ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Actively Ontology Learning from Large Language Models

Tesi di laurea in:
Intelligent Systems Engineering

Relatore
Prof. Andrea Omicini

Candidato
Riccardo Squarcialupi

Correlatori
Dott. Matteo Magnini
Prof. Ana Ozaki

I Sessione di Laurea
Anno Accademico 2023-2024

Abstract

Active learning is a framework where a learner attempts to learn some kind of knowledge by posing questions to a teacher. In computational learning theory, classically, the questions made by the learner are called *membership queries* and are answered with 'yes' or 'no' (or equivalently, with 'true' or 'false'). Here we consider that the teacher is a language model and study the case in which the knowledge is expressed as an ontology. To evaluate the approach, we present results showing the performance of GPT and other language models when answering whether valid expressions on existing $\mathcal{EL}$ are "true" or "false".

Fortes fortuna adiuvat. (Publio Terenzio Afro)
Tu ne cede malis, sed contra audentior ito. (Virgilio)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The appliement of Large Language Models (LLMs) to the ontologies field marks a pioneering advancement in the realm of artificial intelligence (AI), starting a new era of precision and utility in automated reasoning and knowledge extraction. These models, renowned for their exceptional ability to generate human-like text, are now being challenged to extend beyond mere text processing to structuring knowledge. Ontologies delineate and categorize relationships between concepts within a domain, significantly enhancing the semantic interpretation of data. Despite their proficiency in generating and understanding texts, LLMs remain uncertain for ontology learning abilities.

LLMs have now amassed so much information and improved their question-answering abilities that we are willing to interact and learn from them. These prompts range from general knowledge queries, such as basic definitions and historical events, to more domain-specific questions, like scientific facts related to health and medicine. Recently, initial efforts have been made to automate knowledge extraction from LLMs [73], where the knowledge is extracted in the form of an *ontology* [29]. In the field of knowledge representation and reasoning, ontologies are a common method for representing the relevant conceptual knowledge of a domain. The most popular formalism for specifying ontologies is given by *description logic* (DL) [7]. Ontologies have been extensively used to represent hierarchies and support data integration, information retrieval, and automated reasoning in life sciences [60].

This thesis investigates the feasibility of learning description logic ontologies from LLMs using an active learning approach based on Angluin's exact learning [2] framework from computational learning theory.

The methodology employed in this research involves a multi-phase experimental approach to evaluate the performance of LLMs in ontology learning.

In the first phase, various LLMs, including GPT, are probed with a set of predefined queries. These queries are designed to test the models' understanding of basic description logic constructs and their ability to provide accurate concept inclusions. The responses are then analyzed for correctness and logical consistency.

Building on the initial findings, in the second phase we provide a novel non-trivial adaptation of the implementation for $\mathcal{EL}$ ontologies from the literature [24] that poses questions to LLMs, instead of using their synthetic teacher, to methodically extract and organize knowledge so to actively re-construct ontologies. To evaluate our approach, we present results showing the performance of LLMs in determining whether concept inclusions created by an ontology engineer on prototypical $\mathcal{EL}$ ontologies are "true" or "false". This phase uses membership and equivalence queries to iteratively refine the hypothesis ontology. The LLM is queried with specific concept inclusions, and based on the responses, the hypothesis ontology is updated.

The performance are evaluated based on their ability to correctly determine whether the concept inclusions created by an ontology engineer on prototypical $\mathcal{EL}$ ontologies are "true" or "false". Metrics such as precision, recall, and F1-score are used to measure the accuracy of the learned ontologies.

Additionally, the logical consistency of the responses and the efficiency of the learning process are assessed.

Chapter 2

Large Language Models

In recent years, the field of natural language processing (NLP) [56] has witnessed unprecedented progress, largely fueled by the development of LLMs. These models have demonstrated the ability to understand, generate, and manipulate human-like text with remarkable fluency and coherence. The emergence of LLMs has transformed various NLP tasks, including but not limited to language translation, text summarization, sentiment analysis, and question answering.

This introduction serves as a comprehensive overview of LLMs, aiming to provide insights into their architecture, training methodologies and real-world applications.

2.1 Evolution of Large Language Models

The concept of LLMs originated in the 1960s with the creation of Eliza [91]. Eliza's introduction initiated the exploration of NLP, laying the groundwork for more sophisticated LLMs in the future. Nearly three decades later, in 1997, Long Short-Term Memory (LSTM) [37] networks were developed. These networks enabled the creation of deeper and more intricate neural networks capable of processing larger volumes of data. Initial models showcased the potential for text comprehension and generation, but significant advancements were made with the development of architectures that could handle extensive datasets and parameters. In particular, the advent of the transformer architecture revolutionized the field of NLP.

The introduction of the Transformer architecture [88] represented a notable departure from conventional recurrent and convolutional neural network designs. Transformers leverage self-attention mechanisms, enabling them to dynamically assess the importance of various input tokens. This capability empowers transformers to efficiently capture extensive dependencies within text sequences, rendering them highly proficient in tasks such as language modeling.

An influential model built upon the transformer architecture is the Generative Pre-trained Transformer (GPT) series, pioneered by OpenAI [70]. Spanning from GPT-1 to the formidable GPT-4 [62], this series has established unprecedented standards in model scale, data utilization, and performance across various NLP benchmarks [95, 35, 18, 49, 19].

Moreover, the progression of LLMs has been defined by incremental enhancements in architecture, training methodologies, and the magnitude of data and computational resources employed during training. These developments have resulted in exponential advancements in the capabilities of LLMs and their suitability for addressing a wide array of real-world challenges.

In this chapter, we delve into the main crucial aspects for understanding LLMs, offering insights into essential components and methodologies. We direct interested readers to the original literature for in-depth exploration.

2.2 Architecture of Large Language Models

At the core of Large Language Models rests the transformer architecture, which has become widely accepted as the standard approach for handling sequential data, especially within the realm of comprehending and generating natural language.

This architectural framework encompasses the following fundamental components:

- **Input Embeddings**: the input to a Transformer model typically comprises a sequence of tokens, such as words or subwords, represented as embeddings. Each token undergoes a mapping process to a high-dimensional vector space through an embedding layer, encoding semantic information associated with the token.

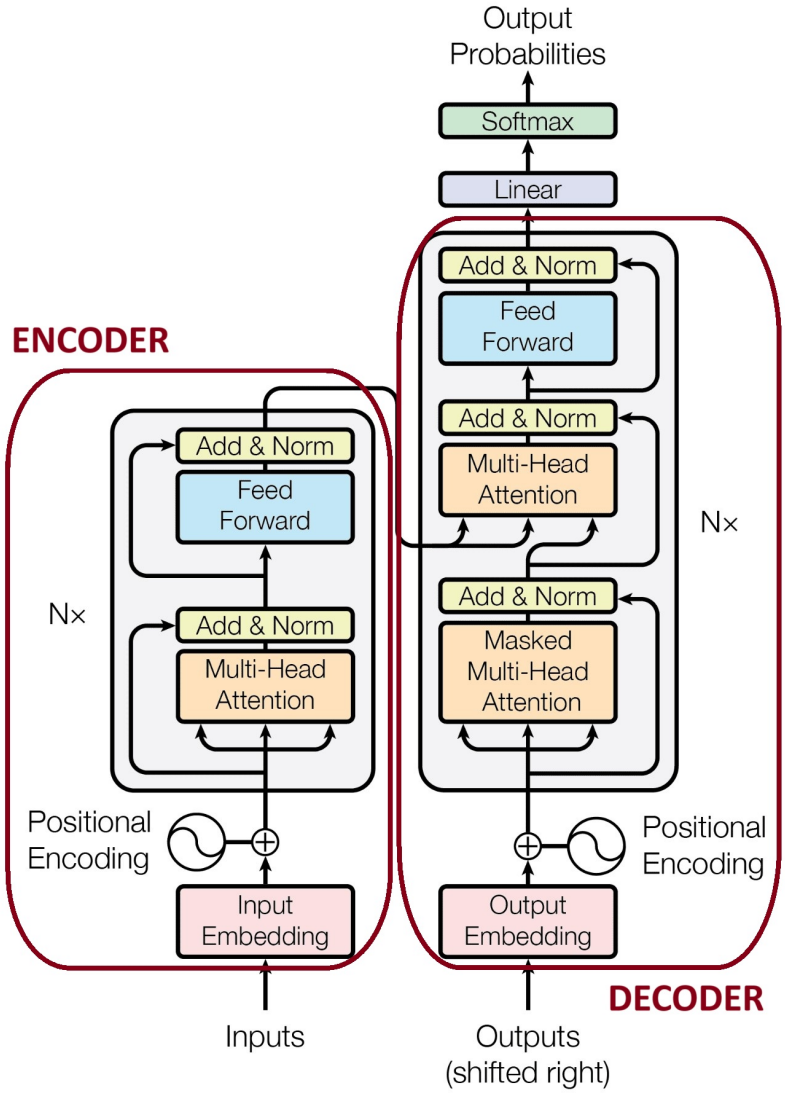

Figure 2.1: The encoder-decoder structure of the Transformer architecture. Taken from [88].

- **Positional Encodings**: unlike recurrent neural networks, Transformers do not inherently retain the sequential order of tokens in the input sequence. To overcome this, positional encodings augment token embeddings with positional embedding vectors. These encodings convey positional data, enabling the model to retain the sequential order within the sequence. Variants such as absolute, relative, or learned positional encodings, including Alibi [67] and RoPE [79], address this limitation.

- **Transformer Encoder and Decoder**: [88] the original Transformer model consists of an encoder and a decoder. The encoder processes input tokens to produce vectors of equal length. Meanwhile, the decoder predicts the target sequence token by token, utilizing the encoder's output. It incorporates cross-attention layers to consider both input and output sequences. Additionally, the decoder's self-attention layers use causal masking to prevent attending to future tokens during prediction. This architecture is commonly known as the encoder-decoder (ED). Prominent models like BART [48] and T5 [71] employ this architecture. T5, upon which T0 [75] is based, achieved impressive zero-shot generalization through extensive multitask fine-tuning, outperforming larger decoder-only models.

- **Self-Attention Mechanism**: central to the Transformer architecture is the self-attention mechanism, which enables the model to discern the significance of various input tokens based on their interrelationships. Each input token is associated with query, key, and value vectors, which are utilized to compute attention scores between token pairs. These scores are subsequently employed to calculate weighted sums of the corresponding values, allowing the model to focus on pertinent segments of the input sequence during token processing.

- **Multi-Head Attention**: transformers employ multi-head attention mechanisms to capture diverse aspects of the input sequence. This entails applying the self-attention mechanism across multiple parallel instances, each featuring distinct sets of query, key, and value vectors. The outputs from these attention heads are consolidated and subjected to linear transformations to

generate the final output.

- **Feed-Forward Neural Networks**: in addition to self-attention layers, Transformers incorporate feed-forward neural networks (FFNNs) at each position within the encoder and decoder. These FFNNs consist of two linear transformations separated by a non-linear activation function, such as the Rectified Linear Unit (ReLU) [57]. They facilitate the model's ability to capture intricate interactions among tokens and capture higher-order dependencies within the input sequence.

- **Layer Normalization and Residual Connections**: transformers integrate layer normalization and residual connections within each layer to enhance training stability and gradient flow. Layer normalization normalizes layer activations across the feature dimension, while residual connections enable gradients to propagate more efficiently by circumventing individual layers during training.

2.2.1 Variants of the transformer architectures

Variants of transformer architectures are:

- **Encoder-only:** an encoder-only transformer architecture, in contrast to the traditional Transformer model, excludes the decoder component. This setup focuses solely on processing input sequences without generating output sequences. The model's task is to understand and encode the input data, capturing its features and representations. Encoder-only transformers are suitable for tasks such as feature extraction, sentence embedding, and classification, where understanding input data is paramount, and output generation is not necessary. By concentrating solely on encoding, these models excel at extracting meaningful information from input sequences, facilitating downstream tasks such as classification or retrieval. Google's language representation model BERT [23] is based on this architecture.

- **Decoder-only:** a decoder-only transformer design, also referred to as an auto-regressive transformer, is a deviation from the traditional Transformer

model by removing the encoder part. It operates by producing output sequences relying exclusively on tokens it has previously generated, without any input sequence input. This setup proves highly effective for language generation, text completion, and sequence-to-sequence tasks. It excels at predicting subsequent tokens in a sequence, leveraging the context provided by preceding tokens. This narrow focus on decoding enables the model to grasp intricate relationships over long spans, resulting in the creation of coherent and contextually appropriate output. OpenAI GPTs series [70, 78, 16, 62] and many other recent high-performing LLMs [96, 43, 77] are based on this architecture.

- **Non-Casual Decoder:** the architecture proposes adjusting the attention mask of decoder-only models to create more detailed representations of input/conditioning text. Unlike traditional causal masking, this approach allows attention to the entire input sequence, similar to the original proposed encoder. Known as a prefix language model, this concept was initially introduced by Liu et al. [50] and later explored by Raffel et al. [71] and Wu et al. [93]. Despite demonstrating similar single-task finetuning performance to encoder-decoder models, it has seen limited adoption in the literature.

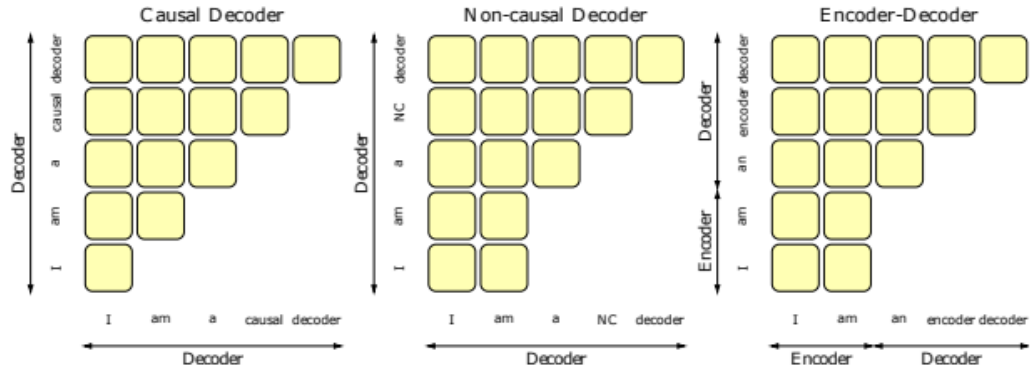

Figure 2.2: An example of attention patterns in language models, image is taken from [59]

2.3 Attention in LLMs

Attention mechanisms allocate weights to input tokens based on their significance, enabling prioritization of relevant tokens. In transformer-based LLMs, attention computes query, key, and value mappings for input sequences, facilitating attention score derivation. Various strategies like self-attention [88] and cross-attention [10], with enhancements like sparse [17] and flash attention [21], improve computational efficiency.

2.4 Activation Functions

Activation functions are crucial in LLMs and neural networks for several reasons. Firstly, they introduce non-linearity into the model, allowing it to learn and represent more complex patterns in the data. Without non-linear activation functions, a neural network would essentially become a linear model, no matter how many layers it has, limiting its ability to capture intricate patterns and dependencies.

Secondly, activation functions affect how gradients are propagated through the network during training. Proper gradient flow is essential for effective learning. Activation functions like the ReLU help in mitigating the vanishing gradient problem [36], where gradients become too small for effective learning in deep networks.

Activation functions enable the construction of deep networks that can learn hierarchical representations of data. Each layer can learn to extract increasingly complex features, crucial for tasks such as language modeling, where understanding context and nuances is essential.

Some activation functions, like ReLU, induce sparsity in the network by setting negative values to zero. This sparsity can act as a form of regularization, reducing the risk of overfitting and improving generalization to new data.

Activation functions like Gaussian Error Linear Unit (GELU) [34], and Swish [25] provide smooth and differentiable approximations, aiding in better convergence during training. This smoothness helps in maintaining stable gradients and avoiding issues like exploding gradients.

The choice of activation function can significantly impact the empirical performance of LLMs. For example in models like GPT-3 and GPT-4, activation

functions like GELU are used to ensure smooth gradient flow and efficient learning, which are essential for handling the vast amounts of data and the complexity of language tasks these models are trained on.

Here a summary of those three:

ReLU: $f(x) = \max(0, x)$

- Advantages: reduces the vanishing gradient problem; computationally efficient.
- Disadvantages: can suffer from the dying ReLU problem [51].

GELU: $f(x) = x \cdot \Phi(x)$

- Advantages: combines properties of dropout and ReLU; often improves performance on NLP tasks.
- Disadvantages: more computationally intensive than ReLU.

Swish: $f(x) = x \cdot sigmoid(x)$

- Advantages: often outperforms ReLU in deeper networks [72]; self-gated and flexible.
- Disadvantages: slightly slower to compute due to complexity.

2.5 Layer Normalization

Layer normalization stabilizes training dynamics, fostering faster convergence. Techniques like LayerNorm [5], RMSNorm [97], and pre-layer normalization [8], address training stability issues in LLMs.

2.6 Data Preprocessing

Data Preprocessing techniques are crucial for LLMs training, prepairing the raw text data by standardizing the input, thus reducing complexity for the model. Key techniques include: Data Cleaning, Parsing, Normalization, Tokenization [90] and Stemming or Lemmatization.

2.6.1 Data Cleaning

It is an essential step in pre-processing data for training LLMs. It involves identifying and correcting inaccuracies, inconsistencies, and extraneous components in raw text data. Typical data cleansing actions include eliminating duplicates, addressing missing or faulty values, and rectifying format discrepancies.

Further, specific text-related cleaning tasks like removing special characters, punctuation, and stop words are undertaken to refine the input text. Through extensive data cleansing practices, the quality and reliability of the training data are greatly enhanced, setting a strong foundation for further phases of LLMs training.

Key Methods:

- **Handling Missing Values**: when some observations or features in a dataset lack data, potentially leading to skewed predictions or a biased model. Techniques to address missing values include impuatation, where missing values are filled in based on other data points, and deletion, where rows or columns with missing values are removed. The choice of method depends on the nature of the data and the extent of the missing values. The choice of technique depends on the data's characteristics and the missing data's extent.

- **Noise Reduction**: irrelevant or random data that can obscure the actual patterns, causing inaccurate predictions. Sources of noise might be human error, malfunctioning devices, or irrelevant attributes. Noise reduction methods include binning, which involves sorting data into groups and smoothing it, and regression, which fits data to a curve or line, both aiding in reducing data variability and enhancing the model's predictive performance.

- **Consistency Checks**: vital to ensure that data throughout a dataset follows consistent formats, rules, or conventions. Inconsistencies can arise from data entry mistakes, differing data sources, or system errors, which can distort training results. Techniques such as cross-field validation, which checks the consistency of combined fields, and checks for duplicate records are used to identify and rectify these discrepancies.

- **Deduplication**: duplicates in data can arise from various sources such as data entry mistakes, dataset mergers, or system errors, skewing data distribu-

tion and affecting model training. Techniques like record linkage, which associates similar records, and exact matching, which identifies identical records, are employed. Removing duplicates makes the dataset more precise and representative, thereby improving the LLMs performance.

2.6.2 Parsing

It involves analyzing data syntax to extract meaningful information. This extracted information serves as input for the LLM. Parsing deals with structured data sources like XML, JSON, or HTML. In the context of NLP parsing refers to identifying the grammatical structure of a sentence or phrase. This can be helpful for tasks like machine translation, text summarization, and sentiment analysis.

Parsing also extracts information from semi-structured or unstructured data sources like email messages, social media posts, or web pages that can be used for tasks like topic modeling, entity recognition, and relation extraction.

2.6.3 Normalization

It is a crucial pre-processing technique for standardizing textual data to ensure uniformity and consistency in language usage and minimize complexity for NLP models. This process involves converting text to a common case, typically lowercase, to eliminate variations arising from capitalization.

Normalization also includes standardizing numerical data, dates, and other non-textual elements to render the entire dataset into a coherent and homogeneous format, facilitating more effective training and improved generalization capabilities for LLMs. Normalization helps reduce the vocabulary size and model complexity, which can improve overall performance and accuracy.

2.6.4 Tokenization

It involves breaking down text into smaller units, called tokens, which can be words, subwords, or characters. Tokenization creates a structured and manageable input for the model. By breaking down the text into tokens, the model gains a

granular understanding of language usage and syntax that enables it to analyze and generate coherent sequences of words.

Tokenization also facilitates the creation of vocabulary and word embeddings pivotal for the model's language comprehension and generation capabilities. This process forms the basis for text processing for LLMs, laying the groundwork for effective language modeling and natural language understanding.

2.6.5 Stemming and Lemmatization

They are two text preprocessing techniques used in NLP to reduce words to their base or root form. While both techniques aim to simplify text analysis and improve performance in tasks like search and data retrieval, they do so in slightly different ways.

Stemming

Stemming involves chopping off the ends of words in the hope of achieving this goal correctly most of the time. It typically removes prefixes and suffixes from words, leaving what is often called the "stem." For example, the stem of the word "running" is "run," and the stemming process removes the ending "ning." Charateristics:

- It is a crude heuristic that chops off word endings based on common patterns.
- It can be faster than lemmatization because it uses simple rules and doesn't require a detailed dictionary.
- The result might not be an actual word but just a root form.
- Common algorithms include the Porter, Snowball, and Lancaster stemmers, each with its own set of rules and intended language coverage.

Lemmatization

Lemmatization involves a more sophisticated analysis of a word to remove inflectional endings only and to return the base or dictionary form of a word, known as

the "lemma." For example, the lemma of "was" is "be," and the lemma of "mice" is "mouse." Characteristics:

- It uses vocabulary and morphological analysis of words, which makes it slower than stemming.
- Requires understanding of the part of speech (POS) as the same word can have multiple lemmas based on how it is used.
- Generally returns a real word which is always a correct lexical item in the language.

2.7 Training Techniques

Training LLMs comprises two main phases: pre-training and fine-tuning. In the pre-training stage, the model learns from a vast amount of text data through unsupervised learning methods [89] like masked language modeling or next-sentence prediction. The primary goal here is to equip the model with deep insights into language patterns and semantics directly from raw text. Throughout pre-training, the model encounters diverse linguistic contexts and structures, fostering a thorough grasp of language intricacies. This phase typically entails optimizing complex neural network architectures via parallelized computations on hardware accelerators such as GPUs or TPUs. Once pre-training is completed, the model proceeds to fine-tuning for specific downstream tasks using annotated datasets. Fine-tuning entails further training the pre-trained model with supervised learning objectives tailored to the target tasks. This process facilitates the model in adjusting its parameters to suit the specifics of the given task, thereby enhancing its performance and adaptability. Fine-tuning encompasses various methodologies, including transfer learning [16], where insights gained during pre-training are leveraged for the target task, and only task-specific parameters are adjusted. Through fine-tuning, LLMs can attain cutting-edge performance across a broad spectrum of NLP tasks such as text classification, named entity recognition, and machine translation.

2.8 Utilization of Large Language Models

Multiple applications across various domains and NLP tasks showcase LLMs versatility and efficacy in addressing real-world challenges. Notable applications are:

- **Text Generation**: LLMs possess the capability to produce coherent and contextually relevant text for endeavors such as story generation, dialogue systems, and creative writing. These models exhibit proficiency in generating high-quality text spanning diverse genres and styles, rendering them invaluable tools for content creation and enhancement.

- **Language Translation**: LLMs demonstrate remarkable prowess in language translation endeavors, facilitating the seamless conversion between multiple languages with high accuracy and fluency. Through the utilization of extensive pre-training and fine-tuning methodologies, LLMs adeptly capture cross-lingual semantics and syntactic structures, thereby enabling precise translation between languages characterized by substantial linguistic disparities.

- **Sentiment Analysis**: LLMs are extensively employed in sentiment analysis tasks, aimed at discerning the sentiment or emotional disposition conveyed within textual content. Sentiment analysis holds relevance across diverse domains, including social media monitoring, customer feedback analysis, and market research. Leveraging their capabilities, LLMs accurately categorize text into positive, negative, or neutral sentiments, empowering businesses to glean insights into customer sentiments and preferences.

- **Question Answering**: LLMs excel in question answering endeavors, which entail furnishing precise and contextually pertinent responses to user inquiries based on provided passages or documents. Question answering constitutes a foundational NLP task with applications spanning information retrieval, virtual assistants, and customer support. Leveraging their extensive pre-trained representations, LLMs adeptly comprehend intricate questions and furnish accurate responses, thereby attaining state-of-the-art performance on benchmark datasets.

2.9 Libraries

Commonly used libraries for LLMs training include Transformers [84], DeepSpeed [22], Megatron-LM [52], JAX [40], Colossal-AI [20] and FastMoE [26]. Frameworks such as MindSpore [54], PyTorch [69], Tensorflow [80], and MXNet [55] help LLMs development.

Chapter 3

Introduction to Description Logic

3.1 Overview

DL [28, 58] is a formal knowledge representation language used to describe and reason about the conceptualization of a domain. DL provides a set of constructs to represent knowledge in a structured and logical manner, allowing for precise and expressive modeling of concepts and their relationships. In this chapter, we will explore the fundamental concepts of DL, its syntax and semantics, reasoning mechanisms, and its applications in various domains.

3.2 Theory

3.2.1 Concepts

In DL, concepts represent sets of individuals that share common characteristics or properties. Concepts are used to describe the types or classes of objects in a domain. For example, in a medical ontology, concepts like *Person*, *Patient*, and *Doctor* can be defined to represent different types of individuals.

3.2.2 Roles

Roles (also known as properties or relations) define binary relationships between individuals or concepts. Roles are used to specify how individuals are related

to each other. For instance, in a family ontology, roles such as *hasParent* and *hasChild* can be defined to represent parent-child relationships.

3.2.3 Axioms

Axioms are logical statements that define the relationships between concepts and roles in a domain. Axioms are used to specify constraints and rules that govern the domain. Common types of axioms include concept inclusion axioms (e.g., $A \sqsubseteq B$), role inclusion axioms (e.g., $R \sqsubseteq S$), and equality axioms (e.g., $A \equiv B$).

3.2.4 Syntax

The syntax is defined by a set of symbols and rules for constructing valid expressions. The syntax includes symbols for concepts, roles and individuals.

Concepts and roles can be combined using various constructors to form more complex expressions:

- **Atomic concepts and roles**: basic building blocks, e.g., Human, parentOf.
- **Conjunction** ($\sqcap$): $C \sqcap D$ (intersection of C and D).
- **Disjunction** ($\sqcup$): $C \sqcup D$ (union of C and D).
- **Negation** ($\neg$): $\neg C$ (complement of C).
- **Existential quantification** ($\exists$): $\exists R.C$ (individuals related by R to some individual in C).
- **Universal quantification** ($\forall$): $\forall R.C$ (individuals related by R only to individuals in C).
- **Number restrictions**: $\geq n\ R.C$ and $\leq n\ R.C$ (number constraints on relations).

3.2.5 Semantics

The semantics specify the meaning of DL expressions in terms of interpretations and models. An interpretation assigns meanings to the symbols used in a DL ontology, while a model satisfies the axioms and constraints specified in the ontology.

The semantics of DL are based on set theory and model theory.

Interpretation: consists of a non-empty domain of individuals and an interpretation function.

Interpretation Function: maps every concept to a subset of the domain and every role to a subset of the domain's Cartesian product (pairs of individuals).

The semantic rules determine whether a particular interpretation satisfies (or models) a description logic statement:

Satisfaction: an interpretation satisfies a concept expression if it correctly maps individuals to the subsets defined by the concept expression.

Model: an interpretation is a model of a knowledge base if it satisfies all the statements (axioms) in the knowledge base.

Constructor	Syntax	Semantics
Top concept	$\top$	Δ^I (the entire domain)
Bottom concept	$\bot$	$\emptyset$ (the empty set)
Atomic concept	A	$A^I \subseteq \Delta^I$
Atomic role	R	$R^I \subseteq \Delta^I \times \Delta^I$
Individual	a	$a^I \in \Delta^I$
Negation	$\neg C$	$\Delta^I \setminus C^I$
Conjunction	$C \sqcap D$	$C^I \cap D^I$
Disjunction	$C \sqcup D$	$C^I \cup D^I$
Existential restriction	$\exists R.C$	$\{x \in \Delta^I \mid \exists y \in \Delta^I((x,y) \in R^I \wedge y \in C^I)\}$
Universal restriction	$\forall R.C$	$\{x \in \Delta^I \mid \forall y \in \Delta^I((x,y) \in R^I \rightarrow y \in C^I)\}$
Nominal	$\{a\}$	$\{a^I\}$
Role inverse	R^-	$\{(y,x) \mid (x,y) \in R^I\}$
Value restriction	$\forall R.C$	$\{x \in \Delta^I \mid \forall y((x,y) \in R^I \rightarrow y \in C^I)\}$

Table 3.1: Syntax and Semantics of Description Logic

Axiom Type	Syntax	Meaning
Concept Inclusion	$C \sqsubseteq D$	$C^I \subseteq D^I$
Concept Equality	$C \equiv D$	$C^I = D^I$
Role Inclusion	$R \sqsubseteq S$	$R^I \subseteq S^I$
Role Equality	$R \equiv S$	$R^I = S^I$
Individual Assertion	$a : C$	$a^I \in C^I$
Role Assertion	$(a,b) : R$	$(a^I, b^I) \in R^I$
Negated Concept Assertion	$a : \neg C$	$a^I \notin C^I$
Negated Role Assertion	$(a,b) : \neg R$	$(a^I, b^I) \notin R^I$

Table 3.2: Notation of Axioms in Description Logic

3.3 Reasoning in Description Logic

In the realm of DL, reasoning involves various computational procedures to analyze the structure and content of a knowledge base. It is a key feature of DL that allows one to derive implicit knowledge from the explicitly stated information. Several key reasoning processes are integral to the effective management and utilization of ontological knowledge. These processes are crucial for the organization, consistency verification, and inferential capabilities of DL systems, particularly

in complex domains like biomedical informatics, semantic web technologies, and artificial intelligence planning.

1. **Subsumption**: this process involves determining whether all instances of one concept (denoted as A) necessarily belong to another concept (denoted as B). Formally, this relationship is expressed as $A \sqsubseteq B$. Subsumption is fundamental to understanding and establishing the hierarchical structure of knowledge bases, enabling inferential reasoning such as transitivity, where if $A \sqsubseteq B$ and $B \sqsubseteq C$, then it logically follows that $A \sqsubseteq C$.

2. **Consistency Checking**: this procedure ascertains that there are no contradictions within the assertions and axioms of a DL ontology. An ontology is deemed consistent if there exists at least one model that satisfies all its stipulated axioms. For example, an ontology that includes the axiom stating "no person can be both a child and a parent simultaneously" must verify, through consistency checking, that no individual is assigned both roles.

3. **Instance Checking**: this is the verification process to determine whether a specific individual (denoted as x) qualifies as an instance of a particular concept (denoted as C). Instance checking involves evaluating whether the properties and relations of x satisfy all criteria defined for C. If x adheres to all conditions of C, then x is confirmed as an instance of C.

4. **Classification**: this process involves the systematic categorization of all known individuals into concepts based on their respective properties and relationships. It also includes identifying and delineating the interrelations among concepts themselves. Classification is pivotal for the construction and refinement of an ontology's taxonomy, facilitating an organized structure where new or modified concepts are appropriately integrated.

5. **Query Answering**: this function exploits the structured knowledge within an ontology to respond to sophisticated queries. A DL reasoner can provide information on the satisfiability of concepts (i.e., whether they can possibly possess instances), verify the consistency of the entire ontology, and identify individuals that meet specified criteria. For instance, a query may seek to

determine the existence of individuals who fulfill the roles of both "teacher" and "researcher" and to elucidate additional properties they might share.

3.4 Applications of Description Logic

DL are applied across a wide array of fields including ontology engineering, knowledge representation, semantic web technologies, biomedical informatics, and NLP. These logical frameworks facilitate the modeling, reasoning, data integration, and interoperability across complex domains, while also bolstering automated reasoning and decision-making systems.

In ontology engineering, DL transcends the role of merely constructing ontological structures. It is instrumental in the maintenance and evolution of these structures, ensuring their robustness and relevance over time. This aspect is particularly crucial in areas like the semantic web, which continuously adapt to evolving standards and technologies. DL provides the flexibility needed for ontologies to evolve and accommodate new web functionalities, making it a foundational component in developing adaptive ontological frameworks.

For semantic web technologies, DL-based ontologies play a pivotal role in semantic annotation. This involves enriching web content with semantic metadata to make the information machine-readable, thereby enhancing search and retrieval capabilities. Such annotations are critical for refining the user experience on the web by enabling more precise and context-aware search results, which are tailored to the semantic content of user queries.

In biomedical informatics, DL proves essential in harmonizing the integration of varied biomedical databases that often feature incompatible data formats and terminologies. By establishing a common ontological framework, DL ensures seamless data integration, which is vital for advancing research in areas like genomics, proteomics, and personalized medicine. For instance, using DL to synchronize different genetic databases can significantly enhance the accuracy of genetic linkage analysis, a key factor in understanding complex genetic disorders.

NLP benefits markedly from DL through enhanced semantic parsing capabilities. DL helps in disambiguating language constructs, thus enabling machines to process human languages with higher accuracy. This improvement is crucial

for developing sophisticated applications such as automated dialogue systems and advanced language models, which rely on DL to facilitate more effective communication between humans and machines. These applications exemplify how DL can extend its utility beyond traditional data-oriented tasks to involve complex language interactions, bridging the gap between human linguistic nuances and machine processing capabilities.

Chapter 4

Ontologies

4.1 Introduction

In the realm of knowledge representation, ontologies [86] serve for organizing information. They play a crucial role in facilitating understanding, sharing, and reuse across various domains. This chapter explores the significance of ontologies in knowledge representation, delving into their structure, uses, and applications. By providing a common vocabulary and a structured representation of knowledge, ontologies enable diverse systems and organizations to communicate information efficiently and effectively.

4.2 Understanding Ontologies

An ontology is a conceptual model that captures the essential entities, attributes, and relationships within a specific domain. It defines a common vocabulary for researchers and practitioners to describe and interact with the domain, specifying the semantics of the concepts and their interrelations. Ontologies are composed of classes (concepts), properties (attributes), and relationships (associations) that are typically organized hierarchically to facilitate complex information retrieval, inference, and data integration tasks.

4.3 Methodologies for Ontology Development

Ontology development methodologies [27] are crucial frameworks guiding the systematic construction of ontologies. These methodologies ensure that the resulting ontologies are robust, relevant, and capable of supporting user and application-specific needs. Common methodologies include:

- **Top-Down Approach:** this method begins at the general level and refines into more specific detail. It is suitable when the domain is well-understood and a clear hierarchy of concepts exists.
- **Bottom-Up Approach:** starts with specific concepts and aggregates them into general categories, used when specific data or existing systems are in place.
- **Middle-Out Approach:** combines elements of both top-down and bottom-up, starting from a well-understood middle layer of abstraction.
- **Methontology:** includes phases like feasibility study, specification, conceptualization, formalization, implementation, and maintenance.
- **Ontology Learning:** involves automatically generating elements of the ontology from existing data sources using machine learning and natural language processing techniques.

4.4 Types of Ontologies

Ontologies are classified based on their scope, expressiveness, and purpose into several types, each serving different aspects of knowledge representation:

1. **Domain Ontologies:** focus on a specific domain, such as medicine, finance, or biology, capturing domain-specific knowledge and concepts. Examples include the Gene Ontology [4] which provides a framework for the representation of genetics information across species.

2. **Upper-level Ontologies:** offer general concepts that are not specific to a particular domain but are reusable across multiple domains. These include DOLCE [13], BFO [63], and SUMO [66], which provide abstract categories like time, space, and event.

3. **Task-specific Ontologies:** designed to support specific tasks, such as medical diagnosis or natural language processing, these ontologies are tailored to enhance the performance of task-specific algorithms and applications.

4.5 Ontology Languages and Tools

To create, edit, and manage ontologies, several formal languages and software tools have been developed. These include:

- **RDF (Resource Description Framework)** [45]: a standard model for data interchange on the Web.

- **OWL (Web Ontology Language)** [11]: a comprehensive ontology language designed for creating and sharing ontologies on the Internet.

- **RDFS (RDF Schema)** [15]: provides basic constructs for defining ontologies.

Tools such as **Protégé** [68], **OntoStudio** [61], and **TopBraid Composer** [82] offer user-friendly graphical interfaces and utilities for ontology development, enabling both experts and novices to construct and modify ontological models.

4.6 Applications of Ontologies

Ontologies are instrumental in several key areas:

- **Semantic Web:** they enhance web content with rich, structured knowledge, making it possible for machines to understand and respond to complex user queries [60].

- **Data Integration:** ontologies facilitate the integration and interoperability among heterogeneous data sources, improving data quality and consistency across different systems [81].
- **Knowledge Management:** they play a critical role in organizing and managing domain-specific knowledge, making it accessible and reusable.
- **NLP:** ontologies enhance semantic analysis in NLP tasks like information retrieval, question answering, and knowledge extraction [56].

4.7 Integration with Other Technologies

Ontologies are increasingly integrated with various technologies to enhance functionality and effectiveness:

- **Machine Learning:** provide a structured knowledge base for training machine learning models [30].
- **Blockchain:** facilitate standardization of terms across different stakeholders, enhancing transparency and interoperability [44].
- **Internet of Things (IoT):** help in managing and semantically processing the vast amount of data generated by IoT devices [9].

4.8 Challenges and Future Directions

While ontologies offer numerous benefits, they also pose significant challenges:

- **Ontology Maintenance:** updating ontologies to reflect new knowledge and changes in the domain remains a complex and resource-intensive task.
- **Scalability:** developing methods to efficiently handle large-scale ontologies is crucial as the volume of information grows.
- **Interoperability:** ensuring that different ontologies can work together seamlessly across various applications and domains is critical.

- **Ontology Alignment:** aligning and integrating ontologies from different domains to avoid redundancy and conflicts is an ongoing issue.

Future research will need to address these challenges, focusing on automated ontology learning, dynamic ontology updates, and the development of standards for semantic interoperability to enhance the utility and scalability of ontological systems.

Chapter 5

Ontology Learning

Ontology learning is aiming to automate the process of ontology development. As ontologies play a critical role in knowledge representation, their construction has traditionally required extensive manual effort, involving domain experts and knowledge engineers. However, the vast and ever-growing amount of information available today, particularly on the web, calls for automated systems that can learn and update ontologies dynamically.

Ontology learning leverages techniques from machine learning, natural language processing, and data mining to extract conceptual knowledge structures from diverse data sources. These sources include structured databases, unstructured text, and semi-structured documents. The primary goal of ontology learning is to simplify and scale the ontology construction process by automatically identifying relevant concepts, attributes, and relationships inherent in the data.

The process of ontology learning typically involves several key phases:

- **Text Processing and Extraction:** this phase deals with extracting useful information from raw data, which often involves tasks such as tokenization, part-of-speech tagging, and named entity recognition.
- **Concept Identification:** using algorithms to detect and define the key concepts within the data. This often involves clustering similar terms or phrases that refer to the same underlying idea.
- **Hierarchy Building:** structuring the identified concepts into a hierarchi-

cal framework that reflects their relationships, often using techniques like hierarchical clustering.

- **Relationship and Attribute Extraction:** identifying how concepts are related to each other and what attributes they possess, which often involves pattern recognition and statistical inference.

- **Ontology Evaluation:** the final ontology is evaluated to ensure its accuracy and relevance, using measures like precision, recall, and domain expert validation.

5.1 The ExactLearner Paradigm

Central to ontology learning is a robust communication model that orchestrates the collaboration between ontology engineers and domain experts. While ontology engineers possess the technical acumen to formalize ontologies, domain experts harbor profound domain knowledge but may lack expertise in formal ontology construction. To formalize this interaction, ExactLearner adopt a communication model grounded in several fundamental assumptions:

- **Perfect Domain Knowledge**: domain experts possess comprehensive domain knowledge but lack the ability to directly formalize ontologies.

- **Shared Vocabulary**: domain experts can communicate the ontology's vocabulary, encompassing concept and role names, to ontology engineers.

- **Membership Queries**: ontology engineers can pose queries to domain experts to ascertain if specific concept inclusions are entailed by the ontology.

- **Equivalence Queries**: ontology engineers can verify if a constructed ontology is complete by asking domain experts for counterexamples.

This model sets the stage for exploring the feasibility of constructing target ontologies using polynomial queries or, ideally, within polynomial time, relative to the ontology's size and the provided counterexamples.

5.1.1 Exact Learning Framework

Aligned with Angluin's learning algorithm [2], the exact learner paradigm employs membership and equivalence queries to systematically construct ontologies. The framework transcends assumptions about the articulation clarity of domain experts, ensuring robustness across varied levels of expertise. Building upon prior research, an algorithm for learning $\mathcal{EL}$ (Description Logic Expressive) terminologies is proposed, demonstrating exponential time complexity in concept expression and vocabulary size. Despite its computational demands, ExactLearner, implementing this algorithm, showcases success in terminating for small and medium-sized ontologies, thereby validating its efficacy in ontology construction.

Given a class of ontologies L (e.g., all ontologies in a specific DL, such as $\mathcal{EL}$ terminologies), the goal is to precisely identify a target ontology $O \in L$ by making queries to an oracle. It is assumed that the learner knows the vocabulary of the target ontology Σ_O.

- **Membership Query:** this involves asking the oracle any type of valid axiom e.g. whether an inclusion $C \sqsubseteq D$ is entailed by O ($O \models C \sqsubseteq D$) where C and D are any type of expression (see Section 3.2) in the DL's vocabulary Σ_O. If $O \models C \sqsubseteq D$, the inclusion is a positive example with respect to the target ontology O; otherwise, it is a negative example.

- **Equivalence Query:** this involves asking the oracle whether a hypothesis ontology H is equivalent to the target ontology O. If they are equivalent, the oracle answers "yes". If not, the oracle provides either a positive example $C \sqsubseteq D$ where $H \not\models C \sqsubseteq D$ or a negative example $E \sqsubseteq F$ where $H \models E \sqsubseteq F$. These examples are referred to as positive and negative counterexamples, respectively.

A class of ontologies L is considered exactly learnable if there exists an algorithm that, for any target ontology $O \in L$, halts and computes a hypothesis $H \in L$ using membership and equivalence queries, such that $H \equiv O$. If this algorithm operates in polynomial time, the class is deemed exactly learnable in polynomial time. Specifically, at each computational step, the time taken is bounded by a

polynomial $p(|O|, |C \sqsubseteq D|)$, where O represents the target ontology and $C \sqsubseteq D$ is the largest counterexample encountered.

The subsumption learning framework aims to precisely identify target ontologies within a specified class of ontologies. By utilizing membership and equivalence queries, learners navigate through the ontology space, striving to accurately pinpoint the target ontology. While classes such as $\mathcal{EL}_{lhs}$ and $\mathcal{EL}_{rhs}$ demonstrate polynomial time learnability, the broader class of $\mathcal{EL}$ ontologies presents challenges for polynomial time learning, reflecting the intricate nature of ontology space exploration. Research by Konev et al. [47] confirms that $\mathcal{EL}_{lhs}$ and $\mathcal{EL}_{rhs}$ are indeed exactly learnable in polynomial time, while the entire class of $\mathcal{EL}$ ontologies does not lend itself to polynomial time learning.

5.2 Learning Algorithm

Designed for the exact learning of $\mathcal{EL}$ terminologies, the learning process exhibits exponential time complexity in the size of the largest concept expression in the ontology, denoted as $|C_O|$, and the size of the ontology vocabulary, $|\Sigma_O|$. However, it does not depend on the size of the entire ontology.

- The learner consistently engages with an oracle by posing equivalence queries.
- A positive response from the oracle, indicating "yes," means that the hypothesis H aligns with the ontology O, leading to the conclusion of the algorithm and the return of H.
- If the oracle returns a counterexample, represented as $C \sqsubseteq D$, the algorithm adapts its hypothesis H accordingly.

5.2.1 Process

- If C' is a concept name, the algorithm integrates D' into the right-hand side of all inclusions in H that have C' on the left, thereby computing a right O-essential counterexample.

- Conversely, if D' is a concept name, a left O-essential counterexample is computed.

5.2.2 Right O-essential Counterexamples

- **Concept Saturation for** O: if $O \models A \sqsubseteq C'$, where C' results from C by including a concept name A' at some node, then $A \sqsubseteq C$ is replaced by $A \sqsubseteq C'$.
- **Sibling Merging for** O: if $O \models A \sqsubseteq C'$, with C' emerging from merging two r-successors in C, then update $A \sqsubseteq C$ to $A \sqsubseteq C'$.
- **Decomposition on the Right for** O: consider d' an r-successor of d in C, with A' in d's node label, and $O \models A' \sqsubseteq \exists r.C_{d'}$, then modify $A \sqsubseteq C$ as follows:
  - $A' \sqsubseteq \exists r.C_{d'}$
  - $A \sqsubseteq C| - d$ if $H \not\models A' \sqsubseteq \exists r.C_{d'}$;
  - Decompose further as $d' \downarrow$, defining C_d as the subtree rooted at d and $C| - d \downarrow$ as the subtree resulting from d's removal.

5.2.3 Left O-essential Counterexamples

- **Concept Saturation for** H: if $H \models C \sqsubseteq C'$ and C' arises by adding a concept name A' to some node in C, then replace $C \sqsubseteq A$ with $C' \sqsubseteq A$.
- **Decomposition on the Left for** O: for a non-root node d such that $O \models C| - d \downarrow \sqsubseteq A'$ and $H \not\models C| - d \downarrow \sqsubseteq A'$, modify $C \sqsubseteq A$ as:
  - $C| - d \downarrow \sqsubseteq A'$
  - $C_d \sqsubseteq A'$ if $O \models C_d \sqsubseteq A'$ and $H \not\models C_d \sqsubseteq A'$.

5.3 Key Observations

- Throughout the process, the maintained hypothesis H always satisfies $O \models H$. Therefore, the counterexamples provided are invariably positive.

- Considering that O represents a terminology, complex expressions C and D in counterexamples are constrained to relationships expressible through concept names, discernible via membership queries.

5.4 Explanation

- Upon receiving a positive counterexample $C \sqsubseteq D$, the algorithm simplifies it into a form $C' \sqsubseteq D'$ where either C' or D' is a concept name.
- This transformation leverages membership queries and the efficiency of this operation is bounded by polynomial time based on the sizes of the hypothesis H, the concept C, and the vocabulary Σ_O.

Algorithm 1 The Learning Algorithm for EL

Require: An EL terminology O given to the oracle; Σ_O given to the learner.
Ensure: An EL terminology H computed by the learner such that $O \equiv H$.

Initialization:
Set $H = \{A \sqsubseteq B \mid O \models A \sqsubseteq B, A, B \in \Sigma_O\}$.
while $H \not\equiv O$ **do**
  While $H \not\equiv O$:
  Get Counterexample:
  Let $C \sqsubseteq D$ be the returned positive counterexample for O relative to H.
  Compute $C' \sqsubseteq D'$:
  Compute $C' \sqsubseteq D'$ with C' or D' in $\Sigma_O \cap NC$ (where NC is the set of concept names).
  Determine O-essential Concept:
  if $C' \in \Sigma_O \cap NC$ **then**
    Compute a right O-essential α from $C' \sqsubseteq D'$.
  else
    if $C' \sqsubseteq F' \in H$ **then**
      Compute a left O-essential α from $C' \sqsubseteq D'$.
    end if
  end if
  Update Hypothesis:
  Add α to H.
end while
return Hypothesis: H

Chapter 6

Probably Approximately Correct Learning Algorithm

6.1 Introduction

In the field of machine learning, Probably Approximately Correct (PAC) [87, 33, 92] learning framework lays down theoretical foundations for understanding how learning happens from data. Developed by Leslie Valiant in the late 1980s, PAC learning provides insights into the feasibility and constraints of learning algorithms. This chapter aims to explain the principles, mechanisms, and outcomes of PAC learning algorithms.

6.2 Foundations of PAC Learning

PAC learning revolves around the concept of sample complexity and computational efficiency. Essentially, it addresses the question: given a hypothesis class H, a target concept c, and a probability distribution D over the instance space, what is the minimum number of samples needed for a learning algorithm to produce a hypothesis h that closely matches the target concept with high confidence?

$$\forall \epsilon > 0, \delta > 0, \exists A \mid \mathrm{p}\left(\mathrm{error}_D(h) \leq \epsilon\right) \geq 1 - \delta \tag{6.1}$$

Where:

- ϵ is the error tolerance parameter, representing the maximum acceptable error of the hypothesis h;
- δ is the confidence parameter, representing the probability that the algorithm's output hypothesis h has an error within ϵ;
- $\text{p}(\text{error}_D(h) \leq \epsilon)$ denotes the probability that the error of hypothesis h with respect to the distribution D is less than or equal to ϵ;
- $\text{error}_D(h)$ represents the error of hypothesis h measured concerning the distribution D;
- A is the learning algorithm.

Explained, a concept class C is considered PAC-learnable if, for any $\epsilon > 0$ and $\delta > 0$, there exists a learning algorithm that, with a probability of at least $1 - \delta$, outputs a hypothesis h such that the error of h (measured concerning the distribution D) is no more than ϵ.

The critical parameters in PAC learning are sample complexity (the required number of samples for learning) and computational complexity (the time and space demands of the learning algorithm).

6.3 The PAC Learning Algorithm

The PAC learning algorithm operates by iteratively refining a hypothesis based on observed data. The general procedure involves these steps:

1. **Initialization**: start with initializing the hypothesis space H and setting the error tolerance parameters ϵ and δ.
2. **Sample Collection**: collect a sufficient number of labeled examples from the distribution D.
3. **Hypothesis Selection**: choose a hypothesis h from the hypothesis space H that best fits the observed data. This step often entails optimization methods such as gradient descent [74] or convex optimization [14].

4. **Error Estimation**: estimate the error of the selected hypothesis h using techniques such as cross-validation or holdout validation.

5. **Convergence Check**: verify whether the error of the hypothesis h meets the desired accuracy threshold ϵ. If not, repeat steps 2-4 with additional samples or a refined hypothesis space.

6. **Output**: finally, output the hypothesis h that fulfills the PAC criteria with high confidence.

6.4 Properties of PAC Learning

PAC learning algorithms possess several distinguishing properties:

1. **Sample Complexity**: PAC learning provides bounds on the minimum number of samples required for learning. These bounds rely on factors like the complexity of the hypothesis class, the desired accuracy level ϵ, and the confidence parameter δ.

2. **Computational Efficiency**: PAC learning algorithms aim for learning with polynomial time and space complexity. However, the computational efficiency often depends on the complexity of the hypothesis class and the optimization techniques used.

3. **Generalization Guarantees**: PAC learning offers theoretical assurances on the generalization performance of learned hypotheses. A hypothesis learned via a PAC algorithm is expected to generalize well to unseen data sampled from the same distribution D.

4. **Model Complexity**: the complexity of the hypothesis class H influences the sample complexity and generalization performance of PAC learning algorithms. A more complex hypothesis class may necessitate more samples for learning but could potentially capture intricate data patterns.

6.5 Applications and Extensions

PAC learning finds applications across diverse domains, including classification, regression, and reinforcement learning. Its theoretical underpinnings have spurred various extensions and variations, such as online learning, active learning, and learning with partial feedback. Additionally, PAC learning principles have influenced the development of advanced machine learning techniques, including deep learning and ensemble methods.

Chapter 7

Methodology

This chapter outlines the procedures and techniques employed in actively learning ontologies from LLMs using an active learning approach. The primary goal is to evaluate the performance of various LLMs when tasked with determining the truth value of concept inclusions within predefined ontologies. This section describes the experimental setup, the ontologies used, the LLMs evaluated, and the metrics for assessing performance.

7.1 First Phase of Experiments

The experiments consist in performing a number of membership queries with multiple LLMs on prototypical ontologies. These are small ontologies taken from ontology repositories and used for experiments in the ExactLearner project [24][1] [2], which focuses on $\mathcal{EL}$ ontologies. In all ontologies considered, the logical closure is finite. We consider the following ontologies:

1. *Animals:* contains knowledge related to the animal realm, including actual animals, subphyla, classes, orders, etc. The ontology has 12 (explicit) logical

[1] `https://github.com/bkonev/ExactLearner/`

[2] Generations, University, and Cell were also part of the Protégé Ontology Library. Not maintained anymore at `https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library` but still accessible via web archive at `https://web.archive.org/web/20210226123540/https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library`

axioms in $\mathcal{EL}$ and 20 logical axioms in the logical closure (that is, taking into account inferred axioms).

2. *Cell:* provides information about different cells based on their type, development stage and organism. The ontology has 24 logical axioms in $\mathcal{EL}$ and 24 in the logical closure.

3. *Football:* is a minimal ontology that describes the relations between football game, teams, players and managers. It has 9 logical axioms in $\mathcal{EL}$ and 12 in the logical closure.

4. *Generations:* describes the members and relations within a family. This ontology has 18 (explicit) logical axioms in $\mathcal{EL}$ and 42 in the logical closure.

5. *University:* is a small ontology, focusing on the professor role, with 4 logical axioms in the logical axioms in $\mathcal{EL}$ and 8 in the logical closure.

We use a total of 5 LLMs: Open AI's GPT 3.5 Turbo [16], Mistral [1], Mixtral [41] and two Llama 2 [83] models (we use Ollama's API[3]). Both Mistral and Mixtral are open models. Llama 2 is free of charge for research while GPT can be expensive as it charges for each query[4]. For each logical axiom in an ontology, we generated a membership query to an LLM using the Manchester OWL syntax. The goal was to assess how accurately an LLM could answer membership queries in different domains without fine-tuning. Accuracy was defined as answering 'true' for axioms present in the ontology.

Next, we generated all inferred axioms using the HermiT [31] reasoner and repeated the experiments with the new ontologies. Probing the LLMs on ontologies with inferred axioms tested their logical consistency. While it was possible that the LLMs had encountered these ontologies during their training, it was unlikely for the inferred axioms, as they were not explicitly present in the original ontologies.

In a third experiment, we employed a naive learning algorithm to actively learn ontologies. We asked membership queries for all concept inclusions of the

[3] `https://github.com/ollama/ollama`

[4] The source code of the experiments is publicly available `https://github.com/MatteoMagnini/ExactLearner`

form $A \sqsubseteq B$ with A and B as concept names within a given signature. The results of these experiments are presented in 8.3.

7.2 Second Phase of Experiments

What can we learn from LLMs? And, given that they can provide false information, is there an automated way to discover whether responses are incorrect or at least inconsistent? To explore these questions, this work investigates an active learning approach to learn *description logic ontologies* from LLMs. Our approach is based on Angluin's exact learning framework [3], where a learner attempts to learn target knowledge from a teacher by posing questions. In this method we consider different LLMs to be the teacher and studies the case in which the knowledge to be learned is expressed as an ontology in the classical $\mathcal{EL}$ description logic [6]. $\mathcal{EL}$ ontologies allow for expressions involving conjunctions and existential quantification. The following example illustrates a concept inclusion in $\mathcal{EL}$ and its translation into controlled natural language.

$$\mathsf{Conifer} \sqsubseteq \mathsf{Gymnosperm} \sqcap \exists \mathsf{bears.Cone}$$

Conifers can be considered a subcategory of Gymnosperms that bear cones.

Exact learning of lightweight description logic ontologies has been studied for theoretical purposes [47, 46, 64]. The only known implementation for exact learning involves a synthetic teacher [24] and assume the ontology submitted must be exclusively of type $\mathcal{EL}$, created for testing using ontologies from the Oxford Ontology Repository [5]. With the advancement of LLMs, we can now investigate the extraction of description logic ontologies from LLMs within Angluin's active learning framework [2].

The following ontologies are considered:

1. *Animals*, *Cell*, *Football*, *Generations* and *University* as showed before in Section 7.1

[5] `https://www.cs.ox.ac.uk/isg/ontologies/`

2. *Bio-primitive:* has information about different kinds of measures in the biological domain such as weight measure, blood pressure and specific metrics (e.g., Hamilton anxiety score). The ontology has 122 logical axioms (no new axioms are inferred by the engine).

3. *Biosphere:* is a rich ontology providing information about animals and plants. The ontology has 86 logical axioms (no new axioms are inferred by the engine).

We use 5 open LLMs: Mistral with 7 billion parameters, Mixtral with 47 billion parameters, Llama2 with 7 and 13 billion parameters and the very recent LLama3 with 8 billion parameters [53] (we use again Ollama's API). We try two different formalism for the queries: the Manchester OWL syntax and the natural language. Finally, we use two different system prompts to provide contextual information to LLMs.

```
Answer with only True or False.
```

This first prompt is minimal and it simply suggests the LLM to answer with a single word, i.e., "True" or "False".

```
You need to classify the following statements as True or False.
The statement will be provided in either Manchester OWL syntax or
natural language. Follow strictly these guidelines:
1. answer with only True or False;
2. entities from different categories are not in a subclass
   relationship;
3. statements or question containing numerous entities are most
   probably False;
4. take a deep breath before answering;
5. if you are unsure about the classification, answer with False.
```

The second prompt instead provides much more context information and tells the LLM to follow some guidelines. Point 1 is the same sentence used in the first prompt. Point 2 is a remark for the LLM to stress out that there could be concept names that refer to different categories and cannot be put in a subclass relation.

Animals	Bio-primitive	Biosphere	Cell	Football	Generation	University
1,082	17,629	11,508	2,242	655	1,669	274

Table 7.1: Minimum number of axioms needed to PAC learn an ontology.

For instance, "Fish" (animal) lives in "Water" (environment) and the two concept names should not be in a subclass relationship like "Fish" $\sqsubseteq$ "Water". Point 3 has the purpose to prevent the LLM to answer "True" to long queries (i.e., queries involving several concept and role names). Long queries may appear when the learner tries to apply the rules in 5.2. Point 4 is an expedient that has been found to be effective to improve LLM performance [94]. The last point is a final remark to be conservative in the replies.

Concerning the simulation of equivalence query using the PAC framework, we set to 0.1 the value of ϵ and we set to 0.2 the value of γ. We report in 7.1 the minimum number of axioms needed to PAC learn the chosen ontologies.

7.2.1 PAC for equivalence queries

Equivalence queries are problematic, it is not feasible, nor reasonable, to ask an LLM if the hypothesis ontology is equivalent to the target ontology. First, because the teacher does not directly have the target ontology at its disposal, and second, because the LLM could struggle to handle a really long input. For these reasons we simulate equivalence queries with the PAC framework.

As mentioned in Chapter 6, PAC utilizes a labeled set of examples drawn from the instance space X using an unknown probability distribution D and attempts to find a hypothesis $h \in H$ where H is a set of possible hypotheses that a learning algorithm considers when trying to approximate the target concept.

The size or complexity of H directly impacts the sample complexity m, which is the number of training examples required to ensure that the learning algorithm can – with high probability – find a hypothesis that is approximately correct. The minimum number of samples needed to PAC learn:

$$m \geq \frac{\log\left(|H|/\gamma\right)}{\epsilon} \tag{7.1}$$

For a given target ontology, we estimate the instance space X by considering

all possible combinations of the following statements: $A \cap B \sqsubseteq C$, $B \sqsubseteq \exists r.A$, and $\exists r.A \sqsubseteq B$. We limit the instance space to these three statements because they are sufficient to cover non-trivial ontologies, in particular the ones used for the experiments.

Of course, one can choose different and additional statements (e.g., use not only statements with chains of length one) but in doing so they should keep into account the size of the input space since it heavily affects the computation.

We then consider the upper bound to the size of the hypothesis class H to be equal to the size of the instance space to the power of the size of the number of axioms in the target ontology:

$$|H| = |X|^{|A|} \tag{7.2}$$

where $|X|$ is the size of the instance space, and $|A|$ is the number of axioms.

An equivalence query can then be simulated by probing random samples from X. If a statement is not entailed by the hypothesis ontology and it is labeled as 'true' by the LLM, then there is no equivalence and the sample is used as a counterexample. Instead, if none counterexample can be found within m samples, we say that the equivalence query is 'true'.

7.2.2 From Manchester OWL Syntax to Natural Language

The Manchester OWL syntax is a formal language for describing ontologies and it is quite close to the natural language. However, because is it most likely that LLMs have been trained more on data in natural language rather than in other formalisms, they could perform better if the query is also presented in natural language. We now describe the translation from the Manchester OWL syntax to natural language that we used in our experiments. All queries in the Manchester OWL syntax are of the form "`[A] SubClassOf [B]`". We substitute it with "Can `[A]` be considered a subcategory of `[B]`? Answer only with yes or no.", where `[A]` and `[B]` are placeholders for the concept expressions on the left and right hand side of the concept inclusion. This corresponds to the third formulation by Funk et al. [29], chosen in this work as it is closer to our setting. The translation of OWL concept expressions into natural language is then recursively applied to `[A]`

and `[B]`.

1. If the query is of the form "`[A] and [B]`" then it is substituted with "`[A] that is also [B]`". The mapping is then recursively applied to `[A]` and `[B]`.

2. If the query is of the form "`[r] some [A]`", where `[r]` is a role name (or object property in OWL terminology), then it is substituted with "`something that [r] [A]`" and the substitution is recursively applied to `[A]`.

We provide examples for the procedure described above.

- If the input message is "Person SubClassOf Human", the function will return: "Can Person be considered a subcategory of Human?"

- If the input message is "Carnivore SubClassOf Animal and eats some Meat" The function will return: "Can Carnivore be considered a subcategory of Animal that is also something that eats Meat?"

- If the input message is "(Bird and Fish) SubClassOf Animal": The function will return: "Can Bird that is also Fish be considered a subcategory of Animal?"

7.3 OWL API Framework

OWL API [38] [6] was the running core of our experiments and is a framework designed for creating, parsing, manipulating, and serializing OWL Ontologies. Simple to use and versatile, the OWL API supports the creation and manipulation of OWL ontologies in various ways. Since version 3.1, it has been specifically tailored to support OWL 2, offering robust functionality to work with ontologies across different applications. Key Features:

- Provides a high-level API for handling OWL 2 ontologies.

- Includes an efficient in-memory implementation for managing ontologies.

[6] `https://github.com/owlcs/owlapi`

- Supports a variety of syntaxes including RDF/XML, OWL/XML, OWL Functional Syntax, Manchester OWL Syntax, Turtle, KRSS, and OBO Flat file formats.

- Facilitates integration with external reasoners like FaCT++ [85]+, HermiT, Pellet[76], ELK [42].

7.4 Problems with LLMs

When using GPT-3.5 Turbo we found some limitations from our paid version(Tier 1, $5 paid, up to $100/month)

- **Rate Limits:** the API imposes rate limits on the number of requests that can be made within a certain timeframe. This partially hind ourself in the course of the first experiments. The specific limits can be found in the OpenAI documentation [7].

- **Cost:** extensive use of GPT-3.5 Turbo can become expensive, especially for high-frequency tasks or large datasets. For this reason we didn't consider it as suitable for the second phase.

Similarly for open LLMs, limitations were found in the overloading of the server hosting our Ollama services, specific precautions are taken:

- **Sleep Intervals:** the system must wait (we found 100 milliseconds to be the sweet spot) between requests. This pause helps prevent overwhelming the server, ensuring stable and reliable performance.

- **Retry Mechanism:** requests are retried up to a maximum of three times if they fail. This ensures that transient issues do not cause permanent failures but also requires careful handling to avoid unnecessary delays.

[7] `https://platform.openai.com/docs/guides/rate-limits/usage-tiers?context=tier-one`

7.5 Caching LLMs' Answer

Caching responses from LLMs is a crucial optimization strategy. It offers several benefits that enhance the efficiency and reliability of experimental workflows.

By caching responses, experiments can be resumed from the last saved state rather than restarting from scratch. This drastically reduces the time required to conduct experiments, especially those involving long-running processes, such as learning medium-sized ontologies. Efficient caching helps in better resource utilization, ensuring that computational power and time are not wasted on redundant calculations or data processing.

7.5.1 Advantages of Caching

- **Time Efficiency:** reduces the need to recompute responses for previously encountered inputs, saving significant time in iterative processes.
- **Resource Utilization:** helps in better management of computational resources, avoiding redundant usage and reducing operational costs.
- **Resilience:** in scenarios where experiments are interrupted due to unforeseen circumstances (e.g., system crashes or power failures), caching allows for seamless continuation without loss of progress.
- **Data Integrity:** mechanisms to validate the integrity of cached files are implemented, ensuring that only complete and correct data is retained. This reduces the likelihood of using corrupted or incomplete data in subsequent experiment runs.
- **Scalability:** facilitates scaling of experiments by allowing reuse of previously computed results, thereby enabling more extensive and comprehensive testing.

7.6 Probing Language Models

In this section we describe the main challenges encountered when probing LLMs with ontology axioms and how we handled them.

7.6.1 Input Format and Unexpected Responses

One important factor is the format of the query.

To systematically query an LLM with the goal of learning an ontology, it is useful to standardise the questions, so as to improve the transparency of the whole process and increase the level of reproducibility.

For the membership queries task, we investigate the use of the Manchester OWL syntax [39], as this is an ontology syntax designed to be closer to natural language. Another aspect to consider is that, in principle, there are no constraints in the answers returned by the language model. An LLM may answer with an arbitrary and unexpected response, even if the expected answer is just a single word like in the case of membership queries in the exact learning model. To mitigate this issue, one can explicitly tell the LLM to answer with 'true' or 'false'. This particular request can be done in the question itself (e.g., appending "Answer with 'true' or 'false'." after the query) or by exploiting some hyper-parameters of the API of the LLM. In the second case, one can use a *system prompt* – a.k.a., integrated text into each query within the chat session – to enrich the model with additional information and useful to harness the response. We highlight that there are also other hyper-parameters that could help driving the LLM's response into the desired format (e.g., maximum number of tokens, temperature, etc.). Even with all these precautions the model may return an unexpected response. For example:

1. the answer can have more text than just 'true' or 'false';

2. both 'true' and 'false' can appear in the answer;

3. the answer does not have 'true' nor 'false'.

While in the first scenario a trivial parsing would determine the correct classification, in the remaining cases, since there is some ambiguity, we considered a third value, which we called 'unknown'.

7.6.2 Correctness and Logical Consistency

We also need to deal with challenges regarding the correctness of the responses (assuming the format of the responses returned by the language model are as expected, see Section 7.6.1). Actively learning ontologies has been investigated for various fragments of $\mathcal{EL}$ [47, 64, 24], though, without using LLMs as teachers. If an LLM is playing the role of the teacher then there is no guarantee that the responses are correct [29] (in the sense of reflecting the 'truth' about the real world) and, moreover, that they are logically consistent with any $\mathcal{EL}$ ontology. Indeed, it is known that LLMs can learn statistical features instead of performing logical reasoning [98]. So, we need to consider the following kinds of errors:

1. $C \sqsubseteq D$ should be 'false' (cf. the real world) but the LLM answers 'true';
2. $C \sqsubseteq D$ should be 'true' (cf. the real world) but the LLM answers 'false';
3. all concept inclusions in $= \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$ are answered with 'true', $\models C \sqsubseteq D$ but $C \sqsubseteq D$ is classified as 'false'.

The last case is a logical inconsistency. One strategy to handle this issue is to consider the *closure* under logical consequence [12]. That is, in Point 3, one could consider $C \sqsubseteq D$ as 'true'.

Chapter 8

Results

8.1 First Phase of Experiments

The results for the membership queries on the original logical axioms are shown in Table 8.1. For the inferred axioms, the results are in Table 8.2. For the third experiment, where we tested active learning of ontologies by querying concept inclusions, the results are detailed in Table 8.3.

Models	Animals			University			Generations			Football			Cell		
	T	**F**	**U**	**T**	**F**	**U**	**T**	**F**	**U**	**T**	**F**	**U**	**T**	**F**	**U**
Mistral (7b)	9	1	2	2	0	2	5	10	3	7	2	0	17	1	6
Mixtral (47b)	11	1	0	4	0	0	3	6	9	9	0	0	15	9	0
Llama2 (7b)	11	1	0	4	0	0	16	1	1	9	0	0	24	0	0
Llama2 (13b)	11	1	0	4	0	0	16	1	1	9	0	0	23	1	0
Gpt3.5	10	2	0	4	0	0	13	4	1	9	0	0	21	3	0

Table 8.1: Results for the experiments testing correctness w.r.t. axioms in the ontologies. Labels T, F and U mean 'true', 'false' and 'unknown' responses count. We indicate the number of parameters in each model in parenthesis (e.g. Mistral has 7 billion). It is not known the number of parameters of GPT 3.5.

To determine the statistical significance of the results, we applied the Chi-squared [65] test to evaluate the relationship between the answers of the LLMs and the ontologies. The null hypothesis was that there is no correlation between the answers and the ontologies. We rejected the null hypothesis in every case (p-value lower than 0.05) except those highlighted in yellow (see Table 8.3).

Animals				University				Generations				Football				Cell			
T	**F**	**U**	**L**	**T**	**F**	**U**	**L**	**T**	**F**	**U**	**L**	**T**	**F**	**U**	**L**	**T**	**F**	**U**	**L**
14	2	4	2	5	1	2	0	10	27	5	2	9	3	0	0	18	1	5	0
18	2	0	0	8	0	0	0	19	13	10	0	12	0	0	0	17	7	0	0
20	0	0	0	8	0	0	0	40	1	1	1	12	0	0	0	24	0	0	0
18	2	0	1	7	1	0	0	35	6	1	4	11	1	0	1	21	3	0	0
20	0	0	0	7	1	0	0	36	5	1	0	12	0	0	0	18	6	0	0

Table 8.2: Results for the experiment testing logical consistency. The number of parameters of each model and the meaning of T, F, U are as in Table 8.1. L stands for logical inconsistencies (an axiom answered as 'false' or 'unknown' which can be inferred from the set of the axioms answered as True, see 7.6.2). Models' names omitted for better readability (they are the same of Table 8.1).

Animals			University			Generations			Football			Cell		
A	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**
0.87	0.52	0.72	0.57	0.67	0.5	0.84	0.71	0.23	0.74	0.44	0.65	0.65	0.48	0.81
0.89	0.57	0.69	0.57	0.48	0.92	0.82	0.64	0.66	0.72	0.43	0.76	0.7	0.32	0.64
0.51	0.2	1	0.24	0.24	1	0.4	0.22	0.88	0.21	0.21	1	0.27	0.18	1
0.73	0.31	0.94	0.45	0.3	0.92	0.63	0.32	0.74	0.44	0.26	0.88	0.44	0.21	0.91
0.71	0.3	1	0.69	0.44	1	0.74	0.41	1	0.68	0.4	1	0.61	0.28	0.91

Table 8.3: Results for the experiments testing negative examples. Labels A, P and R mean 'Accuracy', 'Precision' and 'Recall' respectively [32]. Models' names omitted for better readability (they are the same of Table 8.1).

8.2 Second Phase of Experiments

We evaluate the results of experiments by computing the following metrics: accuracy, precision, recall, and F1-score. The metrics are computed considering the total number of axioms is the size of the instance space $|X|$. Therefore, every possible depth-one axiom in the form of $A \cap B \sqsubseteq C$, $B \sqsubseteq \exists r.A$, and $\exists r.A \sqsubseteq B$. The axioms that are both (resp. not) entailed by the original ontology and the learned ontology are labeled as true positive (resp. true negative). Axioms entailed by the original ontology but not by the learned ontology are labeled as false negative, vice-versa they are labeled as false positive. Below the average table by Ontology 8.4, LLM Model 8.5 and Metric type 8.6, in the appendix 9.1 the whole results are displayed.

Ontology	Accuracy	Recall	Precision	F1-Score
Animals	0.499	0.396	0.993	0.5
Bio-primitive	0.438	0.292	1.0	0.358
Biosphere	0.443	0.235	1.0	0.302
Cell	0.444	0.399	1.0	0.526
Football	0.425	0.396	0.989	0.511
Generations	0.518	0.47	0.959	0.566
University	0.571	0.516	0.977	0.615

Table 8.4: Average Metrics by Ontology

Model	Accuracy	Recall	Precision	F1-Score
Llama2:13b	0.455	0.401	0.989	0.485
Llama2	0.17	0.17	1.0	0.282
Llama3	0.354	0.273	0.994	0.385
Mistral	0.68	0.476	0.977	0.58
Mixtral	0.724	0.613	0.981	0.681

Table 8.5: Average Metrics by Model

Metric Type	Accuracy	Recall	Precision	F1-Score
M. OWL Syntax	0.248	0.186	0.998	0.304
Natural Language	0.439	0.275	0.991	0.403
E. M. OWL Syntax	0.442	0.373	0.987	0.466
E. Natural Language	0.779	0.711	0.977	0.758

Table 8.6: Average Metrics by Metric Type

Chapter 9

Conclusion

This research presents an extensive evaluation of actively learning ontologies using various LLMs within Angluin's exact learning model, combined with the PAC learning framework. Our objective was to assess the ability of LLMs to determine the truth value of concept inclusions within predefined ontologies, using both structured syntax and natural language queries.

Key findings from our experiments include:

1. **Superior Performance of Mixtral**: mixtral model outperformed other models, likely due to its higher parameter count. This indicates that larger models may better understand complex relationships within ontologies.

2. **Natural Language Queries**: models achieved higher accuracy with natural language queries compared to Manchester OWL Syntax. This highlights the effectiveness of leveraging LLMs' natural language training.

3. **Impact of System Prompts**: using custom system prompts significantly improved performance, with gains ranging from 70% to 230%. Tailored prompts are crucial for guiding model behavior effectively.

4. **Challenges with Complex Ontologies**: ontologies with numerous object properties or specialized knowledge were more challenging to learn. Simpler ontologies, primarily consisting of concept names, were learned with higher accuracy, suggesting that complexity directly impacts learning feasibility.

5. **Logical Consistency**: ensuring logical consistency remains a challenge, as LLMs sometimes provided contradictory responses. Our approach to mitigating inconsistencies by considering logical closure proved partially effective.

These findings illuminate the nuanced dynamics of LLM performance across different contexts, showcasing both strengths and areas for improvement. Remarkably, these results were obtained without specific fine-tuning, demonstrating impressive zero-shot learning capabilities. Models like Mistral and Mixtral were competitive with GPT 3.5 Turbo and outperformed Llama 2 in many instances, especially on ontologies containing general knowledge such as Animals and Generations.

9.1 Future Work

Building on these promising results, future research should focus on several key areas:

1. **Model Fine-Tuning**: fine-tuning LLMs on domain-specific corpora and ontology data could enhance their performance and consistency. Future work should explore the impact of targeted fine-tuning on handling complex ontologies.

2. **Expanded Ontology Set**: extending experiments to include larger and more complex ontologies will help evaluate the scalability and robustness of the models. Incorporating a broader range of ontologies, including those with specialized domains, will provide a more comprehensive evaluation.

3. **Advanced Query Formulation**: investigating sophisticated methods for translating ontology axioms into natural language queries may improve model performance. Exploring alternative query formats and linguistic structures could better align with the models' training data.

4. **Enhanced System Prompts**: developing more refined and adaptive system prompts that adjust dynamically based on query context and model responses could further improve performance. Research into optimal prompt design is warranted.

5. **Addressing Logical Inconsistencies**: future work should focus on advanced techniques for handling logical inconsistencies in LLM responses. This could involve developing algorithms to detect and correct contradictions in real-time, ensuring greater reliability.

6. **Cross-Model Analysis**: comparative analyses across different LLM architectures and parameter sizes will provide deeper insights into performance factors. This can guide the development of more efficient models tailored for ontology learning tasks.

7. **Human-in-the-Loop Approaches**: integrating human expertise in the active learning process, where human annotators validate and refine models' responses, could enhance the accuracy and trustworthiness of learned ontologies.

By addressing these areas, future research can advance the application of LLMs in ontology learning, contributing to the development of more intelligent and autonomous systems for knowledge representation and management.

Bibliography

[1] Albert Q. Jiang et al. "Mistral 7B". In: *CoRR* abs/2310.06825 (2023). DOI: 10.48550/ARXIV.2310.06825. arXiv: 2310.06825.

[2] Dana Angluin. "Learning regular sets from queries and counterexamples". In: *Information and Computation* 75.2 (1987), pp. 87–106. ISSN: 0890-5401. DOI: 10.1016/0890-5401(87)90052-6. URL: https://www.sciencedirect.com/science/article/pii/0890540187900526.

[3] Dana Angluin. "Queries and Concept Learning". In: *Mach. Learn.* 2.4 (1987), pp. 319–342. DOI: 10.1007/BF00116828.

[4] Michael Ashburner et al. "Gene Ontology: tool for the unification of biology". In: *Nature Genetics* 25.1 (May 2000), pp. 25–29. ISSN: 1546-1718. DOI: 10.1038/75556. URL: https://doi.org/10.1038/75556.

[5] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *CoRR* abs/1607.06450 (2016). arXiv: 1607.06450. URL: http://arxiv.org/abs/1607.06450.

[6] Franz Baader, Sebastian Brandt, and Carsten Lutz. "Pushing the EL Envelope". In: *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*. Ed. by Leslie Pack Kaelbling and Alessandro Saffiotti. Professional Book Center, 2005, pp. 364–369. URL: http://ijcai.org/Proceedings/05/Papers/0372.pdf.

[7] Franz Baader et al. *An Introduction to Description Logic*. Cambridge University Press, 2017. ISBN: 978-0-521-69542-8. URL: http://www.cambridge.org/de/academic/subjects/computer-science/knowledge-management-

databases-and-data-mining/introduction-description-logic?format=PB#17zVGeWD2TZUeu6s.97.

[8] Alexei Baevski and Michael Auli. "Adaptive Input Representations for Neural Language Modeling". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=ByxZX20qFQ.

[9] Garvita Bajaj et al. "A study of existing Ontologies in the IoT-domain". In: *CoRR* abs/1707.00112 (2017). arXiv: 1707.00112. URL: http://arxiv.org/abs/1707.00112.

[10] Folco Bertini Baldassini et al. "Cross-Attention Watermarking of Large Language Models". In: vol. abs/2401.06829. 2024. DOI: 10.48550/ARXIV.2401.06829. arXiv: 2401.06829. URL: https://doi.org/10.48550/arXiv.2401.06829.

[11] Sean Bechhofer. "OWL: Web Ontology Language". In: *Encyclopedia of Database Systems, Second Edition*. Ed. by Ling Liu and M. Tamer Özsu. Springer, 2018. DOI: 10.1007/978-1-4614-8265-9_1073. URL: https://doi.org/10.1007/978-1-4614-8265-9%5C_1073.

[12] Sophie Blum et al. "Learning Horn envelopes via queries from language models". In: *International Journal of Approximate Reasoning* (2023), p. 109026. ISSN: 0888-613X. DOI: 10.1016/j.ijar.2023.109026.

[13] Stefano Borgo et al. *DOLCE: A Descriptive Ontology for Linguistic and Cognitive Engineering*. 2023. DOI: 10.48550/ARXIV.2308.01597. arXiv: 2308.01597. URL: https://doi.org/10.48550/arXiv.2308.01597.

[14] Stephen Boyd and Lieven Vandenberghe. "Gradient and Optimization". In: *Convex Optimization*. Cambridge University Press, 2004, pp. 83–127.

[15] Dan Brickley and R.V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. World Wide Web Consortium, Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[16] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[17] Rewon Child et al. *Generating Long Sequences with Sparse Transformers*. 2019. arXiv: 1904.10509. URL: http://arxiv.org/abs/1904.10509.

[18] François Chollet. "On the Measure of Intelligence". In: *CoRR* abs/1911.01547 (2019). arXiv: 1911.01547. URL: http://arxiv.org/abs/1911.01547.

[19] Karl Cobbe et al. "Training Verifiers to Solve Math Word Problems". In: *CoRR* abs/2110.14168 (2021). arXiv: 2110.14168. URL: https://arxiv.org/abs/2110.14168.

[20] *Colossal-AI*. https://github.com/fai-ia/ColossalAI.

[21] Tri Dao et al. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*. Ed. by Sanmi Koyejo et al. 2022. URL: http://papers.nips.cc/paper%5C_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html.

[22] *DeepSpeed*. https://github.com/microsoft/DeepSpeed.

[23] 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/V1/N19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

[24] Mario Ricardo Cruz Duarte, Boris Konev, and Ana Ozaki. "ExactLearner: A Tool for Exact Learning of EL Ontologies". In: *KR*. Ed. by Michael Thielscher, Francesca Toni, and Frank Wolter. AAAI Press, 2018, pp. 409–414. URL: https://aaai.org/ocs/index.php/KR/KR18/paper/view/18006.

[25] Steffen Eger, Paul Youssef, and Iryna Gurevych. "Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP tasks". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 4415–4424. DOI: 10.18653/V1/D18-1472. URL: https://doi.org/10.18653/v1/d18-1472.

[26] *FastMoE*. https://github.com/deepmind/fast-moe.

[27] Mariano Fernández-López and Asunción Gómez-Pérez. "Overview and analysis of methodologies for building ontologies". In: *Knowl. Eng. Rev.* 17.2 (2002), pp. 129–156. DOI: 10.1017/S0269888902000462. URL: https://doi.org/10.1017/S0269888902000462.

[28] Baade Franz. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003. URL: https://redirect.cs.umbc.edu/courses/graduate/691/fall17/01/papers/DescriptionLogicHandbook.pdf.

[29] Maurice Funk et al. "Towards Ontology Construction with Language Models". In: *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC) co-located with the 22nd International Semantic Web Conference (ISWC 2023), Athens, Greece, November 6, 2023*. Ed. by Simon Razniewski et al. Vol. 3577. CEUR Workshop Proceedings. CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3577/paper16.pdf.

[30] Sarah Ghidalia et al. "Combining Machine Learning and Ontology: A Systematic Literature Review". In: *CoRR* abs/2401.07744 (2024). DOI: 10.48550/ARXIV.2401.07744. arXiv: 2401.07744. URL: https://doi.org/10.48550/arXiv.2401.07744.

[31] Birte Glimm et al. "HermiT: An OWL 2 Reasoner". In: *J. Autom. Reason.* 53.3 (2014), pp. 245–269. DOI: 10.1007/S10817-014-9305-1. URL: https://doi.org/10.1007/s10817-014-9305-1.

[32] Margherita Grandini, Enrico Bagli, and Giorgio Visani. "Metrics for Multi-Class Classification: an Overview". In: *CoRR* abs/2008.05756 (2020). arXiv: 2008.05756. URL: https://arxiv.org/abs/2008.05756.

[33] David Haussler. "Overview of the probably approximately correct (PAC) learning framework". In: *Information and Computation* 100.1 (1992), pp. 78–150.

[34] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2023. DOI: 10.48550/arXiv.1606.08415.

[35] Dan Hendrycks et al. "Measuring Mathematical Problem Solving With the MATH Dataset". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*. Ed. by Joaquin Vanschoren and Sai-Kit Yeung. 2021. URL: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

[36] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* 6.2 (1998). DOI: 10.1142/S0218488598000094. URL: https://doi.org/10.1142/S0218488598000094.

[37] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf.

[38] Matthew Horridge and Sean Bechhofer. "The OWL API: A Java API for OWL ontologies". In: *Semantic Web* 2.1 (2011), pp. 11–21. DOI: 10.3233/SW-2011-0025. URL: https://doi.org/10.3233/SW-2011-0025.

[39] Matthew Horridge et al. "The Manchester OWL Syntax". In: *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006*. Ed. by Bernardo Cuenca Grau et al. Vol. 216. CEUR Workshop Proceedings. CEUR-WS.org, 2006. URL: https://ceur-ws.org/Vol-216/submission_9.pdf.

[40] *JAX*. https://github.com/google/jax.

[41] Albert Q. Jiang et al. "Mixtral of Experts". In: *CoRR* abs/2401.04088 (2024). DOI: 10.48550/ARXIV.2401.04088. arXiv: 2401.04088.

[42] Yevgeny Kazakov, Markus Krötzsch, and Frantisek Simancik. "ELK Reasoner: Architecture and Evaluation". In: *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012), Manchester, UK, July 1st, 2012*. Ed. by Ian Horrocks, Mikalai Yatskevich, and Ernesto Jiménez-Ruiz. Vol. 858. CEUR Workshop Proceedings. CEUR-WS.org, 2012. URL: https://ceur-ws.org/Vol-858/ore2012%5C_paper10.pdf.

[43] Boseop Kim et al. "What Changes Can Large-scale Language Models Bring? Intensive Study on HyperCLOVA: Billions-scale Korean Generative Pretrained Transformers". In: (2021). Ed. by Marie-Francine Moens et al., pp. 3405–3424. DOI: 10.18653/V1/2021.EMNLP-MAIN.274. URL: https://doi.org/10.18653/v1/2021.emnlp-main.274.

[44] Henry M. Kim, Marek Laskowski, and Ning Nan. "A First Step in the Co-Evolution of Blockchain and Ontologies: Towards Engineering an Ontology of Governance at the Blockchain Protocol Level". In: *CoRR* abs/1801.02027 (2018). arXiv: 1801.02027. URL: http://arxiv.org/abs/1801.02027.

[45] Graham Klyne and Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/ (visited on 03/15/2015).

[46] Boris Konev, Ana Ozaki, and Frank Wolter. "A Model for Learning Description Logic Ontologies Based on Exact Learning". In: *AAAI*. Ed. by Dale Schuurmans and Michael P. Wellman. AAAI Press, 2016, pp. 1008–1015. DOI: 10.1609/AAAI.V30I1.10087.

[47] Boris Konev et al. "Exact Learning of Lightweight Description Logic Ontologies". In: *J. Mach. Learn. Res.* 18 (2017), 201:1–201:63. URL: http://jmlr.org/papers/v18/16-256.html.

[48] Mike Lewis et al. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: (2020). Ed. by Dan Jurafsky et al., pp. 7871–7880. DOI: 10.18653/V1/2020.ACL-MAIN.703. URL: https://doi.org/10.18653/v1/2020.acl-main.703.

[49] Stephanie Lin, Jacob Hilton, and Owain Evans. *TruthfulQA: Measuring How Models Mimic Human Falsehoods*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Association for Computational Linguistics, 2022, pp. 3214–3252. DOI: 10.18653/V1/2022.ACL-LONG.229. URL: https://doi.org/10.18653/v1/2022.acl-long.229.

[50] Peter J. Liu et al. *Generating Wikipedia by Summarizing Long Sequences*. 2018. URL: https://openreview.net/forum?id=Hyg0vbWC-.

[51] Lu Lu et al. "Dying ReLU and Initialization: Theory and Numerical Examples". In: *CoRR* abs/1903.06733 (2019). arXiv: 1903.06733. URL: http://arxiv.org/abs/1903.06733.

[52] *Megatron-LM*. https://github.com/NVIDIA/Megatron-LM.

[53] Meta. *LLaMA-3*. https://llama.meta.com/llama3/. 2023.

[54] *MindSpore*. https://www.mindspore.cn/.

[55] *MXNet*. https://mxnet.apache.org/.

[56] Prakash M. Nadkarni, Lucila Ohno-Machado, and Wendy Webber Chapman. "Natural language processing: an introduction". In: *J. Am. Medical Informatics Assoc.* 18.5 (2011), pp. 544–551. DOI: 10.1136/AMIAJNL-2011-000464. URL: https://doi.org/10.1136/amiajnl-2011-000464.

[57] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. Ed. by Johannes Fürnkranz and Thorsten Joachims. Omnipress, 2010, pp. 807–814. URL: https://icml.cc/Conferences/2010/papers/432.pdf.

[58] Daniele Nardi and Ronald J. Brachman. "An Introduction to Description Logics". In: *The Description Logic Handbook: Theory, Implementation, and Applications*. Ed. by Franz Baader et al. Cambridge University Press, 2003, pp. 1–40.

[59] Humza Naveed et al. "A Comprehensive Overview of Large Language Models". In: *CoRR* abs/2307.06435 (2023). DOI: 10.48550/ARXIV.2307.06435. arXiv: 2307.06435. URL: https://doi.org/10.48550/arXiv.2307.06435.

[60] Natalya Fridman Noy et al. "BioPortal: A Web Repository for Biomedical Ontologies and Data Resources". In: *ISWC*. Ed. by Christian Bizer and Anupam Joshi. Vol. 401. CEUR Workshop Proceedings. CEUR-WS.org, 2008. URL: https://ceur-ws.org/Vol-401/iswc2008pd_submission_25.pdf.

[61] *OntoStudio*. https://www.ontoprise.de/en/home/products/ontostudio/.

[62] OpenAI. "GPT-4 Technical Report". In: *CoRR* abs/2303.08774 (2023). DOI: 10.48550/ARXIV.2303.08774. arXiv: 2303.08774. URL: https://doi.org/10.48550/arXiv.2303.08774.

[63] J. Neil Otte, John Beverley, and Alan Ruttenberg. "BFO: Basic Formal Ontology". In: *Appl. Ontology* 17.1 (2022), pp. 17–43. DOI: 10.3233/AO-220262. URL: https://doi.org/10.3233/AO-220262.

[64] Ana Ozaki, Cosimo Persia, and Andrea Mazzullo. "Learning Query Inseparable ELH Ontologies". In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 2959–2966. DOI: 10.1609/AAAI.V34I03.5688.

[65] Karl Pearson. "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50.302 (1900), pp. 157–175. DOI: 10.1080/14786440009463897.

[66] Adam Pease, Ian Niles, and John Li. "The suggested upper merged ontology: A large ontology for the semantic web and its applications". In: 28 (2002), pp. 7–10. URL: https://cdn.aaai.org/Workshops/2002/WS-02-11/WS02-11-011.pdf.

[67] Ofir Press, Noah A. Smith, and Mike Lewis. "Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=R8sQPpGCv0.

[68] *Protégé*. https://protege.stanford.edu/.

[69] *PyTorch*. https://pytorch.org/.

[70] Alec Radford et al. "Improving Language Understanding by Generative Pre-Training". In: (2018). URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[71] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. URL: http://jmlr.org/papers/v21/20-074.html.

[72] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. "Searching for Activation Functions". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=Hkuq2EkPf.

[73] Simon Razniewski et al., eds. *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC) co-located with the 22nd International Semantic Web Conference (ISWC 2023), Athens, Greece, November 6, 2023*. Vol. 3577. CEUR Workshop Proceedings. CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3577.

[74] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *Insight Centre for Data Analytics, NUI Galway Aylien Ltd., Dublin* (2016). URL: http://sebastianruder.com/optimizing-gradient-descent/index.html.

[75] Victor Sanh et al. "Multitask Prompted Training Enables Zero-Shot Task Generalization". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=9Vrb9D0WI4.

[76] Evren Sirin and Bijan Parsia. "Pellet: An OWL DL Reasoner". In: *Proceedings of the 2004 International Workshop on Description Logics (DL2004), Whistler, British Columbia, Canada, June 6-8, 2004*. Ed. by Volker Haarslev and Ralf Möller. Vol. 104. CEUR Workshop Proceedings. CEUR-WS.org, 2004. URL: https://ceur-ws.org/Vol-104/30Sirin-Parsia.pdf.

[77] Shaden Smith et al. "Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model". In: *CoRR* abs/2201.11990 (2022). arXiv: 2201.11990. URL: https://arxiv.org/abs/2201.11990.

[78] Irene Solaiman et al. "Release Strategies and the Social Impacts of Language Models". In: *CoRR* abs/1908.09203 (2019). arXiv: 1908.09203. URL: http://arxiv.org/abs/1908.09203.

[79] Jianlin Su et al. *RoFormer: Enhanced transformer with Rotary Position Embedding*. 2024. DOI: 10.1016/J.NEUCOM.2023.127063. URL: https://doi.org/10.1016/j.neucom.2023.127063.

[80] *TensorFlow*. https://www.tensorflow.org/.

[81] R. Thirumahal and G. Sudha Sadasivam. "Semantic integration of heterogeneous healthcare data based on hybrid root linked health record ontology". In: *Earth Sci. Informatics* 16.3 (2023), pp. 2661–2674. DOI: 10.1007/S12145-023-01055-Y. URL: https://doi.org/10.1007/s12145-023-01055-y.

[82] *TopBraid Composer*. https://www.topquadrant.com/tools/topbraid-composer/.

[83] Hugo Touvron et al. “Llama 2: Open Foundation and Fine-Tuned Chat Models”. In: *CoRR* abs/2307.09288 (2023). DOI: 10.48550/ARXIV.2307.09288. arXiv: 2307.09288.

[84] *Transformers*. https://huggingface.co/transformers/.

[85] Dmitry Tsarkov and Ian Horrocks. “FaCT++ Description Logic Reasoner: System Description”. In: *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*. Ed. by Ulrich Furbach and Natarajan Shankar. Vol. 4130. Lecture Notes in Computer Science. Springer, 2006, pp. 292–297. DOI: 10.1007/11814771_26. URL: https://doi.org/10.1007/11814771%5C_26.

[86] Mike Uschold and Michael Gruninger. “Ontologies: principles, methods and applications”. In: *Knowl. Eng. Rev.* 11.2 (1996), pp. 93–136. DOI: 10.1017/S0269888900007797. URL: https://doi.org/10.1017/S0269888900007797.

[87] Leslie Valiant. *Probably Approximately Correct: Nature’s Algorithms for Learning and Prospering in a Complex World*. Basic Books, 2013.

[88] Ashish Vaswani et al. *Attention Is All You Need*. 2023. DOI: 10.48550/arXiv.1706.03762.

[89] Thomas Wang et al. “What Language Model Architecture and Pretraining Objective Works Best for Zero-Shot Generalization?” In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 22964–22984. URL: https://proceedings.mlr.press/v162/wang22u.html.

[90] Jonathan J. Webster and Chunyu Kit. “Tokenization As The Initial Phase In NLP”. In: *14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992*. 1992, pp. 1106–1110. URL: https://aclanthology.org/C92-4173/.

[91] Joseph Weizenbaum. “ELIZA—a computer program for the study of natural language communication between man and machine”. In: *Commun. ACM* 9.1 (Jan. 1966), pp. 36–45. ISSN: 0001-0782. DOI: 10.1145/365153.365168.

[92] Wikipedia contributors. *Probably approximately correct learning — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Probably_approximately_correct_learning&oldid=1201095733. 2024.

[93] Shaohua Wu et al. "Yuan 1.0: Large-Scale Pre-trained Language Model in Zero-Shot and Few-Shot Learning". In: *CoRR* abs/2110.04725 (2021). arXiv: 2110.04725. URL: https://arxiv.org/abs/2110.04725.

[94] Chengrun Yang et al. "Large Language Models as Optimizers". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=Bb4VGOWELI.

[95] Rowan Zellers et al. *HellaSwag: Can a Machine Really Finish Your Sentence?* Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez. Association for Computational Linguistics, 2019, pp. 4791–4800. DOI: 10.18653/V1/P19-1472. URL: https://doi.org/10.18653/v1/p19-1472.

[96] Wei Zeng et al. "PanGu-α: Large-scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation". In: *CoRR* abs/2104.12369 (2021). arXiv: 2104.12369. URL: https://arxiv.org/abs/2104.12369.

[97] Biao Zhang and Rico Sennrich. "Root Mean Square Layer Normalization". In: (2019). Ed. by Hanna M. Wallach et al., pp. 12360–12371. URL: https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

[98] Honghua Zhang et al. "On the Paradox of Learning to Reason from Data". In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*. ijcai.org, 2023, pp. 3365–3373. DOI: 10.24963/ijcai.2023/375.

Acknowledgements

I would like to convey my deepest and sincerest gratitude to Professor Omicini for providing me the opportunity to pursue my master's thesis at the University of Oslo. Your support and availability have been crucial.

I am immensely thankful to Matteo Magnini and Professor Ana Ozaki for their exceptional support during my three months in Oslo. Your expertise, encouragement, and constant readiness to assist have significantly impacted my journey. I learned a lot during my time with both of you, and I am truly grateful for your mentorship and presence.

A heartfelt thank you to my family, whose unwavering love and support have laid the foundation for me to chase my dreams. Your belief in me and the sacrifices you have made are the cornerstone of my achievements, and I am forever indebted to you for enabling me to study and grow.

To my friends, thank you for being an essential part of this remarkable journey. Your companionship, laughter, and encouragement have enriched this experience profoundly. I treasure the moments we have shared, and I am saddened that this chapter is coming to an end; you will always hold a special place in my heart.

Lastly, I would like to extend my deepest appreciation to my girlfriend Sofia for her love, patience, understanding, and kindness during my time abroad. Despite our distance, your unwavering support has been a source of strength and comfort, helping me navigate through the challenges and triumphs of this journey.

Thank you all for your contributions to this achievement. Your support and love have made this possible, and I am eternally grateful.

Tables

Model	Prompt & Query	Accuracy	Recall	Precision	F1-Score
Llama2 (13b)	M. OWL Syntax	0.148	0.148	1.0	0.258
	Natural Language	0.394	0.196	0.999	0.328
	E. M. OWL Syntax	0.2	0.156	1.0	0.27
	E. Natural Language	0.995	0.981	0.987	0.984
Llama2 (7b)	M. OWL Syntax	0.148	0.148	1.0	0.258
	Natural Language	0.148	0.148	1.0	0.258
	E. M. OWL Syntax	0.148	0.148	1.0	0.258
	E. Natural Language	0.148	0.148	1.0	0.258
Llama3 (8b)	M. OWL Syntax	0.148	0.148	1.0	0.258
	Natural Language	0.264	0.167	0.999	0.287
	E. M. OWL Syntax	0.306	0.176	0.998	0.299
	E. Natural Language	0.956	0.775	0.993	0.87
Mistral (7b)	M. OWL Syntax	0.222	0.16	0.999	0.276
	Natural Language	0.867	0.528	0.952	0.68
	E. M. OWL Syntax	0.883	0.56	0.987	0.714
	E. Natural Language	0.95	0.752	0.987	0.853
Mixtral (47b)	M. OWL Syntax	0.202	0.156	0.999	0.271
	Natural Language	0.87	0.533	0.993	0.694
	E. M. OWL Syntax	0.979	0.888	0.985	0.934
	E. Natural Language	0.998	1.0	0.985	0.992

Table 1: Metrics for Animals ontology

Model	Prompt & Query	Accuracy	Recall	Precision	F1-Score
Llama2 (13b)	M. OWL Syntax	0.055	0.055	1.0	0.104
	Natural Language	0.103	0.058	1.0	0.109
	E. M. OWL Syntax	0.262	0.069	1.0	0.13
	E. Natural Language	1.0	1.0	1.0	1.0
Llama2 (7b)	M. OWL Syntax	0.055	0.055	1.0	0.104
	Natural Language	0.055	0.055	1.0	0.104
	E. M. OWL Syntax	0.055	0.055	1.0	0.104
	E. Natural Language	0.055	0.055	1.0	0.104
Llama3 (8b)	M. OWL Syntax	0.055	0.055	1.0	0.104
	Natural Language	0.063	0.055	1.0	0.105
	E. M. OWL Syntax	0.159	0.061	1.0	0.115
	E. Natural Language	0.979	0.72	1.0	0.837
Mistral (7b)	M. OWL Syntax	0.65	0.136	1.0	0.239
	Natural Language	0.93	0.44	1.0	0.611
	E. M. OWL Syntax	0.215	0.065	1.0	0.123
	E. Natural Language	1.0	1.0	1.0	1.0
Mixtral (47b)	M. OWL Syntax	0.286	0.071	1.0	0.133
	Natural Language	0.814	0.228	1.0	0.371
	E. M. OWL Syntax	0.971	0.652	1.0	0.789
	E. Natural Language	0.998	0.96	1.0	0.98

Table 2: Metrics for Bio-primitive ontology

Model	Prompt & Query	Accuracy	Recall	Precision	F1-Score
Llama2 (13b)	M. OWL Syntax	0.055	0.054	1.0	0.102
	Natural Language	0.506	0.098	0.999	0.179
	E. M. OWL Syntax	0.133	0.059	1.0	0.111
	E. Natural Language	0.997	0.946	1.0	0.972
Llama2 (7b)	M. OWL Syntax	0.054	0.054	1.0	0.102
	Natural Language	0.054	0.054	1.0	0.102
	E. M. OWL Syntax	0.054	0.054	1.0	0.102
	E. Natural Language	0.054	0.054	1.0	0.102
Llama3 (8b)	M. OWL Syntax	0.054	0.054	1.0	0.102
	Natural Language	0.51	0.099	1.0	0.18
	E. M. OWL Syntax	0.146	0.059	1.0	0.112
	E. Natural Language	0.726	0.164	1.0	0.282
Mistral (7b)	M. OWL Syntax	0.627	0.126	1.0	0.224
	Natural Language	0.691	0.148	1.0	0.258
	E. M. OWL Syntax	0.247	0.067	1.0	0.125
	E. Natural Language	0.915	0.387	1.0	0.558
Mixtral (47b)	M. OWL Syntax	0.283	0.07	1.0	0.131
	Natural Language	0.764	0.186	1.0	0.314
	E. M. OWL Syntax	0.998	0.964	1.0	0.982
	E. Natural Language	1.0	0.997	1.0	0.998

Table 3: Metrics for Biosphere ontology

Model	Prompt & Query	Accuracy	Recall	Precision	F1-Score
Llama2 (13b)	M. OWL Syntax	0.239	0.239	1.0	0.386
	Natural Language	0.239	0.239	1.0	0.386
	E. M. OWL Syntax	0.239	0.239	1.0	0.386
	E. Natural Language	1.0	1.0	1.0	1.0
Llama2 (7b)	M. OWL Syntax	0.239	0.239	1.0	0.386
	Natural Language	0.239	0.239	1.0	0.386
	E. M. OWL Syntax	0.239	0.239	1.0	0.386
	E. Natural Language	0.247	0.241	1.0	0.388
Llama3 (8b)	M. OWL Syntax	0.239	0.239	1.0	0.386
	Natural Language	0.239	0.239	1.0	0.386
	E. M. OWL Syntax	0.239	0.239	1.0	0.386
	E. Natural Language	0.446	0.302	1.0	0.464
Mistral (7b)	M. OWL Syntax	0.36	0.272	1.0	0.428
	Natural Language	0.594	0.371	1.0	0.541
	E. M. OWL Syntax	0.281	0.25	1.0	0.399
	E. Natural Language	0.986	0.944	1.0	0.971
Mixtral (47b)	M. OWL Syntax	0.239	0.239	1.0	0.386
	Natural Language	0.632	0.394	1.0	0.565
	E. M. OWL Syntax	0.945	0.813	1.0	0.897
	E. Natural Language	1.0	1.0	1.0	1.0

Table 4: Metrics for Cell ontology

Model	Prompt & Query	Accuracy	Recall	Precision	F1-Score
Llama2 (13b)	M. OWL Syntax	0.229	0.229	1.0	0.372
	Natural Language	0.319	0.251	1.0	0.402
	E. M. OWL Syntax	0.229	0.229	1.0	0.372
	E. Natural Language	0.97	0.903	0.971	0.936
Llama2 (7b)	M. OWL Syntax	0.229	0.229	1.0	0.372
	Natural Language	0.229	0.229	1.0	0.372
	E. M. OWL Syntax	0.229	0.229	1.0	0.372
	E. Natural Language	0.229	0.229	1.0	0.372
Llama3 (8b)	M. OWL Syntax	0.229	0.229	1.0	0.372
	Natural Language	0.229	0.229	1.0	0.372
	E. M. OWL Syntax	0.229	0.229	1.0	0.372
	E. Natural Language	0.422	0.282	0.989	0.439
Mistral (7b)	M. OWL Syntax	0.229	0.229	1.0	0.372
	Natural Language	0.229	0.229	1.0	0.372
	E. M. OWL Syntax	0.768	0.496	0.953	0.653
	E. Natural Language	0.989	1.0	0.953	0.976
Mixtral (47b)	M. OWL Syntax	0.229	0.229	1.0	0.372
	Natural Language	0.311	0.249	1.0	0.399
	E. M. OWL Syntax	0.989	1.0	0.953	0.976
	E. Natural Language	0.989	1.0	0.953	0.976

Table 5: Metrics for Football ontology

Model	Prompt & Query	Accuracy	Recall	Precision	F1-Score
Llama2 (13b)	M. OWL Syntax	0.234	0.234	1.0	0.379
	Natural Language	0.476	0.305	0.968	0.464
	E. M. OWL Syntax	0.279	0.244	0.991	0.391
	E. Natural Language	0.973	1.0	0.883	0.938
Llama2 (7b)	M. OWL Syntax	0.234	0.234	1.0	0.379
	Natural Language	0.234	0.234	1.0	0.379
	E. M. OWL Syntax	0.234	0.234	1.0	0.379
	E. Natural Language	0.234	0.234	1.0	0.379
Llama3 (8b)	M. OWL Syntax	0.234	0.234	1.0	0.379
	Natural Language	0.234	0.234	1.0	0.379
	E. M. OWL Syntax	0.234	0.234	1.0	0.379
	E. Natural Language	0.903	0.735	0.913	0.815
Mistral (7b)	M. OWL Syntax	0.714	0.447	0.939	0.605
	Natural Language	0.549	0.337	0.957	0.498
	E. M. OWL Syntax	0.964	0.953	0.89	0.921
	E. Natural Language	0.974	0.979	0.907	0.941
Mixtral (47b)	M. OWL Syntax	0.234	0.234	1.0	0.379
	Natural Language	0.472	0.303	0.968	0.462
	E. M. OWL Syntax	0.973	1.0	0.883	0.938
	E. Natural Language	0.973	1.0	0.883	0.938

Table 6: Metrics for Generations ontology

Model	Prompt & Query	Accuracy	Recall	Precision	F1-Score
Llama2 (13b)	M. OWL Syntax	0.231	0.231	1.0	0.375
	Natural Language	0.936	0.805	0.952	0.872
	E. M. OWL Syntax	0.301	0.247	0.986	0.395
	E. Natural Language	0.989	1.0	0.952	0.976
Llama2 (7b)	M. OWL Syntax	0.231	0.231	1.0	0.375
	Natural Language	0.231	0.231	1.0	0.375
	E. M. OWL Syntax	0.231	0.231	1.0	0.375
	E. Natural Language	0.231	0.231	1.0	0.375
Llama3 (8b)	M. OWL Syntax	0.231	0.231	1.0	0.375
	Natural Language	0.231	0.231	1.0	0.375
	E. M. OWL Syntax	0.231	0.231	1.0	0.375
	E. Natural Language	0.986	0.986	0.952	0.969
Mistral (7b)	M. OWL Syntax	0.587	0.357	0.986	0.524
	Natural Language	0.783	0.517	0.952	0.67
	E. M. OWL Syntax	0.89	0.69	0.952	0.8
	E. Natural Language	0.958	0.875	0.952	0.912
Mixtral (47b)	M. OWL Syntax	0.231	0.231	1.0	0.375
	Natural Language	0.925	0.773	0.952	0.854
	E. M. OWL Syntax	0.989	1.0	0.952	0.976
	E. Natural Language	0.989	1.0	0.952	0.976

Table 7: Metrics for University ontology