveda-Villalón et al. “LOT: An industrial oriented ontology engineering framework”. In: *Engineering Applications of Artificial Intelligence* 111 (2022), p. 104755.
- [31] Robin M Schmidt. “Recurrent neural networks (rnns): A gentle introduction and overview”. In: *arXiv preprint arXiv:1912.05911* (2019).

- [32] S Schulhoff et al. “The prompt report: a systematic survey of prompt engineering techniques”. In: *Preprint at <https://arxiv.org/abs/2406.06608>* (2024).
- [33] Xavier Sécheresse, Antoine Villedieu de Torcy, et al. “GAAPO: Genetic Algorithmic Applied to Prompt Optimization”. In: *arXiv preprint arXiv:2504.07157* (2025).
- [34] Minghao Shao et al. “Survey of different large language model architectures: Trends, benchmarks, and challenges”. In: *IEEE Access* (2024).
- [35] Taylor Shin et al. “Autoprompt: Eliciting knowledge from language models with automatically generated prompts”. In: *arXiv preprint arXiv:2010.15980* (2020).
- [36] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
- [37] Xuezhi Wang et al. “Self-consistency improves chain of thought reasoning in language models”. In: *arXiv preprint arXiv:2203.11171* (2022).
- [38] Yuqing Wang and Yun Zhao. “Metacognitive prompting improves understanding in large language models”. In: *arXiv preprint arXiv:2308.05342* (2023).
- [39] Jason Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.
- [40] Chengrun Yang et al. “Large language models as optimizers”. In: *arXiv preprint arXiv:2309.03409* (2023).
- [41] Qinyuan Ye et al. “Prompt engineering a prompt engineer”. In: *arXiv preprint arXiv:2311.05661* (2023).
- [42] Jingqing Zhang et al. “Pegasus: Pre-training with extracted gap-sentences for abstractive summarization”. In: *International conference on machine learning*. PMLR. 2020, pp. 11328–11339.
- [43] Tianjun Zhang et al. “Tempera: Test-time prompting via reinforcement learning”. In: *arXiv preprint arXiv:2211.11890* (2022).
- [44] Wayne Xin Zhao et al. *A Survey of Large Language Models*. 2025. arXiv: 2303.18223 [cs.CL]. URL: <https://arxiv.org/abs/2303.18223>.
- [45] Mingqian Zheng, Jiabin Pei, and David Jurgens. “Is" a helpful assistant" the best role for large language models? a systematic evaluation of social roles in system prompts”. In: *arXiv preprint arXiv:2311.10054* 8 (2023).

-
- [46] Yongchao Zhou et al. “Large language models are human-level prompt engineers”. In: *The Eleventh International Conference on Learning Representations*. 2022.

Acknowledgements

Questo lavoro di tesi conclude un percorso durato tanto tempo e chiude il capitolo della mia vita da studente. Per questo motivo ci tengo a fare dei ringraziamenti, a modo mio.

Innanzitutto, un ringraziamento alla prof.ssa **Claudia d'Amato**, al dott. **Roberto Barile** e al dott. **Andrea Nuzzolese**, che mi hanno seguito in questo progetto e mi hanno accompagnato nella conclusione di un percorso in un ramo dell' AI che ho scelto di approfondire e portare avanti fino all' ultimo dei miei esami. Un particolare grazie alla prof.ssa d'Amato, che mi ha insegnato ad avere più fiducia nel mio operato e a credere nelle mie capacità.

Desidero poi ringraziare i miei compagni universitari. In primis **Francesco Didio**, con cui ho affrontato praticamente tutti gli esami e con cui ci sopportiamo già dalla triennale. Insieme abbiamo scelto di intraprendere questo percorso e, fortunatamente, di portarlo a termine. Oltre all' università, abbiamo condiviso anche la quotidianità tra prese in giro e cazzeggio, proprio per questo non capisco come sia possibile che non mi ritrovo con nessun arto spezzato. A seguire, ringrazio **Gianfederico Poli**, collega universitario con cui ho condiviso il percorso magistrale. Unico elemento serio del trio, è stato una fonte di ispirazione, nonché un vero amico in ogni momento di questi anni, sia sul piano universitario che personale. Nonostante ciò, ancora mi chiedo come abbia fatto a sopportarmi tutto questo tempo. Ringrazio anche **Nicolas Pinto**, amico da ormai troppo tempo: seppur con tanti tira e molla, siamo sempre rimasti in contatto, condividendo esperienze universitarie e non. Infine, un grazie a tutti i colleghi e amici universitari con cui, in qualche modo, ho condiviso questo cammino.

Voglio poi ringraziare i miei amici di tutti i giorni, in particolare chi con me ha passato, e passa, intere giornate su Discord. A **Frank Palladio**, con cui abbiamo buttato fin troppo tempo a guardare i peggiori film mai concepiti dal genere umano (i quali sono forse causa di questo ritardo negli studi), e con cui condivido passioni e interessi. A **Giovanni Pontillo**, amico vero, con cui si può litigare senza mai odiarsi. A **Pietro Liuzzi**, che conosco da più tempo di tutti e con cui, per qualche arcano motivo, continuo a condividere molte esperienze. È l' unica persona con cui riesco a spaziare tra attività completamente diverse

senza un reale motivo. E grazie a tutti gli altri amici con cui ho condiviso anche solo qualche ora, ma sempre senza annoiarmi.

Uno dei ringraziamenti più importanti va alla mia famiglia, in particolare a **mamma** e **papà**, che mi hanno permesso di proseguire gli studi a spese loro, che non mi hanno mai impedito di fare ciò che volevo, che mi hanno convinto a continuare anche nei momenti in cui avrei voluto mollare, e che, per ora, non mi hanno cacciato di casa.

Infine, il ringraziamento più importante va alla mia fidanzata **Miriana**, che mi sopporta da esattamente l'inizio del mio percorso universitario e che, speriamo, continui a farlo anche per il resto della mia (forse breve) vita. È stata un supporto indispensabile nei momenti peggiori. Insieme ne abbiamo passate tante, e gliene sono grato. Chissà per quanto ancora riuscirà a sopportarmi, anche perchè a chi dedicherei il 90% dello stipendio?.

Alcuni ringraziamenti speciali a cose e persone che mi hanno accompagnato in questi anni: Il ragazzo e l'airone, Lillo, Winnie the Pooh, Topolino, Masseria Brigantino, i monopattini, gli estintori, Giordano Comes, Cuphead, Kubrick, Superman, Yankee Candle, DeepL.