Language Models for Ontology Engineering

Yuan He

St Hugh's College

Department of Computer Science

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2024

Supervisors

Prof. Ian Horrocks
Prof. Bernardo Cuenca Grau
Dr. Jiaoyan Chen

Acknowledgements

Embarking on my PhD amidst the COVID-19 pandemic presented unforeseen challenges. The initial isolation, along with the abrupt transition to remote learning, tested my resilience. However, this period also marked the beginning of my fruitful PhD journey.

I owe immense gratitude to Prof. Ian Horrocks, whose mentorship was pivotal. His insights often extend beyond the immediate scope of a single study or project, capturing the bigger picture. Ian's guidance on structuring research around core questions, contributions, and evaluations profoundly influenced my research thinking.

My heartfelt thanks also to Dr. Jiaoyan Chen. His mentorship was instrumental in the research details, from crafting compelling narratives to discussing nuanced techniques, fostering my growth as an independent researcher.

I extend my gratefulness to my colleagues in the KRR group, whose expertise and insights have been valuable. Special appreciation goes to my flatmates in Student Castle and many other friends, who provided cultural exchange and warmth especially during challenging times. My badminton sporting circle offered a necessary respite from academic pursuits, balancing my PhD journey with physical activity.

Dating back to my academic beginnings, I am thankful to Dr. Shay Cohen, my undergraduate supervisor, for providing early research opportunities that paved the way for my PhD pursuits.

Lastly, I am deeply thankful to my parents, whose unwavering support has been my anchor, and to Mengqi, my beloved girlfriend, for her witness and companionship throughout these years.

As I reflect on the initial struggle and adaptation of my PhD journey, I realise how quickly it has ushered me into the next chapter of my life. I am thankful to all the lessons, challenges, and growth it has brought.

Funding Acknowledgements My research was supported by Samsung Research UK (SRUK), and EPSRC projects OASIS (EP/S032347/1), UK FIRES (EP/S019111/1), and ConCur (EP/V050869/1).

Abstract

Ontology, originally a philosophical term, refers to the study of being and existence. The concept was introduced to Artificial Intelligence (AI) as a knowledge-based system that can model and share knowledge about entities and their relationships in a machine-readable format. Ontologies offer a structured and logical formalism of human knowledge, enabling expressive representations and reliable reasoning within defined domains. Meanwhile, modern deep learning-based language models (LMs) represent a significant milestone in the field of Natural Language Processing (NLP), as they incorporate substantial background knowledge from the vast and complex distribution of textual data. This thesis explores the synergy between these two paradigms, focusing primarily on the use of LMs in ontology engineering and, more broadly, in knowledge engineering. The goal is to automate or semi-automate the process of ontology construction and curation.

Ontology engineering includes a wide array of tasks within the life cycle of ontology development. This thesis concentrates on three key aspects: *(i)* ontology alignment, which seeks to align equivalent concepts across different ontologies to achieve data integration; *(ii)* ontology completion, which focuses on filling in missing subsumption relationships between ontology concepts; and *(iii)* hierarchy embedding, which aims to develop versatile and interpretable neural representations for hierarchical structures derived not only from ontologies but also applicable to other forms of hierarchical data. These representations can facilitate a broad spectrum of downstream ontology engineering tasks, such as *(i)* and *(ii)*, and are adaptable for more general applications in hierarchy-aware contexts.

This thesis is organised into three parts. The first part establishes the foundations necessary for understanding ontologies and LMs. The chapter on ontologies initiates with a basic overview of computational ontologies, then provides an introduction of the description logic formalisms that underpin them. It concludes with the formal definitions of the three ontology engineering tasks this thesis focuses on. Transitioning to LMs, the subsequent chapter begins with a chronological overview of their evolution, followed

by detailed exposition of various typical LMs along this evolution. The discussion then proceeds to contemporary transformer-based LMs, elaborating on their architecture and different learning paradigms they adopt. The chapter concludes with a review of how LMs and knowledge bases (including ontologies) interact and influence each other, highlighting the mutual benefits of this integration for both fields of study.

With the comprehensive background provided in the first part, the second part of the thesis delves into specific methodologies that have been developed. This part comprises three chapters, each corresponding to the application of LMs in ontology alignment, ontology completion, and hierarchy embedding, respectively. In the chapter on LMs for ontology alignment, we introduce BERTMap, a novel pipeline system that employs LM fine-tuning for improved alignment prediction and ontology semantics for alignment refinement. We will also mention the Bio-ML track of the Ontology Alignment Evaluation Initiative (OAEI), which has emerged as a benchmarking platform for a variety of ontology alignment systems over the past two years. The chapter on LMs for ontology completion presents OntoLAMA, a collection of LM probing datasets and a prompt-based LM probing approach that effectively predicts subsumptions, even with limited training resources. Lastly, the section on LMs for hierarchy embedding discusses the re-training of LMs as Hierarchy Transformer encoders (HiT), addressing the limitations of LMs in explicitly interpreting and encoding hierarchies, including those extracted from ontologies.

The third part of the thesis details the practical implementations. We mainly present `DeepOnto`, a Python package designed for ontology engineering utilising deep learning, with an emphasis on LMs. `DeepOnto` offers a range of basic to advanced ontology processing functionalities to support deep learning-based ontology engineering development. This package also includes polished implementations of our systems and resources mentioned in Part II.

In summary, this thesis advocates for a more holistic approach in AI development, where the integration of LMs and ontologies can lead to a more advanced, explainable, and useful paradigm in knowledge engineering and beyond.

List of Publications

Main Publications

[1] **Yuan He**, Jiaoyan Chen, Denvar Antonyrajah, and Ian Horrocks. "BERTMap: A BERT-based Ontology Alignment System." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 5, pp. 5684-5691. 2022. (AAAI'22).

[2] **Yuan He**, Jiaoyan Chen, Hang Dong, Ernesto Jiménez-Ruiz, Ali Hadian, and Ian Horrocks. "Machine Learning-friendly Biomedical Datasets for Equivalence and Subsumption Ontology Matching." In *International Semantic Web Conference*, pp. 575-591. Cham: Springer International Publishing, 2022. (ISWC'22; **Best Resource Paper Candidate**).

[3] **Yuan He**, Jiaoyan Chen, Ernesto Jimenez-Ruiz, Hang Dong, and Ian Horrocks. "Language Model Analysis for Ontology Subsumption Inference." In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 3439–3453. Association for Computational Linguistics. 2023. (ACL'23).

[4] **Yuan He**, Jiaoyan Chen, Hang Dong, Ian Horrocks, Carlo Allocca, Taehun Kim, and Brahmananda Sapkota. "DeepOnto: A Python Package for Ontology Engineering with Deep Learning." *Semantic Web*. 2024. (SWJ'24).

[5] **Yuan He**, Jiaoyan Chen, Hang Dong, and Ian Horrocks. "Exploring Large Language Models for Ontology Alignment." In *International Semantic Web Conference (Posters and Demos)* 2023. (ISWC'23).

[6] **Yuan He**, Zhangdie Yuan, Jiaoyan Chen, and Ian Horrocks. "Language Models as Hierarchy Encoders." Under review. 2024.

Other Relevant Publications

[7] Jiaoyan Chen, **Yuan He**, Yuxia Geng, Ernesto Jiménez-Ruiz, Hang Dong, and Ian Horrocks. "Contextual Semantic Embeddings for Ontology Subsumption Prediction." *World Wide Web*, pp.1-23. 2023. (WWWJ'23).

[8] OAEI organisers. "Results of the Ontology Alignment Evaluation Initiative 2022-2023." In *International Workshop on Ontology Matching (OM@ISWC)*. 2022-2023. (ISWC'22-23)

[9] Ryan Brate, Minh-Hoang Dang, Fabian Hoppe, **Yuan He**, Albert Meroño-Peñuela, and Vijay Sadashivaiah. "Improving Language Model Predictions via Prompts Enriched with Knowledge Graphs." In *Workshop on Deep Learning for Knowledge Graphs (DL4KG@ISWC)*. 2022. (ISWC'22).

[10] Hang Dong, Jiaoyan Chen, **Yuan He**, Yinan Liu, Ian Horrocks. "Reveal the Unknown: Out-of-Knowledge-Base Mention Discovery with Entity Linking." In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2023. (CIKM'23).

[11] Hang Dong, Jiaoyan Chen, **Yuan He**, and Ian Horrocks. "Ontology Enrichment from Texts: A Biomedical Dataset for Concept Discovery and Placement." In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2023. (CIKM'23; **Best Resource Paper Runner-Up**).

[12] Hang Dong, Jiaoyan Chen, **Yuan He**, and Ian Horrocks. "A Language Model based Framework for New Concept Placement in Ontologies." In *Extended Semantic Web Conference*. 2024. (ESWC'24).

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Contents

1.1 Motivation

Ontology, originally a term from philosophy, refers to the study of being and existence. In Artificial Intelligence (AI), this concept is adapted to denote a knowledge representation system that models entities and their relationships formally and explicitly in a format that is both machine-readable and shareable. Ontology engineering involves the various stages in the development of ontologies, including their design, implementation, construction, evaluation, and curation (Gómez-Pérez et al., 2006).

In the earlier era of the Web, ontology engineering emphasised the establishment of standards such as the Resource Description Framework (RDF) (Lassila and Swick, 1999), Resource Description Framework Schema (RDFS) (Dan and Guha, 2004) and the Web Ontology Language (OWL) (Bechhofer et al., 2004; W3C OWL Working Group, 2009). These standards are pivotal parts in the Semantic Web technology stack, facilitating the communication and modelling of data on the Web. Additionally, there

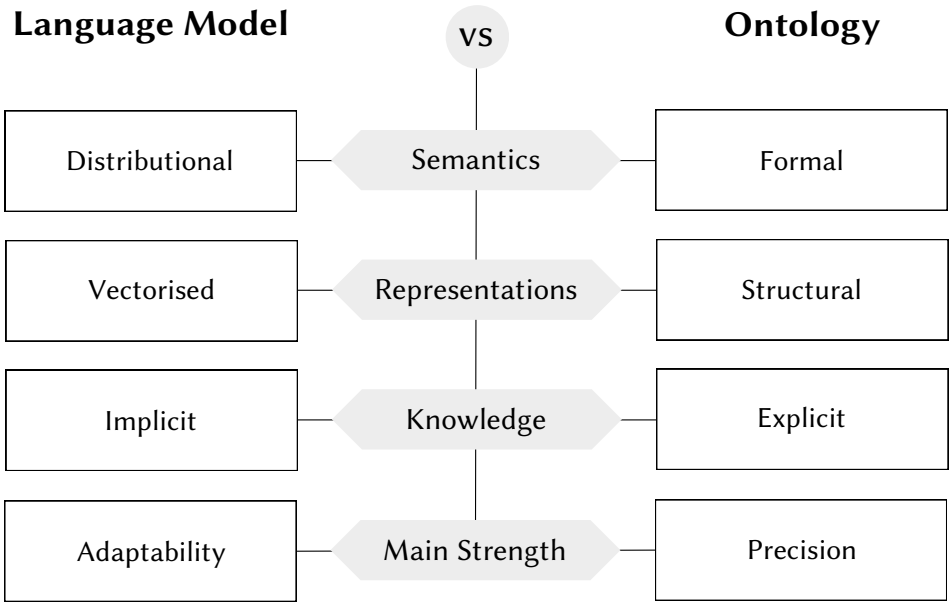


Figure 1.1: A brief comparison of (modern) language models and ontologies.

was an increased focus on collaborative methodologies for the creation and continuous refinement of ontologies, leveraging community-driven paradigms to enrich and validate these frameworks (Simperl and Luczak-Rösch, 2014).

As the web continues to evolve, the exponential increase in data generation necessitates more efficient methods for transforming vast amounts of information into structured knowledge. This has led to a growing interest in automation and semi-automation techniques in ontology engineering. In particular, the paradigm of Neural-Symbolic (NeSy) AI has been extensively investigated, building on the belief that the predictive capabilities of neural models combined with the automated reasoning of symbolic models can lead to a stronger AI system (Garcez et al., 2022).

In the rapidly progressing field of neural and deep learning, transformer-based language models (LMs) such as BERT (Kenton and Toutanova, 2019) and GPT (Radford et al., 2018) have distinguished themselves. They have demonstrated the ability to digest enormous volumes of textual data, effectively encoding this information into re-usable background knowledge. Given the vast scale of training data these models have digested, some have argued that they act as compressors of knowledge (Delétang et al., 2023).

Figure 1.1 provides a side-by-side comparison of modern[1] LMs and ontologies, analysing their characteristics across four key dimensions: semantics, representations, knowledge types, and main strengths. The following paragraphs will detail each dimension, and conclude with the integration of these two paradigms.

Semantics LMs are based on *distributional semantics* (Harris, 1954), learning word meanings from statistical distributions in extensive text corpora, which enables the identification of word similarities and other semantic associations based on patterns. In contrast, ontologies are grounded in *formal semantics* (Van Fraassen and Fraassen, 1971), aiming to express languages with precise, logical structures.

Representations LMs implement distributional semantics using high-dimensional vector spaces, where words are embedded into *vectors* and their proximity in the geometric space indicates relatedness. Ontologies, conversely, employ *structural representations* founded on logical constructs to express entities and their relationships, establishing a semantic network of clearly defined vocabularies.

Knowledge Types LMs, trained on vast text corpora, incorporate *implicit* and often general knowledge learned from a plethora of textual data sources. The knowledge type is *implicit* because the inferences drawn from LMs are probabilistic in nature. Ontologies, however, are repositories of *defined* knowledge, with *explicit* statements of entities and their relationships that yield deterministic and consistent deductions through formal reasoning within their defined scope.

Strength of Each LMs are adept at language processing and understanding [2], exhibiting remarkable *adaptability* to various kinds of downstream (even domain-specific) needs. Although ontologies cannot process information beyond their defined scopes, they support rigorous logical reasoning about entities and their relationships within these scopes. This capability is crucial for applications that require high *precision*[3] and traceability of information sources.

[1] Modern LMs are predominantly statistical. The study of earlier rule-based LMs is beyond the scope of this thesis.

[2] Here "understanding" refers to the ability to process and generate human-like text. However, true understanding in the cognitive sense is limited.

[3] The precision is based on the assumption that knowledge inputted into ontologies by human experts is reliable.

Integration of Both The integration of LMs and ontologies harnesses the fine-grained, context-sensitive language understanding of LMs and the definitive, logical structuring of ontologies. This fusion brings in a semantic framework that capitalises on both the predictive power of statistical models and the preciseness of deductive logic.

This thesis explores the integration of LMs with ontologies to facilitate automated ontology engineering. The following section will detail the specific research questions addressed by this thesis and the corresponding contributions.

1.2 Contribution

This section summarises the contributions of this thesis in relation to specific research questions.

Research Question 1a. *How to effectively utilise LMs to map equivalent concepts across different ontologies for the task of ontology alignment?*

Contribution 1a. We introduce BERTMap (He et al., 2022a), a pipeline system that integrates LM fine-tuning with the lexical, structural, and logical information inherent in input ontologies, to significantly enhance the ontology alignment process. BERTMap overcomes the limitation of traditional rule-based systems in handling linguistic variations such as different word forms and synonyms. By adopting an unsupervised learning strategy that relies the intrinsic information of ontologies, BERTMap sidesteps the dependency on external, annotated alignment data, which is often required in existing machine learning-based alignment approaches. Furthermore, BERTMap design is fairly extensible, allowing seamless incorporation of additional resources, including annotated mappings and auxiliary ontologies. This flexibility makes BERTMap a versatile tool adaptable to various ontology alignment scenarios.

Research Question 1b. *Given the traditional focus of ontology alignment evaluation on rule-based systems and equivalence matching, how can machine learning principles be integrated to establish a holistic evaluation framework that accommodates both concept equivalence and subsumption matching tasks? Furthermore, how can this framework be extended to include the evaluation of recent Large Language Models (LLMs)?*

Contribution 1b. In our work on Bio-ML (He et al., 2022b), we introduce a novel evaluation framework that improves the assessment of ontology alignment systems. This framework offers a comprehensive evaluation for both rule-based and machine learning-based alignment systems by taking into account both *global matching* and *local ranking*. Global matching evaluates the system's overall alignment performance, while local ranking facilitates a more focused and comparative analysis, crucial for the efficient development and debugging of alignment systems, especially the machine learning-based ones.

A notable aspect of our framework is the incorporation of dataset splitting along with unsupervised and semi-supervised settings, offering more detailed understanding of system performance across different scenarios. Importantly, we employ only local ranking for the evaluation of subsumption matching because the ground truth subsumption alignment set is inherently incomplete.

Extending this framework, our subsequent work in He et al. (2023a) integrates the evaluation of LLMs. By focusing on small yet challenging datasets, we offer a practical and efficient methodology for understanding the capabilities and limitations of LLMs in ontology alignment.

Research Question 2a. *To what extent have LMs encoded knowledge of concept subsumption in ontologies?*

Research Question 2b. *Can LMs alone suffice in predicting absent subsumptions between concepts in ontologies, thereby contributing to ontology completion?*

Contribution 2. Research Question 2a actually relates to the realm of LM probing, with a particular focus on examining if LMs can function as Knowledge Bases (KBs) (Petroni et al., 2019a). We are at the forefront of introducing ontologies to this research area through the development of OntoLAMA (He et al., 2023c), a set of LM probing datasets along with a prompt-based probing methodology for ontology subsumption inference. These datasets include ontologies from a various scales and domains, incorporating both atomic and complex concepts. Our findings indicate that, under the zero-shot setting, LMs demonstrate limited background knowledge about concept subsumptions.

Addressing Research Question 2b, which concentrates on the application of LMs in ontology engineering and, more specifically, ontology completion, our study demonstrates that LMs can significantly enhance their prediction of concept subsumptions with the aid of a minimal set of learning examples (few-shot setting). The results highlight the potential of LMs to substantially contribute to ontology completion given appropriate prompt engineering.

Furthermore, our adaption to LM probing has led to a valuable ontology verbalisation tool. This tool adeptly translates complex concept expressions into natural language texts through a recursive process. Such an approach to ontology processing enhances the accessibility of ontologies to LMs and other text-based machine learning systems.

Research Question 3. *How can hierarchical structures, derived not only from ontologies but also from broader sources, be explicitly encoded in LMs to produce a universal neural embedding for hierarchies?*

Contribution 3. We introduce a novel methodology to re-train LMs as Hierarchy Transformer Encoders (HITs) (He et al., 2024), leveraging the expansive nature of hyperbolic geometry. Our approach incorporates two components: hyperbolic clustering and hyperbolic centripetal losses, which are designed to effectively cluster entities based on their relatedness and arrange them hierarchically.

Our contribution is two-fold: Firstly, the HIT models generate universal hierarchy embeddings that are applicable to a range of downstream tasks, extending well into ontology engineering and beyond. Secondly, one of the fundamental challenges with current LMs is their inherent limitation in explicitly interpreting and representing hierarchical relationships. Our methodology directly tackles this challenge, equipping output embeddings LMs with geometric interpretability.

In addition to the methodological advancements described in Contributions 1-3, this thesis also makes a practical contribution by developing and implementing resources and tools for the field.

Contribution 4. We introduce `DeepOnto` (He et al., 2023b), a Python package for ontology engineering with deep learning, with a particular emphasis on LM-related techniques. DeepOnto serves as a pivotal bridge, connecting the robust capabilities of the Java-based OWL API (Horridge and Bechhofer, 2011) with the dynamic and rapidly evolving Python ecosystem, where leading deep learning frameworks such as TensorFlow (Abadi et al., 2016) and PyTorch (Paszke et al., 2019) are predominantly developed. This integration is crucial for advancing ontology engineering, as it combines the strengths of semantic web technologies with the latest advancements in deep learning.

`DeepOnto` encapsulates an extensive array of ontology processing functionalities within its core module, aiming to transform ontologies into various data formats to facilitate the development of deep learning-based ontology engineering systems. It also implements the methodologies discussed in Contributions 1-3, offering them as ready-to-use tools. Furthermore, `DeepOnto` supports the resource construction and evaluation introduced in Bio-ML (Contribution 1b).

1.3 Thesis Outline

The remainder of this thesis is structured into three different parts, outlined as follows:

Part I Foundations paves the way for understanding the subsequent discussions. Chapter 2 provides background information of ontologies and definitions of specific ontology engineering tasks this thesis seeks to address. Chapter 3 introduces preliminaries of language models and existing literature on their applications in knowledge engineering.

Part II Methodologies explores the specific methodologies for integrating LMs and ontologies across several ontology engineering challenges. Chapters 4, 5, and 6 discuss the topics of Language Models for Ontology Alignment, Language Models for Ontology Completion, and Language Models for Hierarchy Embedding, respectively. These chapters directly link to the methodological Contributions 1-3 listed in Section 1.2.

Part III Implementations complements the thesis with practical contributions in implementing tools and resources. Chapter 7 introduces the Python package `DeepOnto` (Contribution 4), ranging from its fundamental ontology processing functionalities to resources and tools it has implemented.

Conclusion This thesis concludes with Chapter 8, summarising the key points for all the chapters.

1.A Publication Correspondences

In Table 1.1, we present the correspondences of chapters in Part II and Part III to the main publications listed on Page vii.

Chapter #	Publications
Chapter 4	• Yuan He et al., "BERTMap: A BERT-based Ontology Alignment System." **AAAI'22**. • Yuan He et al., "Machine learning-friendly biomedical datasets for equivalence and subsumption ontology matching." **ISWC'22, Best Resource Paper Candidate**. • Yuan He et al., "Exploring Large Language Models for Ontology Alignment." **ISWC'23: Posters & Demos**.
Chapter 5	• Yuan He et al., "Language Model Analysis for Ontology Subsumption Inference." **ACL'23: Findings**.
Chapter 6	• Yuan He et al., "Language Models as Hierarchy Encoders." **Under review**.
Chapter 7	• Yuan He et al., "DeepOnto: A Python Package for Ontology Engineering with Deep Learning." **SWJ'24**.

Table 1.1: Correspondences of thesis chapters to relevant publications.

PART I

Language Models for Ontology Engineering

Foundations

CHAPTER 2

Foundations of Ontologies

Contents

This chapter provides essential background information on ontologies and ontology engineering. It commences with an overview of computational ontologies in Section 2.1, where we discuss both their abstract definitions and practical applications. At the end of this overview, we will outline the structure of subsequent sections.

2.1 Overview

Computational ontologies are designed to model and represent entities and their relationships. Their account of the original philosophical meaning, i.e., the study of being

and existence, is the notion that "For AI systems, what 'exists' is that which can be represented." (Gruber, 1993, 1995).

Studer et al. (1998) defined computational ontology as "a formal, explicit specification of a shared conceptualisation." While this definition is widely acknowledged, it remains somewhat abstract. Guarino et al. (2009) delved deeper into the essential components of this definition, elucidating that *conceptualisation* refers to the way of identifying entities and defining vocabularies to describe them. A *formal, explicit specification* implies the use of a language or standard that ensures the intended conceptualisation is clearly understood and implemented. The term *shared* highlights the importance of ontologies being reusable and interoperable across different computer programs. This insight reveals that the early phase of ontology engineering focused on identifying suitable design patterns and developing a language or formalism that could effectively encapsulate and communicate complex conceptualisations.

In the transition from the late 20th to the early 21st century, the World Wide Web Consortium (W3C) has developed a stack of semantic web technologies for data exchange on the Web and ontological modelling. A cornerstone of these technologies, the Resource Description Framework (RDF), was established as a W3C Recommendation in 1999 (Lassila and Swick, 1999). RDF describes data through triples structured as $(\mathsf{subject}, \mathsf{predicate}, \mathsf{object})$, thus constructing a graph with subjects and objects as nodes, and predicates as edges. Building upon the foundational layer provided by RDF, the RDF Schema (RDFS) (Dan and Guha, 2004) extends its capabilities by introducing a rich vocabulary designed to describe semantics about data. The integrated framework of RDF and RDFS facilitates a structured and interoperable approach to implement ontologies.

However, such graph-based data modelling is not underpinned by a rigorously defined logical formalism, which limits its expressiveness for complex ontological modelling. To address this limitation, the Web Ontology Language (OWL) was developed and subsequently adopted as a comprehensive standard for ontology creation, with its specifications detailed in W3C Recommendations (Bechhofer et al., 2004; W3C OWL Working Group, 2009). As illustrated in Figure 2.1, OWL ontology encompasses two types of semantic frameworks. The Direct Semantics, rooted in Description Logics (DLs),

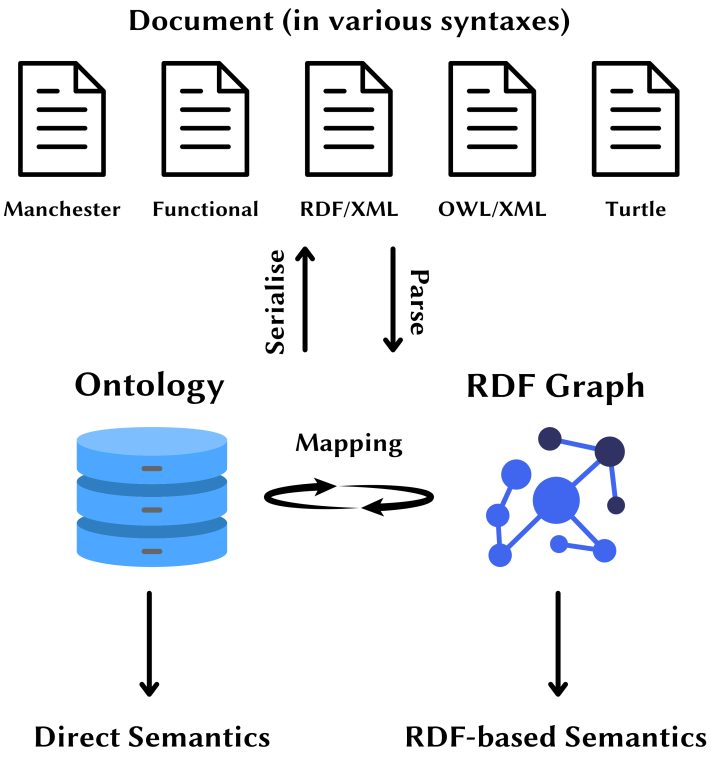


Figure 2.1: Overview of OWL Ontology.

brings a formal logical underpinning to OWL, enabling highly expressive vocabulary and computationally decidable reasoning. On the other hand, the RDF-based Semantics builds directly upon RDF and RDFS, enriching RDF graphs with logical constructs and ensuring compatibility between OWL ontologies and existing RDF data structures.

Additionally, OWL is designed to be flexible in its representation, supporting serialisation in various syntaxes to cater to different needs and preferences. These include the Manchester Syntax, which is known for its readability and ease of use in ontology editing; the Functional-Style Syntax, favoured for its closeness to the underlying formal semantics of OWL; RDF/XML, a widely-used markup language for representing complex data on the Web; OWL/XML, an XML-based representation specific to OWL ontologies; and Turtle Syntax, which offers a more compact and readable alternative to RDF/XML.

For a more in-depth understanding of OWL, it is essential to first grasp the fundamentals of DLs (see next section). Following this, Section 2.3 will delve into OWL in greater detail. Finally, we will discuss ontology engineering in Section 2.4 , with a particular focus on the specific tasks addressed in this thesis.

2.2 Description Logics

2.2.1 Introduction

Description Logics (DLs) are a family of knowledge representation languages that underpin today's ontological modelling. The basic components of a DL knowledge base (KB) are *concepts*, *roles*, and *individuals*.

Concepts represent sets of individuals, embodying the categories or groups to which these individuals belong. For instance, the concept $\mathsf{OxfordStudent}$ might be defined to represent all students at the University of Oxford. If John is an individual studying at Oxford, the KB could include an assertional statement (or a *fact*) like $\mathsf{OxfordStudent}(\mathsf{John})$, indicating John's membership in the $\mathsf{OxfordStudent}$ concept. Additionally, a terminological statement (or a *rule*) such as $\mathsf{OxfordStudent} \sqsubseteq \mathsf{Person}$ demonstrates that all Oxford students are also persons. Complex concepts involving logical operators are also possible. For example, $\mathsf{OxfordStudent} \sqsubseteq \exists \mathsf{studyAt}.\mathsf{University}$ states that an Oxford student is an individual who studies at some university, where $\mathsf{studyAt}$ denotes a binary relation (or *role*) between individuals. Based on these examples, a simple DL KB could be formulated as follows:

Example 2.2.1 (A simple KB)

A simple KB that describe Oxford students is given by:

$$\mathcal{T} = \{\mathsf{OxfordStudent} \sqsubseteq \mathsf{Person},\ \mathsf{OxfordStudent} \sqsubseteq \exists \mathsf{studyAt}.\mathsf{University}\}$$
$$\mathcal{A} = \{\mathsf{OxfordStudent}(\mathsf{John})\}$$

where $\mathcal{T}$ denotes the *TBox* that contains **terminological** axioms and $\mathcal{A}$ denotes the *ABox* that contains **assertional** axioms.

We can then conduct reasoning over this KB. A a direct consequence is $\mathsf{Person}(\mathsf{John})$, derived from applying the rule $\mathsf{OxfordStudent} \sqsubseteq \mathsf{Person}$ to the fact $\mathsf{OxfordStudent}(\mathsf{John})$.

Nevertheless, this KB does not reflect the intended meaning of $\mathsf{OxfordStudent}$. To address this and achieve a precise definition, we can introduce an equivalence axiom:

$\mathsf{OxfordStudent} \equiv \exists \mathsf{studyAt}.\{\mathsf{OxfordUni}\}$. Here, $\{\mathsf{OxfordUni}\}$ represents a *nominal* – a concept that denotes a set of specific individuals. This addition ensures that the concept OxfordStudent is explicitly defined as individuals who study at the specific institution, i.e., the University of Oxford. Furthermore, we can infer that $\mathsf{University}(\mathsf{OxfordUni})$ based on the constructed KB and this newly introduced axiom.

2.2.2 Basic Description Logic $\mathcal{ALC}$

Moving forward, we will formally define key terms in DLs. We start by introducing a basic DL known as $\mathcal{ALC}$, which stands for **A**ttributive concept **L**anguage with **C**omplements. The following table presents the syntax and semantics of $\mathcal{ALC}$ concept constructors.

Constructor	Syntax	Semantics ($\cdot^{\mathcal{I}}$)
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
atomic concept	A	$A^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}(\langle x, y\rangle \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$
universal restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}(\langle x, y\rangle \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\}$

Table 2.1: Syntax and semantics of $\mathcal{ALC}$ concept constructors.

Here, C and D are arbitrary concept expressions in $\mathcal{ALC}$, which can be constructed using the presented syntax recursively. A is a named atomic concept and r is a named role. The semantics is given with regard to *interpretations*, defined as follows:

Definition 2.2.1 (Interpretation)

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of an non-empty set $\Delta^{\mathcal{I}}$ (the domain of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ that maps each concept C to $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role r to $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. If $x \in C^{\mathcal{I}}$, then x is an instance of C in $\mathcal{I}$. The right column of Table 2.1 presents the semantics of each concept constructor in $\mathcal{ALC}$ w.r.t. $\mathcal{I}$.

As illustrated in Example 2.2.1, a DL KB comprises TBox and ABox axioms. In the following, we define each box and their semantics using interpretations.

Definition 2.2.2 (TBox)

A TBox is a finite set of General Concept Inclusion (GCI) axioms of the form $C \sqsubseteq D$. We say an interpretation $\mathcal{I}$ is a model of (or entails) $C \sqsubseteq D$, written as $\mathcal{I} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it is a model of every GCI in $\mathcal{T}$. We use $C \equiv D$ to abbreviate the pair of GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$. A special form of $C \equiv D$ where C is a named atomic concept A is referred to as a definition.

Definition 2.2.3 (ABox)

An ABox is a finite set of concept assertion axioms and role assertion axioms of the forms $C(x)$ and $r(x, y)$, respectively, where x and y are named individuals. We say an interpretation $\mathcal{I} \models C(x)$ if $x^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\mathcal{I} \models r(x, y)$ if $\langle x^{\mathcal{I}}, y^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$. $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it is a model of every concept or role assertion axiom in $\mathcal{A}$.

With the definitions of TBox and ABox, we can conceptualise a KB as a composite structure consisting of both a TBox and an ABox.

Definition 2.2.4 (Knowledge Base)

A knowledge base $\mathcal{K}$ consists of a TBox, $\mathcal{T}$, and an ABox, $\mathcal{A}$, collectively denoted by $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. An interpretation $\mathcal{I} \models \mathcal{K}$ if $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$.

It is important to recognise that the Unique Name Assumption (UNA) and the Closed-World Assumption (CWA) are not made in DLs. The absence of UNA implies that two differently named concepts can, through inference, be demonstrated as equivalent. However, it is possible to make an explicit statement through the individual equality (or inequality) assertions of the form $x \approx y$ (or $x \not\approx y$). The lack of CWA, or in other words, the adoption of the Open-World Assumption (OWA), means that not knowing a

fact does not equate to confirming its opposite. For instance, if a KB $\mathcal{K}$ does not entail $C(x)$, denoted as $\mathcal{K} \not\models C(x)$, it does not logically follow that $\mathcal{K} \models \neg C(x)$.

2.2.3 Reasoning Problems

Having introduced the formal definitions of fundamental DL terms, we now present several basic reasoning problems associated with a KB $\mathcal{K}$.

- **Consistency**: $\mathcal{K}$ is consistent if there exists a model $\mathcal{I}$ of $\mathcal{K}$.
- **Concept Satisfiability**: A concept C is satisfiable w.r.t. $\mathcal{K}$ if there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$.
- **Concept Subsumption**: A concept C is subsumed by a concept D w.r.t. $\mathcal{K}$, written as $\mathcal{K} \models C \sqsubseteq D$, if $\mathcal{I} \models C \sqsubseteq D$ for every model $\mathcal{I}$ of $\mathcal{K}$.
- **Concept Equivalence**: Concepts C and D are equivalent w.r.t. $\mathcal{K}$, written as $\mathcal{K} \models C \equiv D$, if $\mathcal{K} \models C \sqsubseteq D$ and $\mathcal{K} \models D \sqsubseteq C$.
- **Concept Disjointness**: Concepts C and D are disjoint w.r.t. $\mathcal{K}$ if $\mathcal{K} \models C \sqcap D \sqsubseteq \bot$, or equivalently, $\mathcal{K} \models C \sqsubseteq \neg D$. That is, there is no model $\mathcal{I}$ of $\mathcal{K}$ such that both C and D are satisfiable.
- **Concept Membership** An individual x is an instance of a concept C w.r.t. $\mathcal{K}$ if $\mathcal{I} \models C(x)$ for every model $\mathcal{I}$ of $\mathcal{K}$.
- **Role Membership** An pair of individuals (x, y) is an instance of a named role r w.r.t. $\mathcal{K}$ if $\mathcal{I} \models r(x, y)$ for every model $\mathcal{I}$ of $\mathcal{K}$.

The listed reasoning problems can often be reduced to a KB consistency problem. For example, to check if a concept C is satisfiable in $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we can transform it into the problem that whether $\mathcal{K}' = (\mathcal{T}, \mathcal{A} \cup \{C(x_0)\})$ is consistent.

2.2.4 Extensions to $\mathcal{ALC}$

Recall the full name of $\mathcal{ALC}$ on page 15, we can see that the letter $\mathcal{C}$ actually introduces **negation** as a new feature to extend the more restrictive $\mathcal{AL}$. In the following, we introduce more DL extensions:

- $\mathcal{H}$: **Role hierarchies**, also known as RBox, allowing for role inclusion axioms of the form $r \sqsubseteq p$ where r and p are named roles.

  Example 2.2.2

  isMotherOf $\sqsubseteq$ isParentOf expresses that if an individual is the mother of another, then the individual is a parent (broader role) to that person.

- $\mathcal{R}$: **Complex role hierarchies**, introducing complex role inclusion axioms of the forms $r \circ p \sqsubseteq r$ and $p \circ r \sqsubseteq r$, where $r \circ p$ denotes the role composition whose semantics is given by:

$$(r \circ p)^{\mathcal{I}} := \{\langle x, y\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists z \in \Delta^{\mathcal{I}}(\langle x, z\rangle \in r^{\mathcal{I}} \wedge \langle z, y\rangle \in p^{\mathcal{I}})\}$$

  Example 2.2.3

  isMotherOf$\circ$studyAt expresses the relationship of being the mother of someone who studies at some institution.

  Several role characteristics are also allowed, including:

  - *Role disjointness*, e.g., isMotherOf $\sqcap$ isFatherOf $\sqsubseteq \bot$ expresses that no individual can simultaneously be the mother and father of another individual.
  - *Role reflexivity*, e.g., discipline is reflexive as everyone can discipline him/herself.
  - *Role irreflexivity*, e.g., isMotherOf is irreflexive because an individual cannot be his/her own mother,
  - *Role asymmetry*, which is discussed below along with the inverse roles.

- *Negated role assertions*, e.g., $\neg\mathsf{isMotherOf}(\mathsf{Mary}, \mathsf{John})$.
- *Universal role* U with the interpretation $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The *empty role* is not directly defined in DL but can be expressed as $\top \sqsubseteq \neg\exists r.\top$.
- *Local reflexivity* of the form $\exists r.\mathsf{Self}$. E.g., someone who funds him/herself can be expressed as $\exists\mathsf{fund}.\mathsf{Self}$.

- $\mathcal{I}$: **Inverse roles** of the form r^- are allowed here and the semantics is given by:

$$(r^-)^{\mathcal{I}} := \{\langle x, y\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle y, x\rangle \in r^{\mathcal{I}}\}$$

With role hierarchies introduced in $\mathcal{H}$, *role symmetry* can be defined via $r \equiv r^-$. With the role disjointness introduced in $\mathcal{R}$, *role asymmetry* can be determined when r and r^- are disjoint.

Example 2.2.4

$\mathsf{isParentOf}^- \equiv \mathsf{isChildOf}$ expresses the mutual relationship between a parent and his/her child. To further state that both relationships are asymmetrical, we can add in the disjointness $\mathsf{isParentOf} \sqcap \mathsf{isChildOf} \sqsubseteq \bot$.

- $\mathcal{S}$: **Transitive roles** plus $\mathcal{ALC}$ is often abbreviated as $\mathcal{S}$. Transtivity of a role r, denoted as $\mathsf{Trans}(\mathsf{r})$, can be expressed using the role composition $r \circ r \sqsubseteq r$. An interpretation $\mathcal{I} \models \mathsf{Trans}(\mathsf{r})$ if

$$\forall x, y, z \in \Delta^{\mathcal{I}}(\langle x, y\rangle \in r^{\mathcal{I}} \wedge \langle y, z\rangle \in r^{\mathcal{I}} \rightarrow \langle x, z\rangle \in r^{\mathcal{I}})$$

Example 2.2.5

$\mathsf{isSiblingOf} \circ \mathsf{isSiblingOf} \sqsubseteq \mathsf{isSiblingOf}$ expresses the transitivity of the sibling relationship.

- $\mathcal{N}$: **Cardinality restrictions** are complex concepts of the forms $\geq n\ r$ and $\leq n\ r$, where $n \in \mathbb{N}$, used to constrain the size of a role's interpretation. The

corresponding semantics are given by:

$$(\geq n\ r)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid |\{y \in \Delta^{\mathcal{I}} \mid \langle x, y\rangle \in r^{\mathcal{I}}\}| \geq n\}$$
$$(\leq n\ r)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid |\{y \in \Delta^{\mathcal{I}} \mid \langle x, y\rangle \in r^{\mathcal{I}}\}| \leq n\}$$

Example 2.2.6

≤ 2 hasParent expresses the restriction on a group of individuals having at most two parents.

- $\mathcal{Q}$: **Qualified cardinality restrictions** are complex concepts of the forms $\geq n\ r.C$ and $\leq n\ r.C$, where $n \in \mathbb{N}$, used to constrain the size of a role's interpretation specific to a concept C. The corresponding semantics are given by:

$$(\geq n\ r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid |\{y \in C^{\mathcal{I}} \mid \langle x, y\rangle \in r^{\mathcal{I}}\}| \geq n\}$$
$$(\leq n\ r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid |\{y \in C^{\mathcal{I}} \mid \langle x, y\rangle \in r^{\mathcal{I}}\}| \leq n\}$$

Example 2.2.7

≥ 2 hasChild.Doctor expresses the group of individuals having at least two children that are doctors.

- $\mathcal{O}$: **Nominals** allow a concept to be defined by a set of individuals $x_1, x_2, ..., x_n$. The semantics of such a concept is given by:

$$(\{x_1, x_2, ..., x_n\})^{\mathcal{I}} := \{x_1^{\mathcal{I}}, x_2^{\mathcal{I}}, ..., x_n^{\mathcal{I}}\}$$

Example 2.2.8

RusselGroup $\equiv \{$OxfordUni, CambridgeUni, EdinburghUni, ...$\}$ defines the concept of Russel Group using the set of universities in it.

We can combine the listed extensions to form a particular DL. For example, $\mathcal{ALCH}$ is an extension of $\mathcal{ALC}$ with role hierarchies.

Summary This section introduces the foundational elements of DLs that serve as the backbone for OWL ontologies. Specifically, we present formal definitions of DL terminology, focusing on the basic DL, $\mathcal{ALC}$, and discuss common reasoning problems. We the explore important extensions to $\mathcal{ALC}$ that will be used to formalise OWL in the subsequent section. For a deeper understanding of DLs, such as reasoning algorithms and their theoretical connections to other formal logics, readers are encouraged to consult the comprehensive textbook by Baader et al. (2008).

2.3 OWL Ontology

The Web Ontology Language (OWL) is a Semantic Web technology designed to model complex knowledge structures and facilitate robust reasoning about entities and their relationships. Developed by the World Wide Web Consortium (W3C), OWL has evolved through multiple iterations, with OWL 2 (W3C OWL Working Group, 2009) being the latest version. OWL 2 leverages the Description Logic (DL) $\mathcal{SROIQ}$ as its logical underpinning. Hereafter, references to OWL imply OWL 2 unless specified otherwise.

OWL shares core components with DLs, namely *classes*, *properties*, and *individuals*, where classes and properties refer to concepts and roles in DLs. In OWL, *individuals* represent specific objects in the real world, such as a person named John or a cat named Cessy. *Classes* group these individuals under common characteristics, serving as an abstraction layer; for instance, the class Human might encompass all individuals sharing certain genetic markers. *Properties* define the relationships between individuals, for example, $\mathsf{hasPet}(\mathsf{John}, \mathsf{Cessy})$ expresses that John has a pet named Cessy. Moreover, properties can be combined with logical operators to form complex classes, such as $\exists \mathsf{hasPet}.\mathsf{Cat}$, identifying individuals who own a cat as a pet.

As introduced in Section 2.1, OWL incorporates two types of semantic frameworks: **Direct Semantics** (Horrocks et al., 2012) and **RDF-based Semantics** (Schneider et al., 2009). Direct Semantics adheres closely to the formalism of DL $\mathcal{SROIQ}$ (see $\mathcal{SROIQ}$ constructors in Tables 2.2 and 2.3), while RDF-based Semantics extends the capabilities of RDF and RDFS, enabling a broader integration with the existing Semantic Web infrastructure. Within these frameworks, OWL is differentiated into two profiles: OWL

Constructor	Syntax	Semantics ($\cdot^{\mathcal{I}}$)
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
atomic concept	A	$A^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}(\langle x, y\rangle \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$
universal restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}(\langle x, y\rangle \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\}$
at-least restriction	$\geq n\, r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \in C^{\mathcal{I}} \mid \langle x, y\rangle \in r^{\mathcal{I}}\}\rvert \geq n\}$
at-most restriction	$\leq n\, r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \in C^{\mathcal{I}} \mid \langle x, y\rangle \in r^{\mathcal{I}}\}\rvert \leq n\}$
local reflexivity	$\exists r.\mathsf{Self}$	$\{x \in \Delta^{\mathcal{I}} \mid \langle x, x\rangle \in r^{\mathcal{I}}\}$
nominal	$\{x_1, ..., x_n\}$	$\{x^{\mathcal{I}}, ..., x_n^{\mathcal{I}}\}$

Table 2.2: Syntax and semantics of $\mathcal{SROIQ}$ concept constructors.

Constructor	Syntax	Semantics ($\cdot^{\mathcal{I}}$)
universal role	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
atomic role	r	$r^{\mathcal{I}}$
role composition	$r \circ p$	$\{\langle x, y\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists z \in \Delta^{\mathcal{I}}(\langle x, z\rangle \in r^{\mathcal{I}} \wedge \langle z, y\rangle \in p^{\mathcal{I}})\}$
inverse role	r^{-}	$\{\langle x, y\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \langle y, x\rangle \in r^{\mathcal{I}}\}$

Table 2.3: Syntax and semantics of $\mathcal{SROIQ}$ role constructors.

DL[1], associated with Direct Semantics, and OWL Full, aligned with RDF-based Semantics. OWL DL is designed to offer a balance between expressiveness and computational efficiency, ensuring that reasoning tasks remain decidable. In contrast, OWL Full offers the highest level of expressiveness by allowing complete utilisation of RDF and RDFS features alongside OWL constructs. However, this increased expressiveness comes at a cost, as OWL Full does not always guarantee the decidable reasoning available in OWL DL.

The **Functional-style Syntax** is often used for formulating OWL expressions. This syntax facilitates a clear and systematic representation of ontology constructs, align-

[1]OWL DL has several sub-profiles, including OWL EL, OWL QL, and OWL RL, each tailored to optimise performance under particular requirements by constraining the expressiveness of underlying DLs (Motik et al., 2009).

ing closely with the underlying DL formalisms.[2] For instance, an OWL expression ObjectIntersectionOf(C D) directly maps to the DL expression $C \sqcap D$. While many OWL expressions serve as syntactic parallels to $\mathcal{SROIQ}$, OWL further enriches these with syntactic sugars or shortcuts to enhance usability in practical scenarios. For example, we can declare role symmetry directly in OWL using SymmetricObjectProperty(r).

In fact, OWL DL supports several logical features beyond the DL $\mathcal{SROIQ}$. A significant enhancement is the integration of **datatypes** $\mathcal{D}$ such as `int`, `string`, `boolean`, and `float`, which are handled through *concrete domains* equipped with pre-defined semantics in DLs (Baader and Hanschke, 1991). At the syntax level, OWL makes a clear distinction between properties that connect individuals (ObjectProperty) and those that link individuals to datatype values (DatatypeProperty). Additionally, OWL introduces support for **keys**, a logical feature that enables the unique identification of individuals based on a combination of their property values and class memberships. For example, a student might be uniquely identified through their membership in the class Student and specific attributes like the name of their school and student ID.

Beyond these logical enhancements, OWL also incorporates practical features commonly found in other formal languages, such as: *(i)* **importing** axioms from one ontology into another to facilitate modular ontology development and reuse and *(ii)* **annotating** entities and axioms using AnnotationProperty, with `rdfs:label` (a term from RDFS) being frequently employed to provide human-readable names.

Summary In this section, we discuss OWL in more detail, building upon the DL preliminaries in Section 2.2. OWL introduces a comprehensive ontology framework that advances beyond the graph-based data modelling capabilities of RDF/RDFS by integrating DL formalisms. Importantly, it preserves compatibility with RDF structures, ensuring a seamless transition between these semantic layers. Beyond the foundational DL constructs, OWL incorporates a range of features for practical applications, accommodating diverse needs in ontology development and deployment.

[2] Other syntaxes, as illustrated in Figure 2.1, can also be mapped to DLs, but not as direct as the Functional-style syntax.

2.4 Ontology Engineering

2.4.1 Scope

Ontology engineering, a sub-field of knowledge engineering, deals with the entire development life-cycle of ontologies. In the early era of the Web, this field primarily focused on establishing standards, formalisms, and design patterns for constructing ontologies, alongside methods for acquiring knowledge from various sources. However, the exponential increase in web data necessitates automated or semi-automated methods for ontology engineering to ensure efficiency and scalability. This thesis focuses on particular tasks that are critically in need of automation, such as **ontology alignment** and **ontology completion**, and the term "ontology" mainly denotes an OWL ontology.

Ontology alignment aims to integrate ontologies from heterogeneous sources, necessitating automation due to the impracticality of manually identifying overlaps between ontologies, especially as they expand to include vast numbers of classes and properties. Similarly, ontology completion involves identifying and filling gaps in an ontology, such as undetected subsumption relationships between concepts, a task beyond the scope of manual curation due to its complexity.

Furthermore, we are also interested in **hierarchy embedding**, which involves representing the entity hierarchy of an ontology through neural or sub-symbolic methods. Ontologies, by their nature, encode real-world entities in a structured and explicit format. Neural embedding techniques transform these structured representations into geometric spaces, where both entities and their relationships are (often) modelled as vectors. This transformation aims to encapsulate the underlying semantics of the ontologies, thereby providing a versatile representation that can benefit a range of ontology engineering tasks. In the following sub-sections, we will formally define each mentioned task.

2.4.2 Ontology Alignment

Ontology alignment, or ontology matching (OM), addresses the issue of semantic heterogeneity by identifying correspondences between semantically related entities across different ontologies (Shvaiko and Euzenat, 2011). These correspondences are

instrumental in achieving goals such as ontology merging and quality assurance. OM thus plays a crucial role in enabling interoperability of knowledge and data across diverse sources. Formally, OM is defined as:

Definition 2.4.1 (Ontology Alignment)

Given two input ontologies, $\mathcal{O}$ and $\mathcal{O}'$ and their respective signatures (sets of entities), $\mathsf{Sig}(\mathcal{O})$ and $\mathsf{Sig}(\mathcal{O})'$, OM aims to determine a set of *mappings* (or correspondences) represented as triples (e, r, e'), where $e \in \mathsf{Sig}(\mathcal{O})$ and $e' \in \mathsf{Sig}(\mathcal{O})'$ denote the matched entities, and r specifies the semantic relationship between them.

Entities in ontologies are classes (concepts), properties (roles), and individuals. The primary semantic relationships of interest typically involve equivalence ($\equiv$), highlighting direct correspondence between entities. There is also increasing focus on subsumption ($\sqsubseteq$), indicating hierarchical relationships where one entity is a subclass of another. Beyond these, research also explores more complex and nuanced relationships that exist among ontology entities (Thiéblin et al., 2020).

For the purpose of evaluation, mappings that have been expertly curated or are assumed to be correct are commonly referred to as ground truths or reference standards. These serve as benchmarks for assessing the accuracy and reliability of mappings generated by OM systems. In traditional OM research, the following evaluation metrics from information retrieval (Manning et al., 2008) are frequently adopted.

Definition 2.4.2 (Evaluation Metrics)

Let $\mathcal{M}_{\mathsf{ref}}$ denote the set of ground truth or reference mappings, and $\mathcal{M}_{\mathsf{pred}}$ denote the set of mappings predicted by the system. The evaluation metrics Precision (P), Recall (R), and F-score (F_1) are defined as follows:

$$P = \frac{|\mathcal{M}_{\mathsf{pred}} \cap \mathcal{M}_{\mathsf{ref}}|}{|\mathcal{M}_{\mathsf{pred}}|}, \quad R = \frac{|\mathcal{M}_{\mathsf{pred}} \cap \mathcal{M}_{\mathsf{ref}}|}{|\mathcal{M}_{\mathsf{ref}}|}, \quad F_1 = \frac{2PR}{P+R}$$

In the context of machine learning, entity mappings are often augmented with a confidence score, $s(e, r, e')$, representing the confidence of the mapping (e, r, e'). This score is typically normalised to fall within the interval $[0, 1]$, where 0 indicates complete uncertainty and 1 signifies absolute confidence in the mapping. Under this setting, ground truth mappings are assigned a confidence score of 1. For system-predicted mappings, a scoring threshold is often required and treated as a *hyperparameter*[3]. Mappings with a score greater than or equal to this threshold are finalised as output mappings.

Having discussed the basic definitions and settings, we now turn our attention to the key challenges in this domain:

- **Ambiguity in naming schemes**: Ontologies may utilise different terminology to denote the same entities or employ the same term to describe different entities. For instance, "muscle layer" and "muscularis" propria" both describe a specific muscular region across various organs; conversely, the term "bank" might refer to either a financial institution or the side of a river, depending on the context.

- **Diversity of design focus**: Ontologies may vary in the level of detail or granularity with which they treat aspects of a domain. For example, an ontology focusing on dog breeds might categorise breeds based on suitability as pets or for hunting, whereas an ontology on diseases might merely note that dogs can be carriers of the rabies virus.

- **Scalability**: The vast size and complexity of certain ontologies, particularly in fields like biomedicine, introduce substantial computational challenges for alignment processes. Notably, a naive entity-by-entity comparison approach in OM requires quadratic time, meaning that matching two ontologies of 100k entities takes 10 billion comparisons.

Addressing the challenges outlined above, the development of OM systems typically involves two critical aspects:

[3]A hyperparameter is a non-learnable parameter that should be optimised through the validation process, distinct from the model parameters that are learned during training. We will discuss more about such machine learning principles in Section 3.2.7 of Chapter 3.

- **Searching**: By pre-selecting candidates that are likely to match, the system can concentrate its efforts on the plausible entity pairs, thereby effectively reducing the alignment search space and improving scalabiity.

- **Matching**: This involves specific assessment to the semantic relationship between candidate entities. An effective *lexical matching* algorithm is crucial for reconciling discrepancies in naming conventions, leveraging textual annotations such as entity names and descriptions to compute textual similarities, or more complex textual relations particularly in OM scenarios not limited to equivalence. Additionally, *context matching* plays a critical role, as it analyses the context of entities within their ontologies to disambiguate entities with similar names.

It is important to note that searching and matching in OM system design are often interconnected rather than isolated. Furthermore, in the case when semantic relationships between entities are rigorously defined, OM may also require *mapping repair*, i.e., to remove mappings that will cause inconsistency upon merging two ontologies.

Methodology Preview In Chapter 4, we will delve into our OM system, BERTMap, which is designed to effectively address the aforementioned challenges and considerations. To fully understand BERTMap, it is essential to first familiarise ourselves with the fundamentals of language models, which will be covered in the subsequent chapter. Additionally, we will discuss the challenges of OM evaluation and present the Bio-ML resource, which aims to benchmark OM with a comprehensive evaluation framework.

Related Task in KG Engineering OM shares similarities with Entity Alignment (EA) in KGs, particularly because RDF serves as a common standard for both ontologies and KGs. The use of textual and structural information presents a significant overlap in methods applied to both OM and EA, and the most prevalent setting for both domains is *equivalence matching of named entities*. However, KGs often do not adhere to strict logical formalisms, which complicates the detection of inconsistencies arising from resulting alignment. Furthermore, OWL offers greater expressiveness compared to RDF, enabling the exploitation of more complex semantics within ontologies than is possible with common KGs.

2.4.3 Ontology Completion

Ontology completion refers to the task of adding missing information to an existing ontology, serving as a crucial step in ontology curation. It can be broadly divided into two sub-tasks: *(i)* inferring missing rules (TBox or RBox axioms) and *(ii)* inferring missing facts (ABox axioms). We define ontology completion as follows:

Definition 2.4.3 (Ontology Completion)

Given an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, where $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{A}$ denote the TBox, RBox, and ABox, respectively, ontology completion aims to expand $\mathcal{O}$ to $\mathcal{O}'$, such that $\mathcal{O}' = (\mathcal{T} \cup \mathcal{T}', \mathcal{R} \cup \mathcal{R}', \mathcal{A} \cup \mathcal{A}')$. Here, $\mathcal{T}'$, $\mathcal{R}'$, and $\mathcal{A}'$ denote the sets of newly inferred axioms.

It is crucial to understand that "missing" information extends beyond what can be inferred using deductive reasoning with an ontology reasoner. Consider the following example for clarity:

Example 2.4.1

Given a simple ontology with the axioms:

$$\{\mathsf{Influenza} \sqsubseteq \exists \mathsf{hasSymptom.Fever},\ \mathsf{H1N1} \sqsubseteq \mathsf{Influenza},\ \mathsf{Antipyretic} \sqsubseteq \exists \mathsf{treats.Influenza}\}$$

From these rules, we can infer that H1N1, being a subclass of Influenza, also has the symptom of Fever. However, it remains unspecified whether Antipyretic is a treatment for H1N1, despite it being a treatment for Influenza. Ontology completion seeks to identify and fill such knowledge gaps by integrating external information.

The example illustrates the need to identify missing subsumption relationship such as the one between the concepts Influenza and $\exists$treats.Influenza. This thesis focuses on the task of inferring such missing subsumption relationships to enrich an ontology's concept hierarchy, defined as follows:

Definition 2.4.4 (Subsumption Inference)

Given an ontology $\mathcal{O}$ with a TBox $\mathcal{T}$ and a set of (atomic or complex) concepts $\mathcal{C}$, the objective of (concept) subsumption inference is to expand $\mathcal{T}$, to $\mathcal{T} \cup \mathcal{T}'$, where $\mathcal{T}'$ contains newly inferred subsumption axioms $c \sqsubseteq d$ for concepts $c, d \in \mathcal{C}$.

Regarding evaluation, ontology completion can be considered a specialised form of OM, wherein an ontology undergoes self-alignment to identify missing subsumption relationships. This perspective allows us to employ an evaluation framework akin to the one detailed in the previous section. Using Precision, Recall, and F-score as the metrics, we can assess the efficacy of ontology completion by comparing the predicted subsumption axioms $\mathcal{T}'_{\mathsf{pred}}$ (excluding those already included in $\mathcal{T}$) against the reference set $\mathcal{T}'_{\mathsf{ref}}$.

More often, the literature resorts to a partial evaluation approach due to the challenges in acquiring a *complete* set of missing subsumption relationships – a common issue also encountered in OM, which will be further discussed in Chapter 4. In this approach, we establish a possibly incomplete reference set $\mathcal{T}'_{\mathsf{ref}}$ and, for each reference subsumption axiom, we generate a set of negative subsumption examples. The task for ontology completion systems then becomes to identify the correct subsumptions from among these negative candidates. Such identification is associated with the rank defined as follows:

Definition 2.4.5 (Rank)

For each positive subsumption example (c, c^+) where $c \sqsubseteq c^+ \in \mathcal{T}'_{\mathsf{ref}}$, we sample a set of N negative super-classes for c, resulting in the set of negative candidate pairs $\{(c, c_i^-)\}_{i=1}^N$. The rank of c^+ is the ordinal position where a system places the pair (c, c^+) among all $N + 1$ candidate pairs $\{(c, c^+)\} \cup \{(c, c_i^-)\}_{i=1}^N$.

Based on the definition of rank, we can then define the following common ranking-based metrics (Liu and Özsu, 2009):

Definition 2.4.6 (Ranking-based Evaluation Metrics)

The ranking-based metrics Hits@K and MRR (Mean Reciprocal Rank) are defined as:

$$\text{Hits@K} = \frac{1}{|\mathcal{T}'_{\text{ref}}|} \sum_{(c,c^+)\in\mathcal{T}'_{\text{ref}}} \mathbb{I}_{\text{rank}(c^+)\leq K}, \quad \text{MRR} = \frac{1}{|\mathcal{T}'_{\text{ref}}|} \sum_{(c,c^+)\in\mathcal{T}'_{\text{ref}}} \text{rank}(c^+)^{-1}$$

To determine negative subsumptions, the Closed-World Assumption (CWA) is frequently employed, under which a concept c^- is considered a negative super-class of concept c if $\mathcal{O} \not\models c \sqsubseteq c^-$. This approach contrasts with the Open-World Assumption (OWA) typically adopted in Description Logics (DLs), which is discussed on page 16.

Note that ontology completion is a typical task that requires the predictive capabilities of models – capabilities that exceed those of symbolic or rule-based systems. This is the domain where machine learning-based systems shine. Although our definition of subsumption inference is limited to concepts within an ontology, it is common to expect that the relevant machine learning techniques could extend their benefits to ontology construction or enrichment, where concepts are coming from external sources.

Methodology Preview In Chapter 5, we will explore our approach to ontology completion, focusing on subsumption inference. This approach also includes a dataset construction method that finds a middle ground between the CWA and OWA . From the perspective of language models (LMs), this work is also related to the literature of Language Models as Knowledge Bases (LMs-as-KBs), which aims to probe to what extent LMs can function as KBs. We will introduce the relevant literature on LMs-as-KBs in the next chapter.

Related Task in KG Engineering Ontology completion and KG completion both aim to augment a knowledge base by infilling missing information. In KG completion, the focus is typically on predicting the missing element in a triple (s, r, o), enabling predictions ($?$) for $(s, r, ?)$, $(s, ?, o)$, or $(?, r, o)$. The setting of $(s, ?, o)$ is often referred

to as *link prediction*. A fundamental distinction between ontology completion and KG completion lies in their representation: KGs uniformly treat logical relationships (like subsumption and membership) and properties (like studyAt and isParentOf) as predicates in triples (or facts), where the subject s and object o are generally atomic entities. In contrast, ontologies differentiate between rules and facts, allowing logical relationships to connect complex entities that involve logical operators.

Example 2.4.2

A statement like "London is the capital of the UK," can be straightforwardly expressed in a KG as the triple:

$$(\mathsf{London}, \mathsf{capitalOf}, \mathsf{UK})$$

However, a more nuanced statement like "arthritis is a kind of arthropathy with an inflammatory morphology" is challenging to represent with simple triples. In ontology, this can be articulated through a GCI with complex concepts, such as:

$$\mathsf{Arthritis} \sqsubseteq \mathsf{Arthropathy} \sqcap \exists \mathsf{hasMorphology}.\mathsf{Inflammatory}$$

Moreover, in KGs, both $\sqsubseteq$ and hasMorphology will be treated as a predicate.

While the RDF-based semantics allow for the translation of an OWL ontology into a KG without semantic loss, the resultant KG may introduce numerous blank nodes (e.g., to denote existential restrictions), which are rarely seen in typical KGs and do not fit in common KG completion settings.

2.4.4 Hierarchy Embedding

In mathematics, especially in topology and geometry, an *embedding* refers to a function that maps one mathematical structure into another while preserving essential properties (Nash, 1956). For example, embedding a line into a plane involves mapping every point on the line to a unique point on the plane, ensuring that the line's structure is maintained

without any crossings or overlaps. This process guarantees that the embedded object retains its original properties, even when situated within a higher-dimensional space.

In machine learning, the concept of embedding retains the core idea of "structure preservation" but in a more flexible way. Here, an embedding is a **representation of data in a vector space**. This concept allows for complex data forms, such as text or images, to be represented as vectors, facilitating their processing by machine learning algorithms. We will elucidate word embeddings in Section 3.2.3 of Chapter 3. In the current section, our focus is on the application of embeddings in representing entity hierarchies.

We now provide the definitions of *hierarchy* and *hierarchy embedding*.

Definition 2.4.7 (Hierarchy)

A hierarchy is defined as a directed acyclic graph (DAG)[a] $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of vertices that represent the entities, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of directed edges (or arcs) that represent the direct hierarchical relationships between the entities in the hierarchy.

[a] "Directed" implies that the edges have a direction, indicating the relationship from one entity to another, and "acyclic" means there are no closed loops within the graph.

Definition 2.4.8 (Hierarchy Embedding)

A hierarchy embedding is a function $f : \mathcal{V} \to \mathbb{S}^d$ that maps entities in a hierarchy $\mathcal{G}(\mathcal{V}, \mathcal{E})$ to points in a d-dimensional space $\mathbb{S}^d$. This mapping aims to represent the hierarchical relationships in $\mathcal{E}$ such that the directed edges (or arcs) connecting entities in $\mathcal{V}$ are reflected in the spatial relationships between points in $\mathbb{S}^d$.[a]

[a] While $\mathbb{S}$ is commonly the Euclidean space $\mathbb{R}$, it can also represent other types of spaces, such as a complex space $\mathbb{C}$ or a hyperbolic space $\mathbb{H}$.

We anticipate that a hierarchy embedding function should retain the intrinsic properties of the original hierarchy, with *transitivity* being a prime example. However, the embedding itself does not inherently express the relationships between points. Thus, we introduce an additional function g that interprets pairs of embedded entities

from $\mathbb{S}^d \times \mathbb{S}^d$ to ascertain their hierarchical relationships. Ideally, g should act as a "companion" to f, implying that its definition should be derived from f rather than being independently learned from training data. Notably, in specific models like the hyperbolic entailment cone (Ganea et al., 2018b) and the box embedding (Abboud et al., 2020), researchers strive for an *analytical solution* where, in an ideal scenario of zero learning loss, the relationship determination can be mathematically derived. More commonly, g is treated as an empirical scoring function, with its parameters, such as scoring thresholds, optimised during the machine learning validation process.

Difference from Ontology Embedding In this discussion, we deliberately choose "hierarchy embedding" over "ontology embedding" to highlight a specific scope. Hierarchy embedding focuses exclusively on encoding the hierarchical relationships among entities. In contrast, ontology embedding extends to embedding logical axioms, capturing their semantics within a geometric space. This thesis is specifically concerned with embedding the concept hierarchies of ontologies. However, it is crucial to understand that the application of hierarchy embedding extends beyond the realm of ontologies. It is applicable to any hierarchical structure, whether originating from ontologies, taxonomies, knowledge graphs, or syntax trees. On the other hand, ontology embedding is specifically designed to address the logical formalisms underpinning ontologies, dealing with a wider spectrum of logical relationships and properties that go beyond simple hierarchical connections.

Methodology Preview In Chapter 6, we delve into our approach to hierarchy embedding, which capitalises on the strengths of language models (LMs) and hyperbolic geometry, demonstrating significant promise for hierarchy-oriented semantic search. From the perspective of NLP, we address a critical limitation of LMs: While they are adept at capturing word similarities, they typically fall short in explicitly interpreting hierarchical structures. Our methodology aims to bridge this gap, enhancing the LMs' ability to understand and represent hierarchical relationships within data.

Related Task in KG Engineering The literature on embedding has also been extensively studied in the context of KG engineering, where the primary objective often focuses on embedding entities and the diverse relationships between them altogether. Within this framework, hierarchy embedding can be perceived as a special case of KG embedding tailored for only hierarchical relationships.

CHAPTER 3

Foundations of Language Models

Contents

3.1 Overview

Language models (LMs) are statistical tools that determine the probability distribution of various linguistic units in a language, such as characters, words, sentences, or sequences

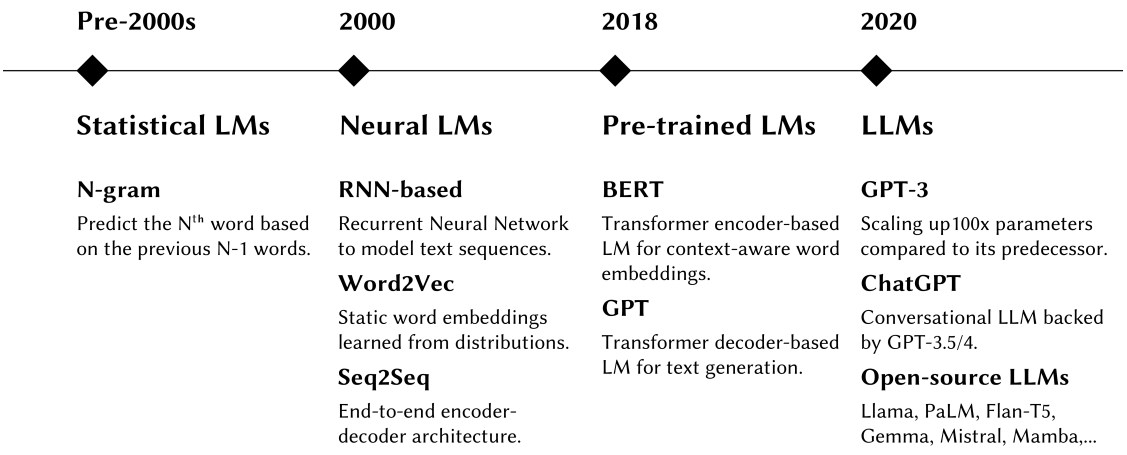


Figure 3.1: Evolution of language models.

of words. For example, consider the three English phrases: "a dog barks", "a dog flies", and "fly dog runs". We would expect LMs to assign probabilities to these phrases in descending order due to their decreasing naturalness or commonality in usage. Essentially, an LM is defined as a system that models the probability $P(s)$ of a text sequence s or the conditional probability $P(s|c)$, where c is some conditioning *context* of s.

LMs play a crucial role in numerous Natural Language Processing (NLP) tasks, including language understanding (Kenton and Toutanova, 2019), machine translation (Wu et al., 2016), sentiment analysis (Araci, 2019), and dialogue systems (Vinyals and Le, 2015), to name a few. The emergence of recent Large Language Models (LLMs) like ChatGPT (OpenAI, 2023) signifies a notable advancement, positioning these models as general task solvers that incorporate vast amounts of background knowledge. Integrated with *multi-modality*, these models leverage text to bridge and interpret diverse data types, such as images, videos, and audio, thus broadening their applicability in real-world scenarios.

Figure 3.1 demonstrates the key milestones in the evolution of LMs. The subsequent paragraphs briefly illustrate each phase mentioned in the figure.

Statistical Language Models The roots of statistical language modelling extend back to the 1950s when Shannon (1951) explored the concept of the N-gram model for the English language. The core idea of an N-gram LM is to predict the most probable Nth

word in a sequence given the preceding $N-1$ words as context. At a broader conceptual level, this research established a fundamental paradigm in LM design, defining LMs as systems that generate new text tokens conditioned on existing ones. This definition has underpinned the majority of the future development in LM research.

Neural Language Models The advent of neural networks (NNs) has led to more complex architectures designed to recognise patterns across various data types. Notable examples are Convolutional Neural Networks (CNNs) (LeCun et al., 1989) for image processing and Recurrent Neural Networks (RNNs) (Rumelhart et al., 1985) for sequential text data.

RNN-based LMs superseded the original N-gram model by incorporating the historical context (preceding text) within a hidden state, which then informs the prediction of subsequent words. Among the RNN variants, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks stand out by employing a *gating* mechanism to maintain long-range dependencies. A famous variant of LSTM is the Gated Recurrent Unit (GRU) (Cho et al., 2014), which simplifies the LSTM architecture while retaining its effectiveness across various tasks. These models, especially when augmented with bidirectional processing – that is, considering context from both forward and backward directions – or with an attention mechanism – which calculates the relevance of words within the context – significantly enhance their predictive capabilities. It is crucial to clarify that RNN $\neq$ LM, but rather a popular component for implementing LMs.

The sequence-to-sequence (Seq2Seq) architecture (Sutskever et al., 2014), introduced in 2014[1], demonstrated the effectiveness of an encoder-decoder framework in language modelling. In this set-up, the context is fed as input into the encoder, and then the decoder utilises the encoded context to predict subsequent words. Typically, the encoder employs a bidirectional RNN with an attention mechanism for fully capturing the contextual nuances, and the decoder employs a unidirectional RNN for sequential token generation.

In parallel, there was a shift towards representing words through dense vectors rather than sparse one-hot encodings. This transition was epitomised by Word2Vec (Mikolov et al., 2013), a novel technique that pioneered the use of NNs to learn word embeddings.

[1]Some have argued that an earlier idea of Seq2Seq showed up in the work of Mikolov et al. (2012).

These embeddings are dense vector representations that capture the semantics of words based on their distribution in large text corpora. Following Word2Vec, subsequent methodologies such as GloVe (Pennington et al., 2014) and FastText (Bojanowski et al., 2017) further advanced the field. GloVe optimised the process by focusing on word co-occurrence statistics, while FastText introduced sub-word information, allowing for a better handling of linguistic variations and rare words.

Pre-trained Language Models In 2017, Vaswani et al. (2017) posited a pivotal question: given the effectiveness of the attention mechanism, particularly in capturing long-range dependencies within text, why not develop an architecture that relies solely on it? Moving away from previous sequential models like RNN-based LMs, Vaswani et al. (2017) proposed the transformer architecture. Its core innovation is the *self-attention* mechanism that allows each token within a sequence to *attend* to every other token simultaneously. This methodology not only enhances the model's capability to capture complex dependencies but also significantly optimises hardware utilisation. As a result, the transformer has set a new standard for subsequent developments in language modelling, and extending its influence beyond NLP (Han et al., 2022).

In 2018-19, the first two typical LMs based on the transformer architecture were introduced: BERT (Kenton and Toutanova, 2019), utilising the transformer's encoder for contextualised word embeddings,[2] and GPT (Radford et al., 2018), utilising the transformer's decoder for fluent text generation. These models have marked a new paradigm in language modelling, characterised by *pre-training* and *fine-tuning*. Pre-training involves training an LM on extensive text corpora to acquire background knowledge, while fine-tuning tailors this pre-trained model to specific downstream tasks using targeted learning objectives. The fine-tuning phase typically requires significantly fewer computational resources due to the background knowledge instilled during pre-training.

[2]It is noteworthy that during this time, ELMo (Peters et al., 2018) also introduced contextual language modelling, albeit not based on the transformer architecture.

Large Language Models The advent of transformer-based LMs marked a significant milestone in NLP, exhibiting impressive performance across a broad spectrum of tasks. To explore the full potential of this architecture, researchers experimented with scaling up the model sizes, reaching billions and even trillions of parameters, thus giving rise to the recent advancements of Large Language Models (LLMs).

In 2020, OpenAI introduced GPT-3, a massive model featuring 175 billion parameters (Brown et al., 2020), setting a precedent for the development of LLMs. This trend continued with the launch of ChatGPT[3] in 2022, a sophisticated conversational agent powered by GPT-3.5 onwards. These models have pushed the boundaries of transformer-based LMs further in terms of scale and capabilities, demonstrating that with more parameters and training data, LMs can achieve remarkable levels of comprehension and fluency.

LLMs have become fundamental in AI, supporting a diverse range of applications from interactive chatbots to sophisticated analytical tools across various scientific domains. Their ability to generate coherent and contextually relevant text has not only advanced the field of NLP but also opened new frontiers in human-computer interaction, automated content creation, AI for sciences, and beyond.

3.2 Formal Preliminaries

3.2.1 Fundamentals of Probability Theory

We begin with a concise overview of key concepts in probability theory that form the backbone of modern machine learning techniques.

Definition 3.2.1 (Conditional Probability)

The conditional probability of an event X occurring given that another event Y has already occurred is defined as $P(X|Y) = \frac{P(X \cap Y)}{P(Y)}$. By rearranging this equation, we obtain $P(X \cap Y) = P(X|Y)P(Y)$, known as the *product rule*. This rule indicates that the joint probability of X and Y is equivalent to the probability of Y occurring multiplied by the probability of X occurring given Y.

[3] OpenAI's ChatGPT interface: `https://chat.openai.com/`; at the time of writing, the most recent version is GPT-4.

When Y is a subset or outcome of X, e.g., $X =$ "John is driving" and $Y =$ "John is drunk driving", then $P(X|Y) = 1$. Conversely, if X is a subset or outcome of Y, as in our previous example but with X and Y interchanged, then $P(X|Y) = \frac{P(X)}{P(Y)}$. The joint probability $P(X \cap Y)$ is also commonly denoted as $P(X, Y)$.

Utilising basic probability laws, we can establish the following useful rules:

Definition 3.2.2 (Chain Rule)

The chain rule extends the product rule to the joint probability of a sequence of events, expressed as:

$$P(X_1 \cap ... \cap X_n) = \prod_{i=1}^{n} P(X_i | \bigcap_{j=1}^{i-1} X_j)$$

Definition 3.2.3 (Sum Rule)

The sum rule provides a method to decompose the probability of an event using a countable set of conditional events:

$$P(X) = \sum_{i=1}^{n} P(X|Y_i)$$

where Y_is are *mutually exclusive* ($P(Y_i \cap Y_j) = 0$ for $i \neq j$) and *collectively exhaustive* ($P(\bigcup_i Y_i) = 1$).

The well-known Bayes' Theorem can be directly derived from the product rule.

Definition 3.2.4 (Bayes' Theorem)

By applying the product rule, we deduce that $P(X \cap Y) = P(X|Y)P(Y) = P(Y|X)P(X)$, leading to the Bayes' Theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Finally, we introduce the concept of *likelihood*, a term frequently misunderstood as synonymous with *probability*.

Definition 3.2.5 (Likelihood Function)

Consider n independent and identically distributed (i.i.d.) random variables $X_1, \ldots, X_n$ with the same probability mass (or density of X is continuous) function $p(x; \theta)$, for a sequence of observations $x_1, \ldots, x_n$, the likelihood function is defined as:

$$L(\theta) = \prod_{i=1}^{n} p(x_i; \theta)$$

This definition illustrates that likelihood is a function of parameters w.r.t. observed data. In the case of discrete distributions, it is synonymous with the joint probability of observing the specific set of data points. In the case of continuous distributions, it refers to the joint probability density evaluated at the observed data.[4]

To illustrate, consider the scenario of coin tossing in the following example.

Example 3.2.1

For a coin that may be biased, we assign the probability of flipping a "Head" to be a parameter θ. The outcomes of the coin tosses can be modelled using a Bernoulli distribution with parameter θ. The probability mass function of this distribution is then:

$$p(x; \theta) = \begin{cases} \theta & \text{if } x = \text{“Head”} \\ 1 - \theta & \text{if } x = \text{“Tail”} \end{cases}$$

Suppose we observe a "Head" for t times in a sequence of n independent coin tosses, the likelihood function is expressed as:

$$L(\theta) = \prod_{i=1}^{n} p(x_i; \theta) = \binom{n}{t} \theta^t (1 - \theta)^{n-t}$$

[4]It is worth noting that $p(x_i; \theta) = P(X_i = x_i; \theta)$ in the discrete case and $p(x_i; \theta)dx = P(x_i \leq X_i \leq x_i + dx; \theta)$ (where dx is infinitely small) in the continuous case.

Definition 3.2.6 (Maximum Likelihood Estimation)

Maximum Likelihood Estimation (MLE) is the process of selecting θ that maximises the likelihood function given observed data, i.e., $\hat{\theta} = \arg\max_{\theta \in \Theta} L(\theta)$, where Θ refers to the parameter space.

For Example 3.2.1, the MLE $\hat{\theta} = \frac{t}{n}$.

MLE is widely adopted in machine learning. A neural network can be viewed as a series of interconnected differentiable functions. Optimisation of the entire network is achieved via *backpropagation* (Rumelhart et al., 1986), a process that iteratively adjusts the network's parameters to **maximise the likelihood of the observed data**.

3.2.2 N-gram

Consider modelling the distribution of a text sequence s, consisting of n words, denoted as $s = (w_1, w_2, \ldots, w_n) = w_{1:n}$. Utilising the chain rule (see Definition 3.2.2), the likelihood function can be decomposed as follows:

$$L(\theta) = P(s; \theta) = P(w_{1:n}; \theta) = P(w_1; \theta) P(w_2 | w_1; \theta) \ldots P(w_n | w_{<n}; \theta)$$

where $w_{<n}$ is an abbreviation of $w_{1:n-1}$. However, accounting for the entire history of preceding words becomes computationally infeasible for large n. The N-gram model addresses this by considering only the preceding $N-1$ words as history.

Definition 3.2.7 (N-gram Estimation)

In N-gram, the conditional probability for the ith word w_i in s, given its preceding context, is approximated as:

$$P(w_i | w_{<i}; \theta) = \begin{cases} P(w_i | w_{i-N+1:i-1}; \theta) & \text{if } i \geq N \\ P(w_i | w_{<i}; \theta) & \text{if } i < N \end{cases}$$

In this context, the parameter θ includes the probabilities of all possible text sequences s from a *training* corpus. Assuming a Multinomial distribution for these sequences, the

MLE for each sequence s is its observed frequency $C(s)$ normalised by the total count of sequences C_{total} in the corpus. This leads to the following expression:

$$\hat{P}(w_i|w_{i-N+1:i-1}) = \frac{\hat{P}(w_{i-N+1:i})}{\hat{P}(w_{i-N+1:i-1})} = \frac{C(w_{i-N+1:i})/C_{total}}{C(w_{i-N+1:i-1})/C_{total}} = \frac{C(w_{i-N+1:i})}{C(w_{i-N+1:i-1})}$$

This formula indicates that the estimated conditional probability $P(w_i|w_{i-N+1:i-1})$ is the *relative frequency* of the sequence $w_{i-N+1:i}$ compared to its immediate predecessor sequence $w_{i-N+1:i-1}$. For example, if the phrase "the cat is cute" appears 8 times and its immediate predecessor "the cat is" appears 10 times in the training corpus, the MLE for the conditional probability $P(\text{"cute"}|\text{"the cat is"})$ is $8/10 = 0.8$.

For a new text sequence s that does not appear in the training corpus, its probability can be estimated using $P(s; \hat{\theta})$, where $\hat{\theta}$ is the set of MLEs derived from the training sequences.

3.2.3 Word Embedding

In Section 2.4.4 of Chapter 2, we define an embedding in machine learning as a transformation that aims to represent data in a vector space. Among these techniques is word embedding, specifically designed to represent words in some vector space. In the literature, the plural form, "word embeddings", appears more frequently, which generally refers to the "outputs" of this transformation, namely, the word vectors.

A fundamental method for representing words as vectors is known as *one-hot encoding*. In this approach, the dimensionality d of the word vector corresponds to the total number of unique words in the vocabulary V.

Definition 3.2.8 (One-hot Encoding)

Assigning each word in V a unique index i, the one-hot encoding for the ith word w_i is expressed as:

$$\mathbf{w}_i = [\delta_{i1}, \delta_{i2}, \ldots, \delta_{id}]^T$$

where δ_{ij} represents the Kronecker delta, defined as 1 when $i = j$ and 0 otherwise; the bold letter $\mathbf{w}_i$ indicates the vector representation of w_i.

Although one-hot encoding offers a straightforward method for representing words, it results in a **sparse** representation where each word is treated in isolation, failing to capture any semantic relationships or associations between words. For example, the cosine similarity between two embeddings $\mathbf{w}_i$ and $\mathbf{w}_j$ is zero. Addressing this limitation, researchers have delved into the development of **dense** word embeddings. These embeddings are learned through neural networks optimised on word distributions across extensive text corpora, thereby uncovering underlying semantic connections between words.

Word2Vec is a prominent model for generating dense word embeddings and offers two variants: the *skip-gram* and *continuous bag of words (CBOW)* models. Here, we focus on the skip-gram model.

Consider the sentence "the bank robber was seen on the river bank". By constructing a *context window* with a *window size* of 2 (see their definitions in the next paragraph), we generate skip-grams as shown in Table 3.1. It is important to note that this process involves sliding the window across the sentence, word by word. For illustrative purposes, the table below demonstrates skip-gram generation only for the center words "bank", "robber", and "river".

Sentence	Skip-grams
"[the bank robber was] seen on the river bank"	(bank, the) (bank, robber) (bank, was)
"[the bank robber was seen] on the river bank"	(robber, the) (robber, bank) (robber, was) (robber, seen)
"the bank robber was seen [on the river bank]"	(river, on) (river, the) (river, bank)

Table 3.1: Skip-gram examples with a window size of 2 from the sentence "the bank robber was seen on the river bank".

Unlike the N-gram model, which predicts the current word based on the preceding $N-1$ words, the skip-gram model predicts the surrounding context words for a given center word.

Definition 3.2.9 (Skip-gram)

Let c denote the the number of words considered on each side of the center word w_i, known as the *windows size*, a skip-gram is defined as a pair comprising the center word and a context word within the window. The likelihood function for this model is expressed as:

$$L(\theta) = \prod_{i=1}^{n} \prod_{-c \leq j \leq c, j \neq 0} P(w_{i+j} | w_i; \theta)$$

where $1 \leq i + j \leq n$ ensures that the context words w_{i+j} are within the sentence bounds. θ here represents all learnable parameters in the neural network.

The conditional probability $P(w_{i+j}|w_i;\theta)$ is often computed using the *softmax* function:

$$P(w_{i+j}|w_i;\theta) = \frac{\exp(\mathbf{w}_{i+j}^T \mathbf{w}_i)}{\sum_{w \in V} \exp(\mathbf{w}^T \mathbf{w}_i)}$$

where $\mathbf{w}_i$ is the embedding of the center word (as an input), $\mathbf{w}_{i+j}$ is the embedding of the context word (as an output), $\mathbf{w}$ is the embedding of a word w in V.

To optimise the model, we minimise the *negative log-likelihood* (NLL) loss:

$$\begin{aligned} \mathcal{L} &= -\log L(\theta) = -\sum_{i=1}^{n} \sum_{-c \leq j \leq c, j \neq 0} \log P(w_{i+j}|w_i;\theta) \\ &= -\sum_{i=1}^{n} \sum_{-c \leq j \leq c, j \neq 0} \left(\mathbf{w}_{i+j}^T \mathbf{w}_i - \log(\sum_{w \in V} \exp(\mathbf{w}^T \mathbf{w}_i)) \right) \end{aligned}$$

which is equivalent to maximising the likelihood function. Using the log form prevents numerical underflow by transforming the product of probabilities into a sum of log probabilities.

The skip-gram model's architecture can be simply described as a two-layer neural network. The first layer, a hidden layer, implements a linear mapping of shape $|V| \times D$, where $|V|$ denotes the vocabulary size and D represents the dimensionality of the word embeddings. When an one-hot encoded vector of a word w from the vocabulary V is

input into the network, this layer outputs the word embedding $\mathbf{w}$ for the input word. The weights of the hidden layer are the word embeddings, where each column (or row, depending on the implementation) in the weight matrix corresponds to a word's embedding in V. The subsequent output layer involves another linear transformation of shape $D \times |V|$, mapping the hidden layer's output to a $|V|$-dimensional space, which is then passed through a softmax function to $\mathcal{L}$ for model optimisation.

However, computing the softmax normalisation term $\sum_{w \in V} \exp(\mathbf{w}^T \mathbf{w}_i)$ for each center word w_i is computationally intensive, particularly with large vocabularies. To address this, *negative sampling* is frequently utilised as an efficient approximation. In this approach, rather than determining the probability of w_{i+j} being in the context of w_i across the entire vocabulary, the model determines whether a specific word is a context word. This binary classification is formulated as follows.

Definition 3.2.10 (Negative Sampling)

Suppose for each positive word-context pair, we have sampled k negative samples, the objective function of negative sampling is defined as:

$$\mathcal{L}_{ij} = -\left(\log \sigma(\mathbf{w}_{i+j}^T \mathbf{w}_i) + k\mathbb{E}_{w' \sim P_{noise}(w)}[\log \sigma(-\mathbf{w}'^T \mathbf{w}_i)]\right)$$

where $\sigma(\cdot)$ denotes the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$, $\mathbb{E}(\cdot)$ refers to the expectation function, and $\mathbf{w}'$ represents the embeddings of the negative context words drawn from some noise distribution $P_{noise}(w)$.

In essence, negative sampling can be interpreted as a simplified variant of Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010), a method that, theoretically, is closely related to MLE.[5] Importantly, empirical evidence demonstrates that negative sampling extends its utility beyond word embeddings, proving to be efficient and effective in a diverse array of applications.

[5]For further insights, refer to the detailed discussion in the blog post at `https://www.yuanhe.wiki/post/softmax`.

The other variant of Word2Vec, i.e., the CBOW model, can be thought of as a "reversed version" of the skip-gram model; it predicts a center word based on a given set of context words, as opposed to predicting context words from a center word in skip-gram.

Among the significant variants of Word2Vec are GloVe (Pennington et al., 2014) and FastText (Bojanowski et al., 2017). GloVe integrates matrix factorisation with local context window approaches to capture *global* word-word co-occurrence patterns, while FastText (Bojanowski et al., 2017) builds upon the Word2Vec framework by representing each word as a collection of character N-grams. This enables FastText to grasp sub-word level semantics and more effectively manage rare words and linguistic variations. Such embedding techniques were also extended from word or token-level to sentence-level and document-level granularities (Arora et al., 2017; Le and Mikolov, 2014).

However, a shared limitation of these techniques is that each word is encoded with a static, context-independent embedding. This single embedding per word, predominantly reflecting the most common meaning in the training dataset, overlooks the nuances of context-specific usage.

3.2.4 Recurrent Neural Network

Recurrent Neural Networks (RNNs) (Rumelhart et al., 1985) are a type of neural networks particularly effective for processing sequential data due to their ability to maintain a memory of previous inputs. This feature is invaluable in language modelling, where context and sequence history play crucial roles.

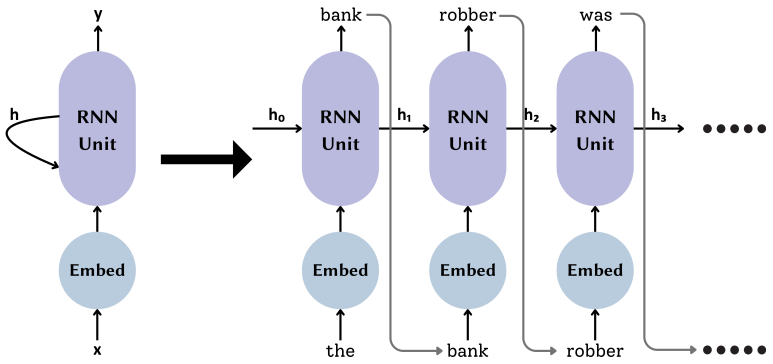


Figure 3.2: A simple illustration of an RNN-based language model.

Figure 3.2 illustrates the structure of an RNN-based LM. At each time step i, the model updates its hidden state and generates a prediction, typically corresponding to the next word in a sequence. The "Embed" block refers to a word embedding function that transforms input words into vector representations. The "RNN Unit" represents the heart of the RNN, performing the recurrent computations that update the hidden state based on the current input and past information.

Definition 3.2.11 (Recurrent Computations in RNNs)

The recurrent computations at each time step i are defined as follows:

$$\mathbf{h}_i = f_h(\mathbf{h}_{i-1}, f_e(w_{i-1})) = f_h(\mathbf{h}_{i-1}, \mathbf{w}_{i-1})$$
$$\hat{\mathbf{y}}_i = f_o(\mathbf{h}_i)$$

where:

- $\mathbf{h}_i$ is the hidden state at time step i; the initial state $\mathbf{h}_0$ is commonly initialised with a random or zero vector.
- f_e maps the input word w_{i-1} to its embedding vector $\mathbf{w}_{i-1}$.
- f_h is the function updating the hidden state, blending the previous state $\mathbf{h}_{i-1}$ with the current input $\mathbf{w}_{i-1}$.
- f_o is the output function that produces the predictive distribution $\hat{\mathbf{y}}_i$, a probability vector over the vocabulary V indicating the likelihood of each word being the next word in the sequence.

The parameters θ in the RNN encompass the learnable parameters within f_e, f_h, and f_o. If word embeddings from f_e are pre-trained and not updated, they are excluded from θ. Relating this to the N-gram model's likelihood function:

$$L(\theta) = P(s;\theta) = \prod_{i=1}^{n} P(w_i|w_{<i};\theta)$$

the RNN language model replaces $P(w_i|w_{<i};\theta)$ with the predicted probability $\hat{y}_{i,w_i}$.[6]

Definition 3.2.12 (Cross-entropy Loss)

Taking an average of the NLL loss over the sequence length n, we obtain the cross-entropy loss expressed as:

$$\mathcal{L} = \frac{1}{n}\left(-\log L(\theta)\right) = -\frac{1}{n}\sum_{i=1}^{n}\log \hat{y}_{i,w_i}$$

RNN-based LMs can be seen as a generalisation of N-gram LMs. While N-gram models are limited by a fixed context size $N-1$, RNNs theoretically have an infinite context size, as they are capable of maintaining a hidden state that theoretically can carry information from arbitrarily distant parts of the input sequence.

However, basic RNNs can struggle with learning long-range dependencies due to issues like *vanishing gradients*. This challenge is partially addressed by more advanced variants like Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) networks (Cho et al., 2014), which introduce the gating mechanism to better control the flow of information and allow the model to retain important information over longer sequences.

3.2.5 Encoder-Decoder Architecture

In sequence modelling, RNNs with the set-ups described in the last section commonly serve as a *decoder*, a term to denote a component for making sequential predictions. In contrast, word embedding techniques like Word2Vec function as an *encoder*, aimed at transforming words into dense vector representations. Interestingly, RNNs can also adopt the role of an encoder, encapsulating an entire sequence into its final hidden state, h_n.

During training, when the complete sequence is accessible, it is feasible to leverage *bidirectional* RNNs (biRNNs). This approach involves processing the sequence in both forward and backward directions, yielding two sets of hidden states that are typically

[6] $\hat{y}_{i,w_i}$ is an element of $\hat{\mathbf{y}}$ that corresponds to the probability of predicting w_i.

concatenated or combined to form a comprehensive representation of each time step in the sequence.

In 2014, Sutskever et al. (2014) proposed the Seq2Seq (Sequence-to-Sequence) model, which is essentially an encoder-decoder architecture. The encoder part is an RNN (or biRNN) that processes the input sequence and compresses the information into a context vector (usually the final hidden state). The decoder part is another RNN (but not biRNN as there is no complete sequence in decoding time) that takes the context vector as its initial state and generates the output sequence token by token. The *beam search* (Freitag and Al-Onaizan, 2017) algorithm is commonly adopted for selecting the most probable sequence. A typical application of the Seq2Seq model is in machine translation, where the encoder transforms a source language sentence into a context vector, and the decoder predicts a target language sentence based on this context vector.

3.2.6 Attention Mechanism

The attention mechanism is a pivotal enhancement to the encoder-decoder architecture. It enables the model to dynamically focus on different parts of the input sequence during the decoding process, improving the context sensitivity of the model.

Definition 3.2.13 (Attention)

An attention function, $f_{\mathsf{attn}} : \mathbb{R}^k \times (\mathbb{R}^k \times \mathbb{R}^v)^n \to \mathbb{R}^v$, accepts a query vector $\mathbf{q} \in \mathbb{R}^k$ and a set of n key-value vector pairs $\{(\mathbf{k} \in \mathbb{R}^d, \mathbf{v} \in \mathbb{R}^v)\}_{i=1}^n$ as input arguments, and it outputs a weighted sum of the values:

$$f_{\mathsf{attn}}(\mathbf{q}, \{(\mathbf{k}_i, \mathbf{v}_i)\}_{i=1}^n) = \sum_{i=1}^n f_{\mathsf{sim}}(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i$$

where $f_{\mathsf{sim}}(\cdot)$ is a *normalised* scoring function evaluating the *similarity* (a scalar value) between the query $\mathbf{q}$ and the key $\mathbf{k}_i$. It is worth noting that the query and key vectors have the same shape, both from $\mathbb{R}^k$.

A frequently used scoring function is the composition of the dot product and the softmax function (Luong et al., 2015), expressed as:

$$f_{\mathsf{sim}}(\mathbf{q}, \mathbf{k}_i) = \mathsf{softmax}(\mathbf{q}^T \mathbf{k}_i)$$

In Seq2Seq models, consider the encoder hidden states $\mathbf{h} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n]$, where n represents the input sequence length, and each $\mathbf{h}_i$ corresponds to the hidden representation at position i. These states serve as the keys and values in the attention mechanism. During decoding, to predict the jth word, the decoder's previous hidden state $\mathbf{d}_{j-1}$ acts as the query. The model computes the similarity scores between $\mathbf{d}_{j-1}$ and each $\mathbf{h}_i$, which results in a set of importance weights for the encoder hidden states. The context vector (attention score) at time step j is computed as:

$$f_{\mathsf{attn}}(\mathbf{d}_{j-1}, \{(\mathbf{h}_i, \mathbf{h}_i)\}_{i=1}^n) = \sum_{i=1}^{n} f_{\mathsf{sim}}(\mathbf{d}_{j-1}, \mathbf{h}_i)\mathbf{h}_i$$

This process dynamically adjusts the encoder's hidden states based on their relevance at the current decoding step. Subsequently, the decoder combines $\mathbf{q}_{j-1}$ and $\mathbf{a}_j$ to predict the jth word and update its hidden state.

3.2.7 Evaluating Language Models

Up to this point, we have reviewed a range of essential works in the literature of language modelling, including the N-gram, word embedding, negative sampling, RNN, encoder-decoder, and the attention mechanism. With these preliminaries, we are well-prepared to delve into the discussion of the state-of-the-art transformer-based LMs. However, before we embark on this exploration, it is crucial to establish an understanding of basic machine learning principles and how to evaluate the performance of LMs.

Training, Validation, and Testing Machine learning models, including LMs, are typically evaluated in three stages: *training*, *validation*, and *testing*. During training, the model learns from a dataset, adjusting its parameters to minimise a pre-defined loss function (or equivalently, to maximise the likelihood). However, to prevent overfitting (where the model learns the training data too well and performs poorly on unseen data), we use a validation set. The validation set helps in tuning the model's *hyperparameters*

and provides an unbiased evaluation of the model's performance during the training process. Hyperparameters, which are parameters not directly learned in model training, include configurations like the learning rate (a weight applied to control the gradient update in backpropagation), the number of epochs (complete passes through the training dataset), and more. After the model is trained and validated, it is tested on a separate, hold-out dataset, i.e., the test set, which **has not been seen** by the model before. This stage provides an unbiased assessment of the model's generalisation capability.

Cross-validation is another evaluation technique, particularly useful when the dataset is limited in size. The most common form is k-fold cross-validation, where the data is divided into k subsets. The model is trained on $k-1$ subsets and validated on the remaining one, and this process is repeated k times so that each subset is used for validation exactly once. The performance measure is then averaged over the k trials. Although powerful, cross-validation can be computationally intensive, particularly with large datasets and complex models, making it less feasible for some large-scale applications in modern machine learning scenarios.

Intrinsic Evaluation concerns the internal evaluation of language modelling. One of the most widely used intrinsic evaluation metrics is the *perplexity*, measuring the model's proficiency in predicting a text sequence. Typically, after partitioning a text corpus into training, validation, and test sets, an LM is trained and validated using the negative log-likelihood (NLL) loss $\mathcal{L} = -\log L(\theta)$ to obtain the optimal parameters $\hat{\theta}$. Perplexity is calculated on the test set to assess the model's prediction capabilities on unseen data.

Definition 3.2.14 (Perplexity)

Perplexity is defined as the exponential of the average NLL, formulated as:

$$\mathsf{ppl}(s) = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log P(w_i|w_{<i};\hat{\theta})\right)$$

where $s = w_{1:n}$ is a text sequence from the test set.

From the perspective of information theory, perplexity quantifies the average branching factor of the LM, or in other words, how many equally probable words the model is uncertain about at each step in the sequence (Whittaker, 2000). A lower perplexity value indicates greater predictive confidence of the model about the "naturalness" of a text sequence.

While perplexity is a prevalent intrinsic evaluation metric, there are additional metrics that target various facets of language modelling. For models generating word embeddings, like Word2Vec or GloVe, similarity-based metrics can be adopted. Word pairs with similar meanings should have embeddings that are close to each other in the vector space, often measured by cosine similarity. Some LMs (typically the transformer-based) can also be evaluated using cloze-style tests, where the model is tasked with predicting a missing word in a sentence given the surrounding context.

Extrinsic Evaluation assesses the performance of LMs within the scope of specific downstream tasks. This approach is crucial for understanding how improvements in language modelling translate to enhancements in practical applications. Common extrinsic evaluations involve deploying LMs in tasks like machine translation, sentiment analysis, and syntax parsing. In machine translation, for example, the model's effectiveness is measured by how accurately it translates text compared to human translations, often measured by metrics like BLEU.

Current LMs have been applied beyond the scope of NLP. In the context of this thesis, which focuses on ontology engineering, extrinsic evaluation could involve assessing how well an LM supports tasks such as ontology alignment, ontology completion, and more.

3.3 Transformer-based Language Models

3.3.1 Transformer

The transformer architecture, introduced by Vaswani et al. (2017), represents a paradigm shift in sequence modelling, moving away from the RNNs that were previously predominant. Unlike RNNs, the transformer adopts an encoder-decoder structure that **relies solely on attention mechanisms**, which enhances training parallelisation and improves the model's ability to handle long-range dependencies in text.

At the core of the transformer is the *self-attention* mechanism, a variant of the attention mechanisms discussed earlier in Section 3.2.6. Recall that an attention function takes a query and a set of key-value pairs as input; in self-attention, the query, keys, and values, all come from the input sequence, but subject to different linear transformations.

Definition 3.3.1 (Self-Attention)

Consider an input sequence $s = (w_1, \ldots, w_n)$, where each token w_i is represented by an embedding $\mathbf{w}_i \in \mathbb{R}^d$. The self-attention function computes the attention for each embedding $\mathbf{w}_i$ by treating it as a query vector, while simultaneously considering the entire sequence s for generating corresponding keys and values:

$$\mathbf{q} = W_q \mathbf{w}_i, \quad \mathbf{k}_j = W_k \mathbf{w}_j, \quad \mathbf{v}_j = W_v \mathbf{w}_j$$

where $W_q \in \mathbb{R}^{k \times d}$, $W_k \in \mathbb{R}^{k \times d}$, and $W_v \in \mathbb{R}^{v \times d}$ are the weight matrices for queries, keys, and values, respectively. The self-attention function for the ith token in the sequence is then given by:

$$\begin{aligned} f_{\mathsf{selfattn}}(i, s) &= f_{\mathsf{attn}}(W_q \mathbf{w}_i, \{(W_k \mathbf{w}_j, W_v \mathbf{w}_j)\}_{j=1}^n) \\ &= \sum_{j=1}^{n} f_{\mathsf{sim}}(W_q \mathbf{w}_i, W_k \mathbf{w}_j) W_v \mathbf{w}_j \end{aligned}$$

In the original implementation by Vaswani et al. (2017), the scoring function $f_{\mathsf{sim}}(\cdot)$ employs a mechanism akin to the dot product attention introduced by Luong et al. (2015) but subject to a scaling factor $\sqrt{d_k}$. This scaling is applied to the dot product scores to mitigate the vanishing gradients issue that arises as the dimensionality d_k of the keys increases. Mathematically, it can be expressed as:

$$f_{\mathsf{sim}}(W_q \mathbf{w}_i, W_k \mathbf{w}_j) = \mathsf{softmax}\Big(\frac{(W_q \mathbf{w}_i)^T (W_k \mathbf{w}_j)}{d_k}\Big)$$

Self-attention allows each token in the input sequence to interact with every other token, assessing their mutual relevance. To further augment the model's capacity

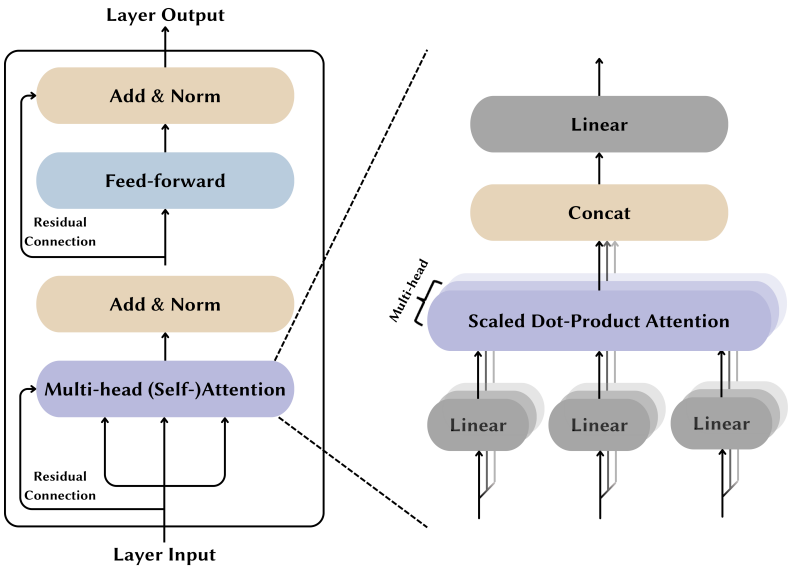


Figure 3.3: Illustration of a transformer layer.

to capture various aspects of data, transformers employ *multi-head attention*. This approach involves performing self-attention multiple times with different, learnable linear transformations to the input data. The results of these multiple attention processes are then concatenated, providing the model with a richer, more nuanced understanding of the input, akin to an ensemble method within a single model framework.

Figure 3.3 illustrates a concrete attention layer in the transformer. The main component, **multi-head (self-)attention**, is highlighted and expanded for detailed examination on the right side of the figure. The **residual connection** means to carry forward the input directly to latter computational steps. The **Add & Norm** block signifies the operation where the residual is added to the output of the attention mechanism or the feed-forward network, followed by layer normalisation across the embedding dimension.

One inherent limitation of transformers is their lack of inherent sequence order awareness – a critical feature in sequence modelling. To address this, transformers use *positional encoding*, initially implemented with sine and cosine functions, to imbue the model with information about the token positions in the sequence.

The depth of the transformer is another defining characteristic, with several layers of attention and feed-forward networks stacked in both the encoder and decoder. This

deep architecture allows the transformer to learn complex patterns and relationships within the data, enhancing its predictive capabilities.

3.3.2 Pre-training

The term *pre-training* has gained prominence with the advent of transformer-based LMs. Pre-training refers to initialising the parameters θ of LMs by training them on extensive text corpora using specific language modelling objectives. Given their deep-layered architecture and vast parameter space, transformer-based models necessitate effective pre-training to ensure a robust initialisation, which is critical for their subsequent adaptation to downstream tasks.

A significant leap in this domain was made by Kenton and Toutanova (2019) with the introduction of BERT (Bidirectional Encoder Representations from Transformers), a transformer encoder-based LM. BERT's main pre-training objective, Masked Language Modelling (MLM), shares a conceptual resemblance with CBOW, as both techniques aim to deduce a word from its context. However, MLM is not limited to a fixed window size and the resultant word embeddings are contextualised. Below, we delve into the specifics of MLM:

Definition 3.3.2 (Masked Language Modelling)

Let $s = (w_1, w_2, \ldots, w_n)$ be an input sequence of length n. During MLM, certain tokens in s are randomly masked, leading to the creation of two subsets: $U = \{u_1, \ldots, u_{n_u}\}$, representing indices of unmasked tokens, and $M = \{m_1, \ldots, m_{n_m}\}$, representing indices of masked tokens, where $n_u + n_m = n$. The MLM loss is then defined as:

$$\mathcal{L}_{\mathsf{MLM}} = -\sum_{m_i \in M} \log P(w_{m_i} | w_U; \theta)$$

where $P(w_{m_i} | w_U; \theta)$ denotes the probability of predicting the masked word w_{m_i} given the unmasked words w_U under the model parameters θ.

Since BERT utilises an encoder-only transformer architecture, the probability of

predicting a masked token w_{m_i} is dynamically adjusted based on the attention scores allocated to the context words w_U. This adaptability ensures that the token embeddings BERT generates are contextually aware, providing a significant enhancement over earlier static word embedding methods like Word2Vec. The term "token embeddings" is preferred over "word embeddings" due to BERT's use of a sub-word level tokeniser, which offers two main advantages: *(i)* it effectively addresses the out-of-vocabulary (OOV) issue by representing unknown words as sequences of known sub-words, and *(ii)* it captures word morphology more accurately.

In BERT's pre-training, an input sequence s is formulated as:

$$s = [\mathsf{CLS}]\ s_A\ [\mathsf{SEP}]\ s_B\ [\mathsf{SEP}]$$

where s_B is a segment that follows s_A in the training corpus. Here, [CLS] is a special token placed at the beginning of every input sequence, serving as an aggregate representation for the whole sequence, while [SEP] is a delimiter token that separates different segments within the sequence. In MLM, specific portions of s_A and/or s_B are masked (represented as [MASK]), prompting the model to predict these masked tokens.

Besides, BERT incorporates a secondary pre-training task known as Next Sentence Prediction (NSP). This task utilises the embedding of the [CLS] token to perform a binary classification, determining whether the segment s_B naturally follows s_A. Although NSP was part of BERT's original formulation, subsequent research, including the development of RoBERTa by (Liu et al., 2019), has shown that focusing solely on MLM can yield models of comparable or superior performance.

It is also important to note that BERT and its derivatives can accommodate additional custom special tokens beyond [CLS] and [SEP], allowing for flexibility and adaptability to a wide range of NLP tasks and domain-specific requirements.

3.3.3 Fine-tuning

The term *fine-tuning* emerged along with the introduction of BERT, indicating a distinct training phase subsequent to pre-training. This process is designed to adapt a pre-trained LM to a specific downstream task using a smaller, task-oriented dataset. Classic fine-tuning incorporates the following characteristics:

- **Task-specific layers:** To cater to the unique demands of the downstream task, additional layers can be integrated into the pre-trained model. This customisation facilitates task-specific processing and output generation.

- **Parameter adjustment:** Fine-tuning involves modifying either all or a portion of the model's parameters. It is common practice to freeze certain layers of the model to retain learned knowledge, while only training specific layers or adapters (Houlsby et al., 2019) that are crucial for the task at hand.

- **Training efficiency:** Since the model has already developed a foundational understanding of language during pre-training, fine-tuning usually requires significantly fewer training epochs and resources.

In a typical BERT fine-tuning scenario for a classification task, such as sentiment analysis, the process is straightforward.

Definition 3.3.3 (Fine-tuning for Classification)

Given an input sequence s with a head token $[\mathsf{CLS}]$, after encoding s with BERT, the embedding of $[\mathsf{CLS}]$ is denoted as $\mathbf{w}_{[\mathsf{CLS}]}$. The classification task in BERT's fine-tuning can be formulated as:

$$\hat{\mathbf{y}} = \mathsf{softmax}\left(f(\mathbf{w}_{[\mathsf{CLS}]})\right)$$

where f represents a task-specific network (typically a simple feed-forward function like a linear transformation), and $\hat{\mathbf{y}}$ refers to the predicted probability distribution over classification labels.

In the context of sentiment analysis, $\hat{\mathbf{y}}$ might contain the probabilities associated with the positive and negative sentiments.

Fine-tuning is an effective method for adapting pre-trained LMs to various specific tasks, leveraging the comprehensive linguistic knowledge these models have acquired.

However, this process is not without its pitfalls, particularly concerning *overfitting* and *catastrophic forgetting*.

- **Overfitting:** During fine-tuning, there is a risk that the model may become too specialised to the task-specific dataset, leading to a loss of generalisability. This overfitting means that while the model may perform exceptionally well on the training data or similar types of data, its ability to generalise to different datasets or tasks could be significantly diminished.
- **Catastrophic Forgetting:** This phenomenon refers to the model's tendency to "forget" the general language understanding it gained during the pre-training phase. As the model is fine-tuned on a specific task, it may overwrite or diminish some of the general knowledge it previously acquired, which could be detrimental if the model needs to be applied to other tasks or broader contexts in the future.

Addressing these challenges requires careful fine-tuning practices, such as selecting an appropriate amount of task-specific data, employing regularisation techniques to prevent overfitting, and possibly using methods like elastic weight consolidation (Kirkpatrick et al., 2017) to mitigate catastrophic forgetting.

Researchers have also explored alternative paradigms for fine-tuning other than the classic approach discussed in this section. In the following two sections, we will explore two other prevalent approaches commonly employed in the fine-tuning process.

3.3.4 Contrastive Learning

As discussed in Section 3.2.3, negative sampling, or more broadly, noise contrastive estimation, is utilised to enhance word embeddings, ensuring that semantically related words are located closer together in the embedding space. This method falls under a wider machine learning paradigm known as *contrastive learning*. Contrastive learning seeks to enhance embeddings (often at the sentence level) thereby offering a robust mechanism for fine-tuning or even pre-training LMs to capture more nuanced semantic relationships (Reimers and Gurevych, 2019; Gunel et al., 2020; Gao et al., 2021b). LMs trained on such learning objectives are particularly adept at tasks like semantic search and paraphrasing.

There are various formulations of contrastive loss used in different contexts. Below, we define a commonly employed form of contrastive loss and then introduce a triplet-based variant.

Definition 3.3.4 (Contrastive Loss)

Given a training dataset $\mathcal{D}$ consisting of sentence pairs (s, s'), each labelled as semantically similar ($y = 1$) or dissimilar ($y = 0$), the contrastive loss is defined as:

$$\mathcal{L} = \sum_{(s,s')\in\mathcal{D}} \Big(y \cdot f^{+}(d(\mathbf{s}, \mathbf{s}')) - (1 - y) \cdot f^{-}(d(\mathbf{s}, \mathbf{s}')) \Big),$$

where d is a distance metric between sentence embeddings $\mathbf{s}$ and $\mathbf{s}'$, f^{+} and f^{-} are *monotonically increasing* functions applied on the distances for positive and negative pairs, respectively.

The reason that f^{+} and f^{-} are both monotonically increasing is that they should not change the intuitive understanding of distances, thereby ensuring that the contrastive loss increases with the distances for positive pairs and decreases with the distances for negative pairs.

It is common practice to introduce a margin to the distances between negative pairs, implementing f^{-} as $f^{-}(d(\mathbf{s}, \mathbf{s}')) = -\max(\gamma - d(\mathbf{s}, \mathbf{s}'), 0)$, where γ is the margin. For positive pairs, f^{+} is typically the identity function, i.e., $f^{+}(d(\mathbf{s}, \mathbf{s}')) = d(\mathbf{s}, \mathbf{s}')$. This set-up enforces a minimum separation between negative pairs while maintaining the original distances for positive pairs. The loss function is thus formulated as:

$$\mathcal{L} = \sum_{(s,s')\in\mathcal{D}} \Big(y \cdot d(\mathbf{s}, \mathbf{s}') + (1 - y) \cdot \max(\gamma - d(\mathbf{s}, \mathbf{s}'), 0) \Big)$$

In certain cases, it would be beneficial to introduce a relaxation to the distances between positive pairs to facilitate smoother and more efficient optimisation. This can be done by accommodating a margin for positive pairs. The loss function that incorporates margins for both positive and negative pairs is formulated as:

$$\mathcal{L} = \sum_{(s,s')\in\mathcal{D}} \Big(y \cdot \max(d(\mathbf{s}, \mathbf{s}') - \gamma^{+}, 0) + (1 - y) \cdot \max(\gamma^{-} - d(\mathbf{s}, \mathbf{s}'), 0) \Big)$$

where γ^+ and γ^- are margins for positive and negative pairs, respectively.

We can enforce even more relaxation by focusing on increasing the *relative distances* between positive pairs and negative pairs, rather than the absolute distances. The *triplet*-based contrastive loss is designed for this purpose.

Definition 3.3.5 (Triplet Contrastive Loss)

Given a training dataset $\mathcal{D}$ composed of triplets (s, s^+, s^-), where s is an anchor sentence, s^+ is semantically related (positive) to s, and s^- is semantically unrelated (negative) to s, the triplet contrastive loss is defined as:

$$\mathcal{L} = \sum_{(s,s^+,s^-)\in\mathcal{D}} \max(f^+(d(\mathbf{s}, \mathbf{s}^+)) - f^-(d(\mathbf{s}, \mathbf{s}^-)), 0)$$

where d, f^+, f^- have the same definitions as in Definition 3.3.4.

Again, it is prevalent to incorporate a margin in f^- and let f^+ be the identity function. With this, the triplet contrastive loss becomes:

$$\mathcal{L} = \sum_{(s,s^+,s^-)\in\mathcal{D}} \max(d(\mathbf{s}, \mathbf{s}^+) - d(\mathbf{s}, \mathbf{s}^-) + \gamma, 0)$$

Having explored the standard contrastive learning losses, we now turn our attention to the representation of sentence embeddings. In the previous section, we discussed how the embedding of the [CLS] token serves as a representative for the entire input sequence, a process known as *pooling*. Besides this method, other prevalent pooling techniques include mean pooling and max pooling. Mean pooling involves calculating the average embedding across all tokens, while max pooling involves selecting the maximum value across each dimension of the embeddings. It is also possible to design a more complex network to learn the pooling function. These methods provide different ways to distill the essence of the sentence into a single, coherent embedding.

Finally, we provide some insights to the distance metric. Mathematically, a distance metric must adhere to several key properties: *(i)* non-negativity, where $d(x, y) \geq 0$ and $d(x, y) = 0$ if and only if $x = y$; *(ii)* symmetry, implying that $d(x, y) = d(y, x)$; and *(iii)*

the triangle inequality, which states that $d(x, y) + d(y, z) \geq d(x, z)$. A classic example of a distance metric in Euclidean space is the L2-norm, denoted as $\|\cdot\|_2$. However, in the context of contrastive learning, strict adherence to these mathematical properties is not always necessary. Take, for instance, the cosine distance, defined as $1 - \cos(\theta)$, where θ is the angle between two vectors. While this measure provides valuable insights into the orientation and similarity of vectors, it does not satisfy the triangle inequality.

3.3.5 Prompt Learning

Classic fine-tuning methods seek to adapt pre-trained LMs to specific downstream tasks. This approach, however, raises an interesting question: *Why not tailor the tasks to align with the pre-training objectives of the LMs, thus better leveraging their vast reservoir of acquired knowledge?* Motivated by this line of thought, the *prompt learning* paradigm has been proposed. To delve deeper into this area, readers are recommended to consult the comprehensive survey by Liu et al. (2023). This section introduces the core principles of prompt learning.

The process of prompt learning typically involves two steps: *(i)* adapting a task to align with a prompt-based language modelling objective, and *(ii)* translating the model's predicted responses into the final output space for the task. The first step necessitates crafting a *template*, a structured approach that integrates the task's specifics into a format the LM can process directly.

Definition 3.3.6 (Template)

A template $T([\mathsf{X}], [\mathsf{A}])$ is a function that, given an input slot $[\mathsf{X}]$ and an answer slot $[\mathsf{A}]$, generates a prompt-augmented sequence. Replacing $[\mathsf{X}]$ with an input text sequence s results in a new sequence $s' = T(s, [\mathsf{A}])$, where $[\mathsf{A}]$ remains as a placeholder for the model's predicted output.

In a template function, a *hard* (or *discrete*) prompt refers to specific natural language instructions, while a *soft* (or *continuous*) prompt operates directly in the embedding

space, influencing the LM's hidden representations. The methodology of designing a suitable and effective prompt is referred to as *prompt engineering*.

With the definition of the template and prompt, we can now define the loss function for prompt-based language modelling as follows:

Definition 3.3.7 (Prompt-based Language Modelling)

Given a template function T, an input sequence s, and a true answer sequence α, the prompt-based language modelling loss is defined as:

$$\mathcal{L}_{\text{prompt}} = -\log P(\alpha|T(s, [\mathsf{A}]); \theta) = -\sum_{i=1}^{|\alpha|} \log P(a_i|T(s, [\mathsf{A}]), a_{<i}; \theta))$$

where θ represents the parameters of the LM, a_i is the ith token in α, and $a_{<i} = a_{1:i-1}$ denotes the tokens in α before position i.

The second step in the process involves associating the answer sequence α with the task's final output space. This association is achieved through a function referred to as *answer mapping*, or *verbaliser*. We prefer the term "answer mapping" over "verbaliser" to avoid confusion with ontology verbalisation, which is the process of converting ontology axioms into natural language text.

Definition 3.3.8 (Answer Mapping)

Answer mapping is a function $V : A \to Y$ that transforms an answer sequence $\alpha \in A$ into the final output $y \in Y$.

In some cases, answer mapping can be trivial, especially when the answer sequence directly corresponds to the final output, as seen in tasks like paraphrase generation. Conversely, in tasks such as classification, the answer sequence needs to be mapped to a specific class label. For example, in sentiment analysis, both "good" and "great" could be mapped to a positive sentiment class. In this scenario, the answers are also called *label words*. The methodology of devising an appropriate answer mapping is referred to as *answer engineering*.

Prompt learning is not only a technique for fine-tuning LMs but is also widely used to extract knowledge from them. For instance, to determine the capital of the UK, one could employ the prompt "[MASK] is the capital of the UK." and prompt the LM to fill in the [MASK] slot. Similarly, to evaluate whether contracting COVID-19 likely leads to coughing – a scenario within the realm of Natural Language Inference (NLI) – one might use the prompt "If someone contracts COVID-19, it is highly probable that they will experience coughing. Answer: [MASK]."

Furthermore, a significant body of research is dedicated to tuning-free prompt learning, also known as *in-context learning*, wherein the LM's parameters remain unchanged. While in-context learning eliminates the risk of catastrophic forgetting and preserves the pre-trained knowledge of the LM, it demands extensive prompt design to achieve reliable results. This approach has gained particular relevance with the emergence of Large Language Models (LLMs). For such models, fine-tuning is often infeasible because it requires substantial computational resources, not to mention pre-training.

3.3.6 Large Language Models

The introduction of GPT-3 in 2020 (Brown et al., 2020) has set a scale of pre-trained LMs that was unprecedented, thereby marking the era of Large Language Models (LLMs). Predominantly, these LLMs utilise the **transformer decoder** architecture for sequential language modelling. They are also characterised as *autoregressive* LMs, stemming from their methodology of predicting each subsequent word based on the preceding sequence of words, thus forming a regressively dependent chain of predictions. In contrast to transformer encoder-based, or BERT-like models, which emphasise token embeddings to understand context, LLMs prioritise generating a continuous and contextually coherent flow of text. This feature renders them particularly adept at simulating conversational agents capable of interacting with humans in a naturalistic manner. Prominent examples of such chat agents, as of the time of this writing, include ChatGPT[7], Gemini[8], and Claude 3[9].

[7] https://chat.openai.com/
[8] https://gemini.google.com/app
[9] https://www.anthropic.com/claude

With their substantially expanded parameter size and context window capacity (despite theoretical capabilities of handling infinite sequences, practical constraints like hardware limitations must be considered), LLMs can process considerably longer and more informative prompts, such as specific task instructions. Typically, these models undergo fine-tuning – or *instruction-tuning* – across a broad spectrum of tasks, which equips them to function as general problem solvers.

Example 3.3.1

A typical instruction prompt follows the structure: "[role assignment] [task context] [task description]". Below is an illustrative task prompt[a] for enhancing a CV, used before the CV's content is uploaded.

> *"Assume the role of a recruiter. I will share details about job openings, and your task is to devise strategies for attracting qualified candidates. This might entail engaging with potential candidates via social media, networking events, or career fairs to identify the best individuals for each position. Initially, I require assistance with improving my CV."*

[a] This prompt was generated by ChatGPT, accessed on March 16, 2024.

To enhance the "reasoning"[10] capabilities of LLMs, the concept of a *chain-of-thought* (CoT) prompt was introduced (Wei et al., 2022). This technique is a type of *demonstration learning*, wherein LMs are instructed to tackle tasks sequentially, unraveling them step by step. This strategy is especially beneficial for tasks that require arithmetic computation or strategic planning. The effectiveness of CoT prompts can be further improved by employing *self-consistency* (Wang et al., 2022), a method akin to ensembling, where the collective output from multiple prompts is synthesised to derive a more reliable and consistent response. Building on the CoT framework, the *tree-of-thought* (ToT) approach was later introduced by Yao et al. (2024), which extends the linear CoT

[10] The form of "reasoning" exhibited by LLMs mainly pertains to identifying and leveraging patterns, contrasting with the structured logical reasoning associated with ontologies.

reasoning into a more complex, branching tree structure, offering a broader exploration of potential solution paths.

Despite their effectiveness, LLMs can produce *hallucinations*, generating outputs that seem plausible but are factually incorrect or logically inconsistent. To mitigate this, researchers have been exploring the *attribution* of responses generated by LLMs, aiming to trace the source or rationale behind the generated content (Yue et al., 2023; Gao et al., 2023; Sarti et al., 2023). A prominent approach in this domain is Retrieval-Augmented Generation (RAG) (Lewis et al., 2020b), which integrates LLMs with information retrieval systems to source relevant context or references from structured data, such as knowledge bases. RAG typically employs a framework that integrates transformer encoder-based LMs with transformer decoder-based LLMs. The encoder-based models serve as retrievals, conducting semantic searches to acquire and enhance context, while the decoder-based models are tasked with synthesising this enriched information into text generation. This collaborative framework enables the generated output to be both contextually rich and anchored in verifiable information, thus reducing the chance of hallucinations.

3.4 Language Models for Knowledge Engineering

After an extensive exploration of the evolution of LMs, tracing their journey from initial N-gram models to modern transformer-based architectures, this concluding background section aims to introduce relevant literature in LMs for knowledge engineering.

3.4.1 Pre-transformer Era

Before the advent of transformer-based models, the success of word embedding techniques such as Word2Vec inspired numerous approaches to construct KB embeddings, adapting the methodologies used for learning word embeddings to accommodate the structured nature of KBs. RDF2Vec (Ristoski and Paulheim, 2016) stands out as one of the notable methods in this domain, adapting the Word2Vec approach to RDF graphs. RDF2Vec generates embeddings by treating graph paths as sentences, using sub-graph extraction techniques like random walks or Weisfeiler-Lehman (WL) kernels (De Vries et al., 2013) to create sequences of nodes (entities) and edges (relations). These sequences

are then used as input for skip-gram or CBOW to produce vector representations of the entities and relations, which then facilitate tasks like entity typing and link prediction.

OWL2Vec* (Chen et al., 2021a) is an extension of RDF2Vec, tailored specifically for OWL ontologies. This approach employs a projection method to convert ontologies into RDF graphs, encapsulating concept hierarchies and logical relationships as triples to facilitate graph-walking algorithms akin to those used in RDF2Vec. In this process, the entity-relation sequences are represented using both IRIs and human-readable labels (`rdfs:label`) to form a combined document. The application of OWL2Vec* extends across various tasks, including ontology alignment, ontology completion, and specialised areas such as gene function prediction.

Besides KB embedding, there is also research focused on refining or fine-tuning[11] Word2Vec-like word embeddings for specific knowledge engineering tasks. One notable example is DeepAlignment (Kolyvakis et al., 2018), which employs the counter-fitting technique (Mrkšić et al., 2016) to infuse paraphrasing semantics from ontologies into word embeddings. Specifically, this method utilises synonyms and antonyms extracted from ontology concept labels to enhance the embeddings for the task of ontology alignment.

3.4.2 Transformer-based

The endeavor to incorporate knowledge into LMs has transitioned into the era of transformer-based models. In this section, we highlight several typical works that leverage KBs to pre-train, fine-tune, or probe transformer-based LMs.

Pre-training LMs with KBs Zhang et al. (2019) introduced ERNIE, a model that inherits the BERT architecture by modifying the masked language modelling (MLM) objective. They incorporated additional special tokens to facilitate entity-token alignment, guiding the model to predict placeholders indicating whether a token span corresponds to an entity.

[11] The term "fine-tuning" emerged later than Word2Vec, but here its meaning of the adaptation of pre-trained embeddings for specific downstream tasks is conveyed.

Bosselut et al. (2019) introduced COMET, a model based on the transformer decoder architecture (akin to GPT). COMET was directly pre-trained to generate triples to support KB completion. Specifically, graph triples of the form (subject, predicate, object) are transformed into natural language sentences, and the model is asked to generate tokens for object given tokens for subject and predicate.

Liu et al. (2020b) introduced K-BERT, which enhances the input sequence with relevant triples extracted from a knowledge graph (KG). For instance, given the sentence "Tim Cook is visiting China", K-BERT augments it to "Tim Cook CEO Apple is visiting China" with the grayed-out prompt derived from the triple (TimCook, isCEOOf, Apple). The model's attention mechanism is tailored so that the information from the triple is only utilised when computing attention scores for relevant tokens, i.e., "Tim" and "Cook".

Liu et al. (2021a) introduced the SapBERT model which adopts the sentence transformer architecture (Reimers and Gurevych, 2019). It treats the entity names as sentences, learning their embeddings using a contrastive loss where positive samples come from synonyms of entity names and negative samples are generated across entity distribution over UMLS, a comprehensive thesarus for biomedical ontologies (Bodenreider, 2004).

Melnyk et al. (2021) introduced Grapher, a model that addresses the challenge of KB construction from text. Grapher utilises the text-to-text transformer model (T5) (Raffel et al., 2020) as the basis to facilitate a two-stage process for KB construction. The first stage is to generate nodes representing entities, while the subsequent stage focuses on generating edges, which establish the relationships between the nodes.

Fine-tuning LMs with KBs Wu et al. (2020) introduced BLINK, an entity linking system that operates in two stages: retrieval and re-ranking. The system utilises a dual-BERT model approach, where the first BERT model functions as a bi-encoder, responsible for selecting candidate entities, and the second BERT model serves as a cross-encoder, tasked with refining these selections to identify the final linked entities. Dong et al. (2023b) (our work) extended BLINK to identify entities out of a KB with a system called BLINKOut and released relevant datasets (Dong et al., 2023a).

Tang et al. (2021) introduced BERT-INT, an entity alignment system that utilises BERT to embed multiple aspects of an entity – its name, description, attributes, and

neighbors – by treating each aspect as a separate sentence, which is subject to the sentence head ([CLS]) pooling. It then constructs the neighbor-view and attribute-view interaction models to compute interactions between these varied embeddings.

Ye et al. (2022) introduced Onto-Prompt, a methodology for few-shot fine-tuning that applies a unique template for each of three knowledge engineering tasks: relation extraction, event extraction, and KG completion. This approach infuses ontological information into the input sequence through these customised templates. Similar to the technique utilised in K-BERT, Onto-Prompt incorporates a visible attention mechanism, which ensures that the infused knowledge selectively interacts with the relevant portions of the input sequence.

Probing LMs with KBs Petroni et al. (2019a) introduced "LMs-as-KBs", a line of research that investigates the potential of LMs to act as KBs for retrieving knowledge. To assess the capabilities of LMs in this role, they constructed a series of probing datasets (LAMA) derived from commonsense KGs and a set of prompt-based probes to examine LMs.

Hu et al. (2022) introduced Knowledgeable Prompt-tuning (KPT), an approach designed to refine the probing of LMs by integrating knowledge from KBs to construct an enhanced answer mapping. For instance, when addressing a cloze-style question like "Albert Einstein is a [MASK]", potential responses might include "theoretical physicist" to "physicist", or "scientist". KPT's idea is to expand the answer space, utilising the breadth of information available in KBs to allow for more accurate probing evaluation of LMs.

Instruction-tuning LLMs with KBs LUO et al. (2023) introduced RoG, a strategy designed to mitigate the issues of knowledge deficits and hallucinations in LLMs. The approach begins with instruction-tuning the LLM to generate *plans*, which are relevant paths that can be grounded using KGs. It then extracts relevant information from the KGs according to the plans to provide additional context for the LLM.

Caufield et al. (2024) introduced SPIRES for enriching existing KBs utilising LLMs. This method employs an iterative cycle where prompts are generated, completed, and then parsed into structured representations that are grounded to predefined schemas.

SPIRES has been made available as a tool in OntoGPT[12], and some of the functions have further been integrated as a plugin for ChatGPT.

Future Insights A persistent pursuit in the field of knowledge engineering involves constructing knowledge bases automatically from diverse unstructured sources. The advent of LLMs has made this goal more achievable. Nevertheless, challenges such as addressing the hallucination problem in LLMs are critical to ensure the generation of faithful and truthful text, which can subsequently be transformed into reliable knowledge. Promising future directions include employing the RAG paradigm, which strategically integrates encoder-based models for information retrieval with decoder-based language models for text generation. Additionally, the concept of knowledge editing aims to refine the outputs or adjust the weights of LLMs using semantic components. There is also a focus on enhancing the efficiency of LLMs through techniques such as model distillation and quantisation, alongside hardware optimisation, to make these models more accessible and efficient for researchers and users alike.

3.A Literature Summary

Table 3.2 in the next page provides a summary of the literature (excluding ours) discussed in Section 3.4. This table categorises works according to the following dimensions: model architectures, learning paradigms, key methodologies, types of KBs, and the specific tasks (if applicable) relevant to KB construction and curation.

[12] https://github.com/monarch-initiative/ontogpt

Work	Features
RDF2Vec (2016)	Word2Vec Pre-train Graph Walk KG Embedding
OWL2Vec* (2021a)	Word2Vec Pre-train Graph Walk Onto Embedding
DeepAlign (2018)	Word2Vec Fine-tune Contrastive Onto Alignment
ERINE (2019)	Transformer Encoder Pre-train Prompt KG
COMET (2019)	Transformer Decoder Pre-train Prompt KG Construction
K-BERT (2020b)	Transformer Encoder Pre-train Prompt KG
SapBERT (2021a)	Transformer Encoder Pre-train Contrastive Onto Embedding
Grapher (2021)	Transformer Encoder-Decoder Pre-train Prompt KG Construction
BLINK (2020)	Transformer Encoder Fine-tune Contrastive KG Entity Linking
BERT-INT (2021)	Transformer Encoder Fine-tune KG Alignment
Onto-Prompt (2022)	Transformer Encoder Fine-tune Prompt Onto Completion
LAMA (2019a)	Transformer Encoder Probing Prompt KG Completion QA
KPT (2022)	Transformer Encoder Probing Prompt KG Completion QA
RoG (2023)	Transformer Decoder LLM Fine-tune Prompt KG QA
SPIRES (2024)	Transformer Decoder LLM Fine-tune Prompt Onto Construction

Table 3.2: Summary of literature on language models for knowledge engineering.

PART II

Language Models for Ontology Engineering

Methodologies

CHAPTER 4

Language Models for Ontology Alignment

Contents

This chapter delves into our contributions to ontology alignment, including the BERTMap system (He et al., 2022a) and the Bio-ML evaluation resource (He et al., 2022b, 2023a). BERTMap is designed for concept equivalence matching and it is among the first that introduces transformer-based language models into the realm of ontology

alignment, while Bio-ML aims to benchmark both concept equivalence and subsumption matching with a comprehensive evaluation framework for both rule-based and machine learning-based systems.

4.1 Related Work

Ontology alignment, or ontology matching (OM), is the task of identifying semantically related entities across different ontologies, where a semantic relationship – typically equivalence or subsumption – between two matched entities is referred to as a mapping. In Section 2.4.2 of Chapter 2, we have covered a detailed discussion of relevant challenges in OM. This section explores the related work in OM. We will highlight key contributions in conventional, rule-based OM systems, and more recent, machine learning-based OM systems, as well as research efforts for benchmarking OM. We will also mention BERTMap along the discussion, to emphasise important comparisons.

4.1.1 Rule-based Systems

Classic rule-based OM systems mainly focus on concept equivalence matching, dissecting this into three core aspects: lexical matching, structure matching, and logical inference (Otero-Cerdeira et al., 2015). Prominent among these systems are LogMap (Jiménez-Ruiz and Cuenca Grau, 2011) and AML (Faria et al., 2013), both of which continue to set as the state-of-the-art in various OM evaluation tracks. LogMap harnesses a word-level lexical index to establish initial mappings as anchors, subsequently iterating between exploiting the ontology's structure for mapping extension and the logical inference for mapping repair. AML, on the other hand, integrates multiple strategies for computing lexical matches before proceeding to similar stages of mapping extension and repair.

Although these systems have shown significant effectiveness, their approach to lexical matching relies primarily on the surface text, overlooking the subtleties of language understanding. In contrast, BERTMap builds upon the foundations laid by these classic systems but goes beyond their capabilities by integrating LMs for enhanced lexical matching. This integration allows BERTMap to exploit textual semantics and context in mapping prediction, reducing the reliance on text pre-processing and dictionary

consultations as seen in LogMap, or the necessity of combining various string-matching heuristics as in AML. It is worth mentioning that AML has latter been succeeded by a new system named Matcha, which includes a deep learning-based variant known as Matcha-DL (Cotovio et al., 2024).

4.1.2 Machine Learning-based Systems

Opposed to rule-based systems are machine learning-based OM solutions, which, instead of relying on manually designed rules and heuristics, aim to learn patterns from ontologies to facilitate OM. They are classified into two main categories based on their reliance on annotated data. *(i)* Supervised approaches require a given set of annotated mappings for training, where the system learns to predict matches based on these examples. This methodology is powerful but hinges on the availability of high-quality, annotated training data, which is often not available in real-world OM. *(ii)* Unsupervised approaches do not require annotated mappings. Instead, they aim to capture patterns and relationships directly from the input ontologies, leveraging inherent structural and lexical information for mapping prediction. The unsupervised methods offer flexibility and applicability in most OM applications, but to make it effective, a sophisticated strategy is often required to effectively extract and utilise knowledge from the ontologies without explicit training signals.

Supervised OM Systems Notable supervised OM solutions include: *(i)* Nkisi-Orji et al. (2019) combined hand-crafted features like string similarities with Word2Vec embeddings, and then applied a random forest classifier; *(ii)* Wang et al. (2018) incorporated a customised neural network to learn contextual word features, combining with several hand-crafted features for prediction. *(iii)* Chen et al. (2021b) proposed LogMap-ML, a machine learning extension of LogMap, utilising path context and ontology-tailored word embeddings generated by OWL2Vec* (Chen et al., 2021a) to enhance its matching process. *(iv)* Iyer et al. (2021) introduced a "dual attention" mechanism in their model called VeeAlign, focusing on both path-level and node-level relevance to refine its concept embeddings.

A shared limitation of these works is their reliance on ad-hoc feature engineering, which can hinder the model's generalisability. Another challenge is the need of high-quality annotated mappings, which, as mentioned previously, are typically scarce and expensive to manually curate. Although strategies like distant supervision (Chen et al., 2021b) and sample transfer (Nkisi-Orji et al., 2019) have been explored to mitigate this issue, the variability in sample quality can still constrain their effectiveness.

Unsupervised OM Systems At the time BERTMap was introduced, unsupervised OM systems were relatively underexplored. Among these, DeepAlignment, as discussed in Section 3.4 of Chapter 3, stands out. It employs a synonym semantics refinement technique known as counter-fitting to derive concept representations without supervision. Another work is ERSOM (Xiang et al., 2015), which adopts an auto-encoder architecture for learning concept representations from textual descriptions, and an iterative similarity propagation method for discovering mappings. However, a shared limitation in both works is their lack of consideration for word context in their lexical matching.

An Early Attempt on LMs for OM Neutel and de Boer (2021) conducted an early exploration into utilising BERT for OM. Their study explored two approaches: *(i)* employing pre-trained BERT to encode concepts into token embeddings and subsequently measuring their cosine similarity (unsupervised); *(ii)* fine-tuning concept embeddings using the sentence transformer architecture (Reimers and Gurevych, 2019), which requires a substantial set of annotated mappings for effective training (supervised).

In our replication of approach *(i)*, we observed that its performance was no better than traditional string-matching techniques. Additionally, Neutel and de Boer (2021) reported that while method *(ii)* exhibited greater coverage, its mean reciprocal rank score was notably lower compared to the non-contextual word embedding model, FastText (Bojanowski et al., 2017). These outcomes of this early attempt do not deny the potential of adopting LMs for OM, but rather show that it is non-trivial to explore an effective strategy for this purpose.

Follow-up Works of BERTMap Since the introduction of BERTMap, there has been a growing interest in leveraging LMs for OM. A significant work from our team is BERTSubs (Chen et al., 2023), which extends BERTMap's approach to address *concept subsumption matching*. This enhancement involves fine-tuning a BERT classifier with various templates to predict whether one concept is a subsumer of another. Additionally, BERTSubs is equipped to handle complex concepts involving existential restrictions, and supports both intra-ontology and inter-ontology subsumption prediction.

Following the line of LMs for OM, other studies have explored different methodologies beyond the typical classification-based fine-tuning. For instance, SORBET (Gosselin and Zouaq, 2023) adopts the contrastive learning paradigm to train concept embeddings, which are then applied to OM. Another recent work, OLaLa (Hertling and Paulheim, 2023), employs the Retrieval-Augmented Generation (RAG) paradigm, utilising a sentence transformer for initial candidate retrieval, followed by re-ranking these candidates using large language models.

4.1.3 Evaluation Efforts

The Ontology Alignment Evaluation Initiative[1] (OAEI) organises an annual evaluation campaign to assess ontology matching (OM) systems. This campaign has gone through nearly two decades of evolution, featuring several tracks that cover different aspects of ontology matching, including TBox/Schema matching, instance matching, and tabular data matching. The main evaluation metrics used across these tracks are Precision, Recall, and F-score but each track can have a tailored evaluation setting.

Some other studies, particularly those involving machine learning-based OM approaches, have introduced non-standard metrics and have utilised datasets with incomplete reference mappings. For instance, Chen et al. (2021b) employed the LogMap-ML system to match the Food Ontology (FoodOn) (Dooley et al., 2018) with the Health and Lifestyle Ontology (HeLiS) (Dragoni et al., 2018). They assessed the performance using approximate Precision and Recall on partial reference mappings, supplemented by sampling and manual verification. Neutel and de Boer (2021) focused on coverage

[1] http://oaei.ontologymatching.org/

and ranking to evaluate OM on industrial data, incorporating human judgement to address the lack of complete reference standards. The diversity of evaluating machine learning-based OM solutions urges the need for a unified and comprehensive evaluation framework.

4.2 BERTMap System

4.2.1 Overview

We present BERTMap, a language model (BERT)-based pipeline system for *concept equivalence* OM, marking one of the initial endeavors to incorporate transformer-based language models (LMs) into the realm of OM. BERTMap leverages the fine-tuning capabilities of BERT for mapping prediction and the structural and logical information of ontologies for mapping refinement. It allows flexible configurations of unsupervised, semi-supervised, and data augmented modes, and has been proven effective in several OM benchmarks including OAEI LargeBio and Bio-ML.

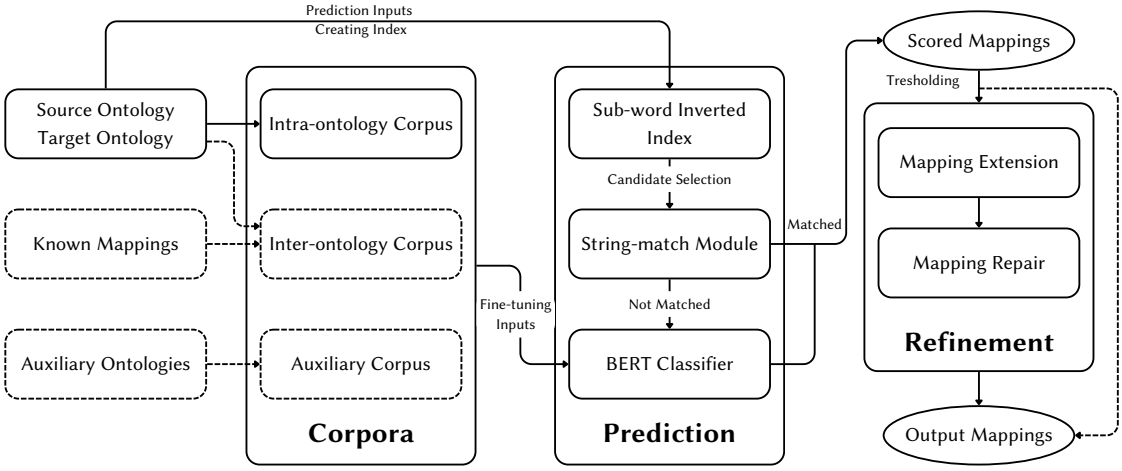


Figure 4.1: Illustration of the BERTMap system, where the dotted lines indicate optional paths and the dotted rectangles indicate optional modules.

Illustrated in Figure 4.1, the main steps of BERTMap are described as follows:

- **Corpora Construction:** This initial step involves the extraction of synonym and non-synonym pairs from input source and target ontologies, and other optional

sources including given mappings and/or auxiliary ontologies.

- **Fine-Tuning:** An appropriate pre-trained BERT model is selected and fine-tuned on the corpora constructed in the previous step. This step adapts BERT's language understanding to the context of ontologies.

- **Mapping Prediction:** In this phase, potential mapping candidates are first selected using a sub-word inverted index, after which the string-matching module and the fine-tuned BERT classifier are used to assign a score to each candidate. The scored mappings can then be filtered using a threshold.

- **Mapping Refinement:** The final step focuses on refining the scored mappings. It improves the recall by extending additional mappings from neighbouring concepts of highly scored mappings and improves the precision by removing mappings that could lead to logical inconsistencies.

In the next few sections, we will delve into detail of each step outlined above.

4.2.2 Text Semantics Corpora

In real-world ontologies, especially domain-specific ones, it is common for a named concept to be associated with multiple labels, which are specified using annotation properties like `rdfs:label`. These labels serve as the concept's *aliases*. For clarity, we refer to a label that has undergone preprocessing (which includes lowercasing and removing underscore symbols) as s, and the complete set of a concept c's pre-processed labels as $S(c)$.

Labels that belong to the same concept, or to semantically equivalent concepts, are naturally considered synonymous within the context of input ontologies. Conversely, labels from semantically distinct concepts are treated as non-synonymous. We construct text semantics corpora comprising these synonyms and non-synonyms for BERT fine-tuning. According to the source, the corpora are divided into three categories[2] as follows:

[2]The inter-ontology corpus and auxiliary corpus are referred to as cross-ontology corpus and complementary corpus in the original paper (He et al., 2022a).

Intra-ontology Corpus Within each input ontology, we examine every named concept c to identify all synonyms (s_1, s_2), where $s_1, s_2 \in S(c)$. These include *identity synonyms*, where $s_1 = s_2$. For non-synonyms, we categorise them into two types: *(i) soft non-synonyms*, which are labels from two arbitrarily selected, unrelated concepts, and *(ii) hard non-synonyms*, which are labels from concepts that are logically disjoint.

Recognising that explicit concept disjointness is often unspecified in ontologies, we adopt a heuristic approach by presuming that sibling concepts – those sharing a common parent – are disjoint. This assumption serves as a pragmatic, albeit simplistic, method to deduce disjointness based on the ontology's concept hierarchy.

Inter-ontology Corpus The scarcity of annotated mappings in OM poses challenges for applying traditional supervised learning techniques. To mitigate this, we employ a semi-supervised approach, leveraging a limited set of expert-annotated mappings to extract synonymous and non-synonymous label pairs across ontologies.

Given a pair of matched concepts c and c', we extract all possible synonyms (s, s'), where each pair is derived from the Cartesian product of their label sets, $S(c) \times S(c')$. For non-synonymous pairs, we rely on randomly aligned concept pairs, assuming they are likely to be non-synonymous due to the absence of explicit mapping. However, identifying hard non-synonyms in this context is more complex since determining logical disjointness between aligned concepts from different ontologies is not straightforward without extensive domain knowledge or additional annotation. Consequently, our cross-ontology corpus primarily focuses on generating synonyms from known mappings and non-synonyms from random alignments, providing an option for semi-supervised learning.

Auxiliary Corpus When facing a lack[3] of labels in the input ontologies, we can enhance our corpora by incorporating external sources, specifically auxiliary ontologies, for data augmentation. This additional corpus is constructed following the same methodology as the intra-ontology corpus.

To maintain data quality and manage the corpus size effectively, we select auxiliary ontologies that align with the domain of the input ontologies. Moreover, we focus on

[3]For example, top-level ontologies like Schema.org do not usually define multiple labels.

named concepts within these auxiliary ontologies that have overlapping labels with the concepts in the input ontologies. This selective approach helps to minimise data noise and ensures that the augmented data is likely to be beneficial for the fine-tuning process.

Further Processing For each of the constructed corpora, we enhance the learning of symmetry in synonym relationships by appending reversed pairs of synonyms, i.e., if (s_1, s_2) is a synonym specified in a corpus, (s_2, s_1) is also included. Furthermore, to prevent contradictory training signals, we remove any non-synonymous pairs that conflict with the established synonym pairs in a corpus.

Notations In the following, we will refer to the input source ontology as $\mathcal{O}_{\mathsf{src}}$ and the input target ontology as $\mathcal{O}_{\mathsf{tgt}}$. It is important to note that referring to a concept c from an ontology $\mathcal{O}$ using the notation $c \in \mathcal{O}$ is technically imprecise, as ontologies are not merely sets of concepts. Therefore, we will stick a phrase like "a concept c from/in an ontology $\mathcal{O}$".

4.2.3 BERT Fine-tuning

Having constructed the text semantics corpora, we proceed to fine-tune a pre-trained BERT model attached with a downstream linear classification layer, using cross-entropy loss for the task of synonym classification. This process aligns with the standard fine-tuning practice, detailed in Definition 3.3.3 of Chapter 3. We opt to not conduct pre-training but instead utilise an existing model from the `HuggingFace Hub`[4], selecting a pre-trained checkpoint that is most relevant to the domain of the input ontologies.

For the fine-tuning process, the input to BERT consists of tokenised label pairs wrapped into a template as follows:

$$[\mathsf{CLS}]\ T(s_1)\ [\mathsf{SEP}]\ T(s_2)\ [\mathsf{SEP}]$$

Here, $T(\cdot)$ represents the sub-word tokenisation function native to the pre-trained BERT, with each input sequence capped at a maximum length of 128 tokens.

The classification layer, equipped with a dropout mechanism, leverages the embedding of the $[\mathsf{CLS}]$ token derived from BERT's final layer. This embedding is transformed

[4] `https://huggingface.co/models`

by the linear layer into a 2-dimensional vector, which then passes through a softmax function to produce the final output. The model's optimisation is conducted using the AdamW algorithm (Loshchilov and Hutter, 2018). The output from the model is a pair $\langle 1 - s, s \rangle$, where $s \in [0, 1]$ represents the probability that the given label pair is synonymous.

4.2.4 Mapping Prediction

To identify matched concepts in $\mathcal{O}_{\mathsf{tgt}}$ for a specific concept c from $\mathcal{O}_{\mathsf{src}}$, a naive approach would be to iterate over the entire set of concepts in $\mathcal{O}_{\mathsf{tgt}}$. However, this method has a time complexity of $O(n^2)$, rendering it impractical for large ontologies due to significant computational demands.

To streamline the search process, BERTMap implements an efficient strategy that narrows down the search space. Initially, it employs a sub-word inverted index to select a subset of potential candidate concepts from $\mathcal{O}_{\mathsf{tgt}}$ that are likely matches for c. Following this raw selection, each candidate mapping undergoes evaluation using a string-matching module in conjunction with the fine-tuned BERT model described in the previous section.

Candidate Selection The underlying assumption guiding the candidate selection in BERTMap is that *matched concepts are likely to share labels with overlapping sub-tokens.* Traditional approaches in previous studies often utilise word-level inverted indices, complemented by additional text processing techniques like stemming and consulting dictionaries (Jiménez-Ruiz and Cuenca Grau, 2011; Wang et al., 2018).

In contrast, BERTMap leverages a sub-word inverted index, offering two significant advantages: *(i)* it naturally accommodates various word forms without the need for supplementary processing, and *(ii)* it decomposes unknown words into a sequence of known sub-words, rather than reducing them to a single "unknown" token. This method ensures that subtle linguistic variations are effectively captured and thus facilitating **high recall** for potential concept matches.

We construct sub-word inverted indices utilising BERT's inherent WordPiece tokeniser (Wu et al., 2016), which is developed through an iterative process that incrementally merges individual characters (from the corpus) into the most probable sub-words at each step. Rather than re-training this tokeniser on our specific corpora, we choose to employ the pre-existing tokeniser provided with BERT. This decision is grounded in the fact that the built-in tokeniser has been trained on a vast and diverse corpus, comprising approximately 3.3 billion words and encompassing a wide range of topics (Kenton and Toutanova, 2019), and in this context we consider generality to be preferable to task specificity.

Without loss of generality, we regard the source ontology O_{src} as the anchor for our alignment search. We construct[5] a sub-word inverted index I_{tgt} for the target ontology O_{tgt}. In this index, each key represents a sub-word, and its associated values are the concepts containing at least one label with this sub-word after tokenisation. Retrieving target concepts linked to a token t is denoted by $I_{\mathsf{tgt}}[t]$.

The candidate selection process is presented as follows: for each concept c in the source ontology, we select target concepts that share at least one sub-word token with c, represented by $\bigcup_{t \in T(c)} I_{\mathsf{tgt}}[t]$. These candidates are then ranked using a scoring metric based on the *inverted document*[6] *frequency* (idf):

$$f_{\mathsf{select}}(c, c') = \sum_{t \in T(c) \cap T(c')} \mathsf{idf}(t) = \sum_{t \in T(c) \cap T(c')} \log_{10} \frac{|C'|}{|I_{\mathsf{tgt}}[t]|}$$

where c' is a target concept and $|\cdot|$ denotes set cardinality. $T(\cdot)$ still refers to the sub-word tokenisation function as previously mentioned, but when applied to a concept, it returns the set of sub-word tokens derived from all its labels.

Subsequently, we select the top k scoring target concepts for c, establishing potential mappings for further scoring. This strategy significantly reduces the computational complexity from $O(n^2)$ to $O(kn)$, where $k \ll n$ is the cut-off for candidate selection.

It is noteworthy that, given the design of this candidate selection process, efficiency can be further enhanced by selecting the smaller-sized input ontology as the anchor.

[5] Index construction is linear w.r.t. the number of sub-words.

[6] A "document" in this context refers to an ontology concept.

This is because querying a sub-word inverted index is an operation with constant time complexity. Hence, anchoring the process on the smaller ontology minimises the number of queries to be made.

Mapping Scoring BERTMap employs a combination of string matching and the fine-tuned BERT classifier (f_{bert}) to compute the mapping score between two concepts c and c'. The scoring function f_{align} is defined as follows:

$$f_{\text{align}}(c, c') = \begin{cases} 1.0 & \text{if } S(c) \bigcap S(c') \neq \emptyset \\ f_{\text{bert}}(S(c), S(c')) & \text{otherwise} \end{cases}$$

where the condition $S(c) \bigcap S(c') \neq \emptyset$ indicates that there is at least one label in c that exactly matches a label in c'. The function $f_{\text{bert}}(\cdot, \cdot)$ calculates the average synonym score for all label pairs $(s, s') \in S(c) \times S(c')$, as determined by the BERT classifier.

This scoring approach is designed to optimise computational efficiency: if there is an exact label match between c and c' (a straightforward case), the mapping is immediately scored as 1.0, bypassing the need for the more resource-intensive BERT classifier. This string-matching step effectively filters out "easy" mappings, reserving the BERT classifier for cases where nuanced semantic analysis is necessary.

Mapping Filtering In its initial implementation, as detailed in (He et al., 2022a), BERTMap adopted a one-to-one mapping strategy, selecting only the highest-scoring candidate for each concept c from the source ontology. Additionally, it employed a score-based filtering mechanism, wherein mappings were retained or discarded based on a threshold, a hyperparameter that should be determined on a validation set. However, subsequent integration of BERTMap into the `DeepOnto` package marked a shift from this rigid one-to-one mapping assumption, allowing for multiple potential mappings for a single concept.

4.2.5 Mapping Refinement

With the predicted mappings and their scores, BERTMap further extends and repairs the mappings by utilising the graphical and logical information of input ontologies.

Mapping Extension Our mapping extension algorithm aims to mitigate the potential reduction in recall caused by the candidate selection criteria, which only considers concepts with overlapping sub-tokens as potential matches. This algorithm is grounded in the *locality principle* (Grau et al., 2007; Jiménez-Ruiz et al., 2020), defined as follows:

Definition 4.2.1 (Locality Principle)

If two concepts c and c' from respective ontologies are aligned, it is probable that their semantically related concepts (e.g., parent and child concepts) within each ontology are also aligned.

This principle leverages the concept hierarchies of ontologies, asserting that mappings are not isolated incidents but part of a broader, coherent alignment across the ontological structures. BERTMap utilises this principle to discover new mappings with an *iterative mapping extension* algorithm as follows:

Algorithm 1 Iterative Mapping Extension

Input: Prediction mapping set, $\mathcal{M}_{\text{pred}}$
Parameter: Extension threshold, κ
Output: Extended prediction mapping set, $\mathcal{M}_{\text{ex}}$

Initialise the frontier: $\mathcal{M}_{\text{fr}} \leftarrow \mathcal{M}_{\text{pred}}$
Initialise the extended prediction mapping set: $\mathcal{M}_{\text{ex}} \leftarrow \mathcal{M}_{\text{pred}}$
while $\mathcal{M}_{\text{fr}}$ is not empty **do**
    Start with an empty set for new extensions: $\mathcal{M}_{\text{new}} \leftarrow \{\}$
    for each mapping $(c, c', f_{\text{align}}(c, c')) \in \mathcal{M}_{\text{fr}}$ **do**
        for $(e, e') \in (\text{ParentOf}(c) \times \text{ParentOf}(c')) \cup (\text{ChildOf}(c) \times \text{ChildOf}(c'))$ **do**
            $m \leftarrow (e, e', f_{\text{align}}(e, e'))$
            if $f_{\text{align}}(e, e') \geq \kappa$ **and** $m \notin \mathcal{M}_{\text{ex}}$ **then**
                $\mathcal{M}_{\text{new}} \leftarrow \mathcal{M}_{\text{new}} \cup \{m\}$
            end if
        end for
    end for
    Expand the extended set with new mappings: $\mathcal{M}_{\text{ex}} \leftarrow \mathcal{M}_{\text{ex}} \cup \mathcal{M}_{\text{new}}$
    Prepare for the next iteration: $\mathcal{M}_{\text{fr}} \leftarrow \mathcal{M}_{\text{new}}$
end while
return $\mathcal{M}_{\text{ex}}$

This algorithm selectively retains new mappings that have not been previously included in $\mathcal{M}_{\text{ex}}$ and exhibit scores greater than or equal to κ (as specified in Line 8), where κ represents the extension threshold. While κ serves as a hyperparameter, empirical observations indicate that the algorithm's outcomes are insensitive to variations of κ. Consequently, we adopt a fixed value of $\kappa = 0.9$. The algorithm terminates when it no longer identifies new eligible mappings.

Mapping Repair Lastly, BERTMap incorporates a crucial refinement step known as *mapping repair*, which utilises the logical information of the input ontologies to remove any predicted mappings that could introduce inconsistencies upon the integration of the two ontologies. We follow the definition from Jiménez-Ruiz et al. (2013):

Definition 4.2.2 (Mapping Repair)

Given two input ontologies, $\mathcal{O}_{\text{src}}$ and $\mathcal{O}_{\text{tgt}}$, and a set of predicted mappings between them, $\mathcal{M}_{\text{pred}}$, each ontology can be viewed as a set of logical axioms. The predicted mappings in $\mathcal{M}_{\text{pred}}$ can be converted into axioms that define the logical relationships between the aligned concepts. If the combined set of axioms $(\mathcal{O}_{\text{src}} \cup \mathcal{O}_{\text{tgt}} \cup \mathcal{M}_{\text{pred}})$ results in a logical inconsistency, the mapping repair process involves removing the *minimum number of mappings*[a] from $\mathcal{M}_{\text{pred}}$ to restore consistency. The resulting set, $\mathcal{O}_{\text{src}} \cup \mathcal{O}_{\text{tgt}} \cup (\mathcal{M}_{\text{pred}} - \mathcal{M}_{\text{repair}})$, is then consistent.

[a] This is provided that the predicted mappings are already highly scored. Otherwise, it is more reasonable to remove a set of minimum scored mappings.

However, computing a "perfect repair" (also known as a *diagnosis*) can be expensive and might not yield a unique solution. To address this, Jiménez-Ruiz et al. (2013) introduced a repair methodology based on propositional logic that facilitates the computation of an *approximate repair*. This approach ensures that: *(i)* the approximate repair is a subset of the diagnosis (thus there is no sacrifice of correct mappings); and *(ii)* the number of unsatisfiable concepts after applying this repair is minimised.

While mapping repair is a commonly seen in traditional OM systems, it is less frequently encountered in machine learning-based ones. BERTMap integrates this

essential step into its workflow by utilising the repair algorithm proposed by Jiménez-Ruiz et al. (2013).

4.3 Bio-ML Resource

4.3.1 Overview

During the time when BERTMap was initially introduced, we observed that existing OM benchmarks predominantly favoured traditional rule-based OM systems and were not optimally structured to accommodate machine learning (ML)-based methodologies. Motivated by this observation, we developed a new OM resource named Bio-ML. This resource focuses on "biomedical ontologies", providing five ontology pairs for both concept equivalence and subsumption matching. It is also characterised as "ML-friendly", incorporating a comprehensive evaluation framework to support fair comparisons for both rule-based and ML-based OM systems. As a new track that was first integrated into the Ontology Alignment Evaluation Initiative[7] (OAEI) in 2022, Bio-ML aims to tackle the following limitations of existing benchmarks:

- **Limited Evaluation Metrics:** The standard evaluation metrics like Precision, Recall, and F-score, often fall short in providing a comprehensive evaluation when dealing with incomplete reference mappings. They may inadvertently penalise systems that find mappings that are correct but absent in the reference set. Alternative metrics, such as approximate Precision and Recall derived from sampling and human verification (Chen et al., 2021b), or Accuracy by differentiating one positive mapping from a set of loosely constructed negative mappings (Nguyen et al., 2021) consensus-based approaches from multiple systems (Harrow et al., 2017), have been proposed. However, these methods are often ad-hoc and not universally applicable.

- **Suboptimal Reference Mappings:** The reference mappings in many OM datasets often suffer from incompleteness or inaccuracies. These mappings are sometimes referred to as *silver standards* to differentiate them from (supposedly) gold standard

[7] http://oaei.ontologymatching.org/

mappings. The reliance on silver standards can result in skewed comparisons between different OM systems. For instance, DeepAlignment (Kolyvakis et al., 2018) showed superior performance over LogMap and AML in evaluations using silver standard mappings between the Schema.org and DBPedia ontologies. Notably, AML recorded zero Recall in this context; however, a detailed review of the results indicated that AML did identify some plausible mappings not present in the reference set. In the context of the OAEI LargeBio track, certain reference mappings are tagged as "ignored" following an algorithmic repair of logical inconsistencies caused by ontology integration (Pesquita et al., 2013). However, some of the discarded mappings may still be considered correct by domain experts.

- **Ignoring Subsumption Mappings:** The majority of existing resources in OM focus on equivalence matching. However, real-world ontologies often contain a greater number of subsumption mappings compared to equivalence mappings. These subsumption relationships are pivotal for effective knowledge integration and ontology curation. The limited emphasis on subsumption matching in traditional rule-based OM systems can also be attributed to their limited predictive capabilities in identifying such relationships. Subsumption is challenging to discern solely through surface-form string-matching techniques employed by these conventional systems.

- **Inadequate Support for ML-based Systems:** Most existing OAEI tracks do not support evaluation for ML-based systems. A notable limitation is the lack of proper data splitting protocols, which are essential for training and evaluating supervised or semi-supervised ML models. Additionally, traditional evaluation metrics like Precision, Recall, and F-score may not always be relevant for ML-based systems, particularly when these systems are modular rather than end-to-end, making the computation of a full alignment both time-consuming and not entirely indicative of the performance of individual ML components. In the context of ML development, ranking-based metrics, which are prevalent in tasks like knowledge graph (KG) completion, offer a more fitting evaluation framework yet are seldom

adopted in the OM community. This gap results in non-standardised and less convenient evaluation processes for ML-based OM systems, highlighting a need for a shift towards more ML-friendly evaluation practices.

The outlined limitations represent persistent challenges in the evaluation of OM systems. In an effort to tackle these issues, or at least mitigate them, we introduce Bio-ML, an extensive OM resource derived from Mondo (Shefchek et al., 2020) and UMLS (Bodenreider, 2004). Accompanying this resource is a comprehensive evaluation framework designed to be compatible with both ML-based and traditional OM systems.

4.3.2 Resource Construction

This section outlines the methodologies employed in the development of the Bio-ML resource.

Ontology Preprocessing We undertake two primary preprocessing steps for each ontology in the Bio-ML resource: *(i)* We remove obsolete or deprecated concepts. These are typically concepts with updated alternatives or those no longer in active use, ensuring the ontology reflects current knowledge. *(ii)* We remove annotation properties that link to external data sources (e.g., obo:hasDbXref). Such annotations could inadvertently provide clues about correct alignments, which might bias the evaluation of OM systems. Contrary to the approach in the OAEI LargeBio track,[8] where some annotations are merged into `rdfs:label`, we retain other annotation properties to allow OM systems to interpret them independently.

Ontology Pruning To facilitate manageable and focused evaluation, particularly in contexts with limited reference mappings, we implement *ontology pruning*. This process involves extracting relevant sub-ontologies while preserving the hierarchical relationships inherent to the original ontology. The pruning process (see Figure 4.2) is as follows: if a concept c meets predefined relevance criteria (e.g., association with reference mappings), it is retained. Otherwise, c is removed, along with all related axioms. To

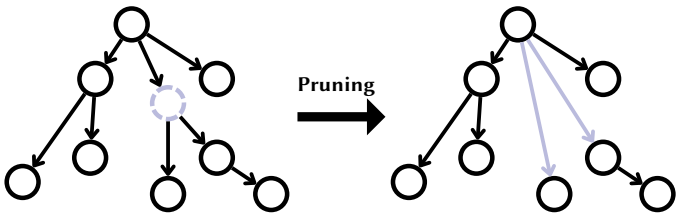


Figure 4.2: Illustration of ontology pruning, where the coloured circle on the left represents a concept targeted for removal and the coloured arrows on the right represent re-asserted subsumption axioms.

maintain the concept hierarchy, the children of c are then asserted directly under each parent of c.

Such pruning not only yields ontologies of a practical scale for OM tasks but also enhances the relevance and coverage of reference mappings w.r.t. to the pruned ontologies.

Subsumption Mapping Construction We construct subsumption reference mappings based on the existing equivalence reference mappings. For every equivalence mapping $(c, \equiv, c')$, a corresponding subsumption mapping $(c, \sqsubseteq, c'')$ is established, where c'' is asserted as a subsumer of c' in c's ontology. For instance, given `DOID:4321` ("Large Cell Acanthoma") equivalently matched to `NCIT:C27518`, and `DOID:174` ("Acanthoma") as its parent concept, a plausible subsumption mapping is (`NCIT:C27518`, $\sqsubseteq$, `DOID:174`).

Acknowledging that c and c' may have numerous asserted and inferred subsumers, incorporating all possible subsumers would generate an overwhelming number of subsumption mappings for each equivalence pair. To manage this, we choose one of the most specific subsumers of c', introducing a set of selective yet incomplete subsumption mappings. Consequently, during subsumption matching evaluation, we omit Recall as a metric (detailed in Section 4.3.4).

To assess a system's proficiency in deducing inter-ontology subsumptions directly, we exclude the original equivalence mapping (c, c') by eliminating c'. Similar to ontology pruning, we ensure the preservation of the hierarchical structure by asserting the parent concepts of c' as subsumers of its child concepts after its deletion. If the deleted concept is involved in other equivalence or pre-constructed subsumption mappings, we either

[8]LargeBio has been superceded by Bio-ML.

bypass those equivalence mappings or discard the affected subsumption mappings. This approach, termed *online subsumption mapping construction*, enables a dynamic generation of subsumption mappings.

Negative Candidate Generation To facilitate ranking-based evaluation, we introduce a *negative candidate generation* algorithm to create representative negative samples for each reference mapping $m = (c, r, c')$, where r is either $\equiv$ or $\sqsubseteq$. We implement three different strategies to generate these negative candidates:

1. **Text-similarity Search:** This method aims to generate hard negative candidates that are textually similar to the target concept, thereby introducing ambiguity. We leverage the candidate selection algorithm from BERTMap (detailed in Section 4.2.4), selecting the top-N textually similar concepts within the same ontology as c'. Unlike in BERTMap, where candidates are chosen from the opposing ontology, here the selection is conducted within the same ontology as the ground truth concept.

2. **Neighbourhood Search:** This approach generates hard negative candidates that are hierarchically proximate to the target concept. Utilising the asserted subsumption axioms, we construct a graph where nodes represent named concepts and edges represent subsumption relations (i.e., a concept hierarchy as in Definition 2.4.7 of Chapter 2). We then apply a breadth-first search (BFS) across the subsumption edges (in both directions) to identify and include neighbouring concepts of c' as candidates. The search progresses from one-hop neighbours to two-hop, and so forth, halting when the candidate count surpasses N or upon reaching a predetermined maximum hop limit. If more than N candidates emerge at the same level, we randomly sample from these to obtain precisely N candidates. Importantly, the root concept `owl:Thing` is excluded from this search, ensuring high-quality negative candidates are chosen within c's specific branch, thus enhancing search relevance and efficiency.

3. **Random Selection:** This method complements the previous strategies by randomly choosing negative candidates from the entire concept set.

To consistently obtain the required number of unique negative candidates, we employ an algorithm (in Algorithm 2) that integrates the three aforementioned strategies.

Algorithm 2 Negative Candidate Generation

Input: A reference mapping, $m = (c, r, c')$; Generation strategies $\{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_n\}$, and their corresponding numbers of negative candidates to generate $\{N_1, N_2, ..., N_n\}$
Output: A set of negative candidates for m, $\mathcal{N}(m)$

1: Determine invalid negative candidates for m as $\mathcal{I}(m)$.
2: Initialise the set of negative candidates: $\mathcal{N}(m) \leftarrow \{\}$
3: **for** $i \leftarrow 1$ to n **do**
4: Generate unique $|\mathcal{N}(m)| + |\mathcal{I}(m)| + N_i$ raw samples with strategy $\mathcal{S}_i$ as $\mathcal{N}_i(m)$
5: Exclude previously selected and invalid samples: $\mathcal{N}_i(m) \leftarrow \mathcal{N}_i(m) \setminus (\mathcal{N}(m) \cup \mathcal{I}(m))$
6: Limit $\mathcal{N}_i(m)$ to the top N_i samples if $|\mathcal{N}_i(m)| > N_i$
7: **while** $|\mathcal{N}_i(m)| < N_i$ **do**
8: Randomly select $N_i - |\mathcal{N}_i(m)|$ unique candidates as $\mathcal{R}$
9: $\mathcal{N}_i(m) \leftarrow (\mathcal{N}_i(m) \cup \mathcal{R}) \setminus (\mathcal{N}(m) \cup \mathcal{I}(m))$
10: **end while**
11: $\mathcal{N}(m) \leftarrow \mathcal{N}(m) \cup \mathcal{N}_i(m)$
12: **end for**
13: **return** $\mathcal{N}(m)$

Key considerations for this algorithm include:

1. **Filtering Positive Candidates:** The strategies employed might sometimes generate positive candidates, i.e., concepts that could be correctly matched with c through the semantic relationship r. These potential matches are computed (in Line 1, marked as $\mathcal{I}(m)$) and subsequently excluded from the negative candidate set. In the context of subsumption matching, $\mathcal{I}(m)$ is expanded to include both asserted and inferred super-classes of c', as pairings of c with these concepts do not constitute valid negative subsumption mappings. It is important to note that this process does not mean we do not consider one-to-many mappings, but instead, we consider each of them as a separate ground truth among a different set of sampled negative candidates.

2. **Ensuring Sufficient Generation:** During the i^{th} iteration, if strategy $\mathcal{S}_i$ fails to yield N_i unique new candidates, where N_i is significantly smaller than the total number

of possible candidates $|C'|$, the random sampling is invoked to supplement the candidate set. However, before adopting this compensation mechanism, we need to ensure that $\mathcal{S}_i$ is fully utilised. Therefore, the algorithm generates a larger pool of $|\mathcal{N}(m)| + |\mathcal{I}(m)| + N_i$ raw candidates (Line 3) as a buffer to accommodate the potential removal of duplicated (already in $\mathcal{N}(m)$) or invalid (in $\mathcal{I}(m)$) candidates.

Locality Module Enrichment In the 2023 version of Bio-ML, we enhanced the resource by integrating the locality module technique (Jiménez-Ruiz and Cuenca Grau, 2011), designed to augment pruned ontologies with logical modules, which provide additional contextual information for the existing concepts.

To maintain the integrity of reference mappings while introducing new concepts via locality modules, we mark these added concepts with a custom annotation property `use-in-alignment` set to false. This annotation explicitly indicates that these concepts, while available for providing context and supporting the OM process, should not be directly involved in the output alignment. In our updated evaluation protocol, we ensure that these supplementary concepts, even if included in the system's output mappings, are not considered in the evaluation metric calculations.

4.3.3 Datasets

Utilising the resource construction methodologies outlined in the previous section, we developed five OM tasks derived from Mondo and UMLS. The information about the source ontologies is presented in Table 4.1.

Mapping Source	Ontology	Ontology Source & Version	#Concepts
Mondo	OMIM	Mondo	44,729
	ORDO	BioPortal, `V3.2`	14,886
	NCIT	BioPortal, `V18.05d`	140,144
	DOID	BioPortal, `2017-11-28`	12,498
UMLS	SNOMED	UMLS, `US.2021.09.01`	358,222
	FMA	BioPortal, `V4.14.0`	104,523
	NCIT	BioPortal, `V21.02d`	163,842

Table 4.1: Information of the source ontologies used for creating the Bio-ML resource.

Mondo Datasets Mondo is an integrated disease ontology, with its concepts cross referenced to various source ontologies (Shefchek et al., 2020). During Mondo's construction, curators first gathered reference mappings from diverse sources, including UMLS, MeSH (Medical Subject Headings), and ICD (International Classification of Diseases). However, these initial mappings are considered semantically loose due to their inability to guarantee logical coherence when merged. To address this, the k-BOOM ontology construction tool, which employs logical reasoning and Bayesian inference, was utilised for the merging process (Mungall et al., 2016), followed by domain expert verification. The resulting ontology offers a comprehensive terminology for rare diseases (Haendel et al., 2020).

Guided by recommendations from the Mondo team, we focused on two ontology pairs, OMIM-ORDO and NCIT-DOID, noted for their relative recentness in Mondo at the time of Bio-ML's construction. Basic information of these source ontologies is presented as follows:

- **OMIM** (Online Mendelian Inheritance in Man) (Hamosh et al., 2005), a key online repository for genes and genetic phenotypes, predominantly exists in text form without an official ontology. To address this, we employed a data pipeline tool provided by Mondo[9] to construct a Mondo version of OMIM ontology. Notably, the maximum concept depth of the OMIM ontology is 2, making it a typical example of a "flat" ontology. Such ontologies have limited structural information, thus posing challenges to OM systems.

- **ORDO** (Orphanet Rare Disease Ontology) (Vasant et al., 2014) categorises rare diseases and their associations with genes and epidemiological aspects, drawing from the Orphanet database, which is populated by literature curation and validated by international experts. Given the genetic focus of many rare diseases, ORDO substantially intersects with OMIM, and the overlap has been integrated into Mondo.

[9] https://github.com/monarch-initiative/omim

- **NCIT** (National Cancer Institute Thesaurus) (Sioutos et al., 2007) spans various cancer-related concepts, including diseases, findings, and drugs, thus having a relatively smaller overlap with Mondo.

- **DOID** (Human Disease Ontology) (Schriml et al., 2012) maintains a comprehensive catalog of human diseases, with a significant portion integrated into Mondo. The alignment between NCIT and DOID primarily focus on cancer-related diseases.

The versions of the selected ORDO, NCIT, and DOID ontologies (see Table 4.1) were the closest to the most recent update of the Mondo mappings at that time, according to Mondo's documentation.

To derive relevant subsets from each source ontology w.r.t. Mondo, we utilised the ontology preprocessing and pruning methods described in the previous section, preserving only concepts that are cross-referenced to Mondo via the `skos:exactMatch` property. Subsequently, we extracted equivalence mappings based on these cross-references. For instance, `NCIT:C2751812` and `DOID:4321` are identified as equivalent because they both correspond to the same Mondo concept, `MONDO:0002961` ("Large Cell Acanthoma"). Building upon these established equivalence mappings, we generated subsumption mappings, a process that has been elaborated in the previous section.

UMLS Datasets UMLS (Unified Medical Language System) (Bodenreider, 2004) stands as a pivotal terminology integration effort in the biomedical domain, incorporating over 200 vocabularies to forge a comprehensive meta-thesaurus. This integrated ontology includes notable terminologies such as SNOMED CT (Stearns et al., 2001), FMA (Rosse and Mejino Jr, 2008), NCIT (Sioutos et al., 2007), and GO (Gene Ontology) (Ashburner et al., 2000), collectively describing millions of biomedical concepts and the relationships among them. Each concept in UMLS is categorised under one or more *semantic types*, such as "Finding", "Chemicals", and "Substance".

For the creation of OM datasets derived from UMLS, we leveraged its most recent version at the time of our experiments, `2021AB`. We selected three corresponding ontologies, SNOMED CT, FMA, and NCIT, all of which are extensive in scope, each

encompassing over 100K named concepts (see Table 4.1). Basic information of these source ontologies is presented as follows:

- **SNOMED** (SNOMED Clinical Terms) (Stearns et al., 2001) offers a broad coverage of clinical terms, enhancing the interoperability of electronic healthcare systems and clinical documentation. The SNOMED ontology was constructed using the official `snomed-owl-toolkit`[10].
- **FMA** (Foundational Model of Anatomy) (Rosse and Mejino Jr, 2008) provides a comprehensive vocabulary that represents a wide arrary of anatomical entities and relationships within the human body.
- **NCIT** has been introduced in the section of Mondo datasets.

We applied the same ontology preprocessing procedure to each ontology, and then created category-specific alignment tasks by pruning the original ontologies based on semantic types, meaning we retained only those concepts classified under a selected semantic type. We then extracted equivalence reference mappings from concepts across different ontologies that are matched to the same UMLS concept (Jiménez-Ruiz et al., 2011). Building upon these equivalence mappings, we generated subsumption reference mappings using the same methodology employed for the Mondo datasets.

Dataset Statistics We present the dataset statistics for the Bio-ML resource in Table 4.2 (OAEI 2022 Version) and Table 4.3 (OAEI 2023 Version). Here, $|\cdot|_{\mathcal{C}}$ denotes the cardinality of the set of named concepts, $\mathcal{C}$, in an ontology's signature. The tables show a notable decrease in the number of concepts for both source ($|\mathcal{O}_{\mathsf{src}}|_{\mathcal{C}}$) and target ($|\mathcal{O}_{\mathsf{tgt}}|_{\mathcal{C}}$) ontologies compared to the original ontologies in Table 4.1, resulting from the ontology pruning process. Furthermore, in the context of subsumption matching, the target ontology's concept count ($|\mathcal{O}_{\mathsf{tgt}}|_{\mathcal{C},\sqsubseteq}$) is further reduced due to the exclusion of certain target concepts during the generation of subsumption mappings.

In the 2023 edition, we incorporated locality-based enrichment, leading to a noticeable increase in the concept count. These additional concepts provide contextual

[10] `https://github.com/IHTSDO/snomed-owl-toolkit`

Task	Category	$\lvert\mathcal{O}_{\text{src}}\rvert_{\mathcal{C}}$	$\lvert\mathcal{O}_{\text{tgt}}\rvert_{\mathcal{C}}$	$\lvert\mathcal{O}_{\text{tgt}}\rvert_{\mathcal{C},\sqsubseteq}$	$\lvert\mathcal{M}_{\text{ref},\equiv}\rvert$	$\lvert\mathcal{M}_{\text{ref},\sqsubseteq}\rvert$
OMIM-ORDO	Disease	9,642	8,838	8,735	3,721	103
NCIT-DOID	Disease	6,835	8,448	5,113	4,686	3,339
SNOMED-FMA	Body	24,182	64,726	59,567	7,256	5,506
SNOMED-NCIT	Pharm	16,045	15,250	12,462	5,803	4,225
SNOMED-NCIT	Neoplas	11,271	13,956	13,790	3,804	213

Table 4.2: Dataset statistics of Bio-ML (OAEI 2022 Version).

Task	Category	$\lvert\mathcal{O}_{\text{src}}\rvert_{\mathcal{C}}$	$\lvert\mathcal{O}_{\text{tgt}}\rvert_{\mathcal{C}}$	$\lvert\mathcal{M}_{\text{ref},\equiv}\rvert$	$\lvert\mathcal{M}_{\text{ref},\sqsubseteq}\rvert$
OMIM-ORDO	Disease	9,648	9,275	3,721	103
NCIT-DOID	Disease	15,762	8,465	4,686	3,339
SNOMED-FMA	Body	34,418	88,955	7,256	5,506
SNOMED-NCIT	Pharm	29,500	22,136	5,803	4,225
SNOMED-NCIT	Neoplas	22,971	20,247	3,804	213

Table 4.3: Dataset statistics of Bio-ML (OAEI 2023 Version).

information but were excluded from the final evaluation to ensure they do not influence the outcome. Unlike the previous edition, we did not remove target concepts in the subsumption matching of the 2023 version because the effects of such deletions were offset by the newly added enrichment process.

4.3.4 Evaluation Framework

Our evaluation framework is designed to be comprehensive, incorporating both *local ranking* and *global matching* metrics to assess performance under *unsupervised* and *semi-supervised* settings. Local ranking metrics focus on an OM system's ability to differentiate between true mappings and challenging false mappings, while global matching metrics evaluate the system's predicted mappings against the reference mappings. The semi-supervised setting allows for the assessment of ML-based systems that necessitate training data.

Local Ranking In local ranking, for a given reference mapping $m = (c, r, c')$, an OM system is tasked with ranking m among a set of negative mappings ($\mathcal{N}(m)$), which is formed by combining c with a set of mismatched candidate concepts from the target ontology $\mathcal{O}_{\text{tgt}}$. The negative candidate generation approach has been detailed in Section

4.3.2. Utilising the scores assigned by the OM system, we apply ranking-based metrics such as Hits@K and MRR, defined in Definition 2.4.6 of Chapter 2.

Global Matching Global matching evaluates the OM system's overall performance by considering its output against reference mappings. This evaluation concerns the joint performance of various system components like mapping search, blocking, extension, and repair, beyond just the scoring module. Standard metrics here include Precision, Recall, and F-score, as defined in Definition 2.4.2 of Chapter 2.

Though global matching offers a holistic view of an OM system's efficacy, it is less suited for developing ML-based mapping scoring modules due to the dependency on multiple system components and the intensive computation involved in mapping search. Moreover, when reference mappings are not complete, systems with high Recall may be unduly penalised. Local ranking metrics, by focusing on the system's scoring capability in isolation, offer a targeted and efficient evaluation approach, complementing the broader insights provided by global matching metrics.

Data Splitting For both evaluation paradigms, we adopt two data splitting strategies for handling reference mappings:

- **Unsupervised setting:** Reference mappings are divided into two subsets: a hold-out validation set comprising 10% of the reference mappings for hyperparameter tuning or model selection,[11] and a 90% test set for the final assessment. This setup can be used to evaluate both fully automatic non-ML-based OM systems and unsupervised ML-based systems.

- **Semi-supervised setting:** Reference mappings are divided into three separate portions: 20% for training, 10% for validation, and the remaining 70% for testing. This setup is tailored for ML-based OM systems that can benefit from a limited set of training mappings. The conventional supervised learning data split, which typically involves a substantial training set, is less suitable for OM due to the significant imbalance between correct and incorrect mappings.

[11] The validation set has been removed in the 2023 edition of Bio-ML.

When computing Precision, Recall, and F-score for a specific reference mapping subset, it is crucial to exclude reference mappings outside this subset. For instance, to calculate Precision and Recall on the test set $\mathcal{M}_{\text{test}}$, we use the following formulas:

$$P_{\text{test}} = \frac{|\mathcal{M}_{\text{pred}} \cap \mathcal{M}_{\text{test}}|}{|\mathcal{M}_{\text{pred}} \setminus (\mathcal{M}_{\text{ref}} \setminus \mathcal{M}_{\text{test}})|}, \quad R_{\text{test}} = \frac{|\mathcal{M}_{\text{pred}} \cap \mathcal{M}_{\text{test}}|}{|\mathcal{M}_{\text{ref}} \setminus (\mathcal{M}_{\text{ref}} \setminus \mathcal{M}_{\text{test}})|}$$

Moreover, during the validation phase, while direct computation of Precision, Recall, and F-score is not feasible due to the incomplete mapping set (the test set has not been revealed yet), hyperparameters can still be tuned using alternative metrics like accuracy and ranking-based ones.

Nevertheless, we have recently identified an issue where our method of data splitting might allow for leakage of subsumption mappings. That is, certain test mappings could be inferred from training mappings through trivial subsumption reasoning. In our analysis of five tasks–OMIM-ORDO, NCIT-DOID, SNOMED-FMA (Body), SNOMED-NCIT (Pharm), and SNOMED-NCIT (Neoplas)–we discovered 0, 1, 53, 1,and 0 training mappings, respectively, that contributed to this information leak. To mitigate this, we plan to exclude these mappings from the training set in the next release of the OAEI.

4.3.5 Bio-LLM Sub-track for Large Language Models

Another update we introduced in the 2023 edition of Bio-ML is a specific sub-track (Bio-LLM) for evaluating large language models (LLMs) in OM.

Resource Construction Considering the extensive time and resources required to evaluate LLMs using standard or large-scale OM datasets, we have curated two focused subsets from the NCIT-DOID and SNOMED-FMA (Body) equivalence matching datasets. These subsets were chosen due to the rich hierarchical structure present in their ontologies.

To construct these subsets, we started by selecting 50 concept pairs from the reference mappings of each dataset. This selection explicitly excludes pairs that could be aligned through direct string matching, thereby diminishing the impact of surface-form lexical matching techniques. For each selected source ontology concept, we then identified 99 non-matching concepts from the target ontology, ensuring a pool of 100 candidate mappings, including the reference mapping. The selection leans on a sub-word inverted

index-based idf scoring method similar to that employed in BERTMap's candidate selection process (see Section 4.2.4), which adeptly generates target concepts lexically similar to the source concept. Subsequently, we selected an additional 50 source concepts that lack corresponding matches in the target ontology, each paired with 100 candidate mappings, thereby creating balanced scenarios for evaluation.

Thus, each subset contains 50 source concepts with corresponding target matches and another 50 without, with every concept paired to 100 candidate mappings. This setup results in a total of 10,000 (i.e., (50+50)*100) concept pairings per subset, crafted to offer a concentrated yet comprehensive examination of LLMs' capabilities in OM tasks.

Evaluation Metrics For evaluation, we employ both local ranking metrics (Hits@K and MRR) and global matching metrics (Precision, Recall, and F-score), all calculated with respect to the tailored subsets. In addition, for the 50 unmatched source concepts, we compute the Rejection Rate (RR) to assess the system's ability to correctly identify that there are no valid matches. A rejection is deemed successful if the system predicts all candidate mappings for an unmatched source concept as false. These unmatched source concepts are tagged with a "null" concept, denoted as c_{null}, leading to a collection of "unreferenced" mappings, $\mathcal{M}_{\mathsf{unref}}$. The RR is thus defined as:

$$\mathsf{RR} = \frac{1}{|\mathcal{M}_{\mathsf{unref}}|} \sum_{(c,c_{null}) \in \mathcal{M}_{\mathsf{unref}}} \prod_{d \in \mathsf{candidates}(c)} (1 - \mathbb{I}_{c \equiv d})$$

where $\mathbb{I}_{c \equiv d}$ is an indicator function that returns 1 if a match between c and d is predicted, and 0 otherwise. The product term equals 1 only when all candidates in $\mathcal{T}_c$ are classified as incorrect matches.

4.4 Evaluation

This section delves into the evaluation of our BERTMap system, focusing on its performance with the LargeBio dataset and then Bio-ML. Additionally, we introduce a preliminary analysis of LLMs for OM using Bio-LLM.

4.4.1 Evaluation on LargeBio

The evaluation considers the FMA-SNOMED and FMA-NCIT small fragment tasks from the OAEI LargeBio track. The reference mappings for these tasks were derived from an older version of UMLS, approximately dating back to 2011-2013. Table 4.4 provides a summary of the concept counts in the source and target ontologies, along with the number of reference mappings. The notations used are consistent with those introduced for Bio-ML in Tables 4.2 and 4.3. An additional notation $\mathcal{M}_?$ is used to represent reference mappings (which have been excluded from $\mathcal{M}_{\text{ref}}$) that will cause logical inconsistencies after alignment. In LargeBio, such mappings are considered null mappings and are excluded from the evaluation, i.e., neither correct nor incorrect.

Task	$\lvert\mathcal{O}_{\text{src}}\rvert_{\mathcal{C}}$	$\lvert\mathcal{O}_{\text{tgt}}\rvert_{\mathcal{C}}$	$\lvert\mathcal{M}_{\text{ref}}\rvert$	$\lvert\mathcal{M}_?\rvert$
FMA-SNOMED	10,157	13,412	6,026	2,982
FMA-NCIT	3,696	6,488	2,686	338

Table 4.4: Dataset statistics of LargeBio's FMA-SNOMED and FMA-NCI tasks.

In addition to the standard FMA-SNOMED task, we introduce an enhanced version, termed FMA-SNOMED+, which expands the target ontology by incorporating labels from the most recent version of SNOMED (v20210131) available at the time of BERTMap's introduction. This expansion addresses the issue of the outdated SNOMED version used in LargeBio, where significant updates have occurred, including changes to the naming scheme and the addition of numerous concept labels.

To construct SNOMED+, we implement the following procedure: For each concept c in SNOMED, we identify its set of labels $S(c)$. For every label s in $S(c)$, we search the latest SNOMED version for concepts with s as a label. We then integrate all labels from these found concepts into LargeBio's SNOMED, thus creating SNOMED+.

Furthermore, these additional labels are employed to generate an auxiliary corpus for the FMA-SNOMED task. The primary distinction lies in their application: In the FMA-SNOMED task, these labels are solely utilised for fine-tuning BERTMap as they are "auxiliary", whereas, in the FMA-SNOMED+ task, they contribute to both fine-tuning and the prediction process because they have been included in the input ontologies.

Evaluation Metrics The global matching evaluation metrics, Precision, Recall, and F-score are adopted except that these metrics are modified to account for $\mathcal{M}_?$ by excluding $\mathcal{M}_?$ from both the predicted and reference mappings. As a result, the Precision and Recall on the test set are defined as:

$$P_{\text{test}} = \frac{|(\mathcal{M}_{\text{pred}} \cap \mathcal{M}_{\text{test}}) \backslash \mathcal{M}_?|}{|\mathcal{M}_{\text{pred}} \setminus (\mathcal{M}_{\text{ref}} \setminus \mathcal{M}_{\text{test}}) \backslash \mathcal{M}_?|}, \quad R_{\text{test}} = \frac{|(\mathcal{M}_{\text{pred}} \cap \mathcal{M}_{\text{test}}) \backslash \mathcal{M}_?|}{|\mathcal{M}_{\text{ref}} \setminus (\mathcal{M}_{\text{ref}} \setminus \mathcal{M}_{\text{test}}) \backslash \mathcal{M}_?|}$$

Data Splitting Similar to BIO-ML,[12] we apply data splitting to LargeBio, resulting in two settings: *(i)* the unsupervised setting with 10% for validation ($\mathcal{M}_{\text{val}}$) and 90% for testing ($\mathcal{M}_{\text{test}}$); *(ii)* the semi-supervised setting with 20% for training ($\mathcal{M}_{\text{train}}$), 10% for validation ($\mathcal{M}_{\text{val}}$), and 70% for testing ($\mathcal{M}_{\text{test}}$).

Implementation We maintain a positive-negative sample ratio of $1:4$ across all text semantics corpora utilised. Specifically, for each synonym pair in an inter-ontology corpus (available in the semi-supervised setting), we sample 4 non-synonyms. In the case of intra-ontology corpora, the approach is to select 2 soft non-synonyms and 2 hard non-synonyms for every synonym identified.

For our model, we choose Bio-Clinical BERT, which is pre-trained on biomedical and clinical domain corpora, aligning well with our dataset's context (Alsentzer et al., 2019). We fine-tune this BERT model over 3 epochs using a batch size of 32. The model's performance is periodically evaluated on the validation set after every 0.1 epoch interval. Through this evaluation, we select the best-performing checkpoint based on the lowest cross-entropy loss for subsequent mapping prediction. It is important to note that the validation set for fine-tuning does not refer to the validation mappings, $\mathcal{M}_{val}$, but rather 10% of the label pairs in the overall text semantics corpora. The cut-off limit of the candidate selection process is set to 200.

The original implementation[13] of BERTMAP depends on *(i)* `owlready2` for ontology processing and *(ii)* `transformers` for BERT. The reliance on *(i)* has been lifted when BERTMAP was integrated into `DeepOnto` as a tool[14].

[12] In fact, the evaluation on LargeBio came before the creation of BIO-ML; the latter formally defined the relevant settings.

[13] Original BERTMAP repository: `https://github.com/KRR-Oxford/BERTMap`

[14] BERTMAP in `DeepOnto`: `https://krr-oxford.github.io/DeepOnto/bertmap/`

Baselines We compare BERTMap with various baselines as follows: *(i)* BERTMapLt, a light-weight (Lt) version of BERTMap, where the mapping prediction relies only on the string-matching module; *(ii)* LogMap and *(iii)* AML, which are the leading systems in many OAEI tracks and beyond; *(iv)* LogMapLt, the lexical matching part of LogMap; *(v)* LogMap-ML*, which is a variant of LogMap-ML (Chen et al., 2021b) using no branch conflicts but only LogMap anchor mappings for extracting samples for training, where Word2Vec is used to embed the concept label and a Siamese Neural Network is used as the classifier. For the internal baseline (BERTMapLt) we ensure consistency by applying the same candidate selection process and hyperparameter tuning as used in BERTMap. In contrast, the external systems (LogMap, AML, LogMapLt, and LogMap-ML*) are evaluated using their default configurations.; whereas *(iii)* to *(v)* are external systems with default implementations. Note that by comparing BERTMap against these LogMap and AML, there are various indirect comparisons to systems (e.g., ALOD2Vec (Portisch and Paulheim, 2018) and Wiktionary (Portisch et al., 2020)) that participated the LargeBio Track.

Results The overall results are presented in Table 4.5. BERTMap outperforms all baselines in terms of the F-score on the FMA-SNOMED and FMA-SNOMED+ tasks. In the unsupervised setting, BERTMap surpasses AML and LogMap by 1.4 and 4.2 points in F-score, respectively, on the FMA-SNOMED task. This margin increases in the semi-supervised setting, where BERTMap exceeds AML and LogMap by 3.0 and 5.4 points, respectively. The trend continues with the FMA-SNOMED+ task, where the improvements are 2.5 (vs. AML) and 1.8 (vs. LogMap) in the unsupervised setting, and 3.3 (vs. AML) and 2.7 (vs. LogMap) in the semi-supervised setting. On the FMA-NCIT task, BERTMap does not fare as well, trailing behind LogMap and AML in F-scores. Specifically, in the unsupervised setting, BERTMap's best F-score lags by 2.5 and 2.6 points behind AML and LogMap, respectively. This gap slightly narrows in the semi-supervised setting, with BERTMap falling short by 2.3 points against both AML and LogMap. Across all tasks, BERTMap demonstrates better performance than LogMap-ML*, LogMapLt, and the internal baseline BERTMapLt.

	Unsupervised			Semi-supervised		
System	Precision	Recall	F-score	Precision	Recall	F-score
FMA-SNOMED						
BERTMap	0.905	0.771	**0.833**	0.892	0.786	**0.836**
BERTMapLt	0.971	0.209	0.343	0.963	0.208	0.343
LogMapLt	0.965	0.206	0.339	0.956	0.204	0.336
LogMap	0.935	0.685	0.791	0.918	0.681	0.782
AML	0.892	0.757	0.819	0.865	0.754	0.806
LogMap-ML*	0.944	0.205	0.337	0.928	0.208	0.340
FMA-SNOMED+						
BERTMap	0.924	0.851	**0.886**	0.908	0.852	**0.879**
BERTMapLt	0.978	0.728	0.834	0.972	0.724	0.830
LogMapLt	0.953	0.717	0.819	0.940	0.709	0.808
LogMap	0.869	0.867	0.868	0.838	0.868	0.852
AML	0.895	0.829	0.861	0.868	0.825	0.846
LogMap-ML*	0.955	0.684	0.797	0.942	0.700	0.803
FMA-NCIT						
BERTMap	0.938	0.852	0.893	0.922	0.854	0.887
BERTMapLt	0.976	0.768	0.860	0.970	0.774	0.861
LogMapLt	0.963	0.815	0.883	0.953	0.812	0.877
LogMap	0.938	0.900	**0.919**	0.922	0.897	**0.909**
AML	0.936	0.900	0.918	0.919	0.898	**0.909**
LogMap-ML*	0.968	0.715	0.822	0.959	0.714	0.818

Table 4.5: Results on the FMA-SNOMED, FMA-SNOMED+, and FMA-NCIT tasks.

Ablation Study To assess the contribution of each component within BERTMap, we conducted an ablation study, the results of which are detailed in Table 4.6. This study focuses on the unsupervised setting of the FMA-SNOMED task and the semi-supervised setting of the FMA-SNOMED+ task. Across both settings, incremental enhancements in F-score are observed with the addition of each component. These components include the intra-ontology corpus (+intra), inter-ontology corpus (+inter), auxiliary corpus (+aux), mapping extension (+extend), and mapping repair (+repair). Notably, the intra-ontology corpus is the most influential factor of BERTMap's performance, particularly when input ontologies like FMA-SNOMED+ are rich in concept labels. Conversely, in scenarios where concept labels are sparse, as in FMA-SNOMED, the

Setting	Precision	Recall	F-score
FMA-SNOMED (Unsupervised)			
BERTMap (+intra)	0.835	0.347	0.490
+aux	0.910	0.758	0.827
+extend	0.896	0.771	0.829
+repair	0.905	0.771	0.833
FMA-SNOMED+ (Semi-supervised)			
BERTMap (+intra)	0.906	0.832	0.868
+inter	0.913	0.836	0.873
+extend	0.899	0.852	0.875
+repair	0.908	0.852	0.879

Table 4.6: Ablation results of BERTMap on two selected settings.

auxiliary corpus can play an important role.

FMA Concept	SNOMED Concept
Third Cervical Spinal Ganglion	C3 Spinal Ganglion
Deep Posterior Sacrococcygeal Ligament	Structure of Deep Dorsal Sacrococcygeal Ligament
Wall of Smooth Endoplasmic Reticulum	Agranular Endoplasmic Reticulum Membrane

Table 4.7: Typical examples of reference mappings that are predicted by BERTMap but missed by LogMap and AML.

Case Study In Table 4.7, we present examples of reference mappings successfully identified by BERTMap but missed by LogMap and AML. The BERT classifier effectively associates "third cervical" with "C3" (first example), "posterior" with "dorsal" (second example), and "wall" with "membrane" (third example), demonstrating the advantage of utilising contextualised language models over traditional lexical matching.

4.4.2 Evaluation on Bio-ML

Bio-ML was introduced to supersede the LargeBio track in OAEI, which we have elaborated on in Section 4.3.4. In this section, we focus on presenting selected results of equivalence matching tasks relevant to BERTMap. We will also discuss some subsumption matching results relevant to BERTSubs, a sibling system of BERTMap as described in Section 4.1.

		Unsupervised					Semi-supervised				
Task	**System**	**P**	**R**	**F1**	**MRR**	**H@1**	**P**	**R**	**F1**	**MRR**	**H@1**
OMIM-ORDO (Disease)	BERTMap	0.730	0.572	0.641	0.873	0.817	0.762	0.548	0.637	0.877	0.823
	BERTMapLt	0.819	0.499	0.620	0.776	0.729	0.781	0.507	0.615	0.777	0.727
	LogMap	0.827	0.498	0.622	0.803	0.742	0.788	0.501	0.612	0.805	0.744
	AML	0.749	0.510	0.607	NA	NA	0.702	0.517	0.596	NA	NA
NCIT-DOID (Disease)	BERTMap	0.912	0.829	0.868	0.967	0.953	0.823	0.887	0.854	0.968	0.955
	BERTMapLt	0.912	0.776	0.838	0.904	0.884	0.889	0.771	0.826	0.903	0.883
	LogMap	0.918	0.667	0.773	0.559	0.364	0.896	0.661	0.761	0.559	0.363
	AML	0.873	0.773	0.820	NA	NA	0.841	0.770	0.804	NA	NA
SNOMED-FMA (Body)	BERTMap	0.997	0.639	0.773	0.954	0.930	0.811	0.708	0.756	0.967	0.950
	BERTMapLt	0.976	0.660	0.787	0.895	0.869	0.970	0.665	0.789	0.897	0.871
	LogMap	0.702	0.581	0.636	0.545	0.330	0.646	0.580	0.611	0.542	0.328
	AML	0.841	0.776	0.807	NA	NA	0.805	0.779	0.792	NA	NA
SNOMED-NCIT (Pharm)	BERTMap	0.966	0.606	0.745	0.919	0.876	0.941	0.724	0.818	0.963	0.941
	BERTMapLt	0.979	0.432	0.600	0.836	0.760	0.973	0.429	0.595	0.835	0.758
	LogMap	0.915	0.612	0.733	0.820	0.695	0.893	0.609	0.724	0.821	0.699
	AML	0.940	0.615	0.743	NA	NA	0.924	0.609	0.734	NA	NA
SNOMED-NCIT (Neoplas)	BERTMap	0.655	0.777	0.711	0.960	0.939	0.575	0.784	0.664	0.965	0.947
	BERTMapLt	0.815	0.709	0.759	0.900	0.876	0.775	0.713	0.743	0.900	0.876
	LogMap	0.823	0.547	0.657	0.824	0.747	0.783	0.547	0.644	0.821	0.743
	AML	0.747	0.554	0.636	NA	NA	0.696	0.552	0.616	NA	NA

Table 4.8: Partial Results of Equivalence Matching in OAEI Bio-ML 2022.

Equivalence Matching The results for equivalence matching are presented in Table 4.8. Analysing the global matching outcomes (comparing Precision, Recall, and F-score), it is evident that OMIM-ORDO (Disease) poses the most significant challenge, reflected by the lowest average F1 score, while NCIT-DOID (Disease) emerges as the least challenging task. BERTMap attains the highest F1 scores on OMIM-ORDO (Disease), NCIT-DOID (Disease), and SNOMED-NCIT (Pharm) tasks, while AML outperforms other methods on the SNOMED-NCIT (Body) task. It is somewhat unexpected to find that the more basic BERTMapLt method achieves the highest F1 on SNOMED-NCIT (Neoplas). This could be attributed to the task's ontologies, which may not provide substantial hierarchical information for more complex methods to leverage

		Unsupervised				Semi-supervised			
Task	**System**	**MRR**	**H@1**	**H@5**	**H@10**	**MRR**	**H@1**	**H@5**	**H@10**
OMIM-ORDO (Disease)	Word2Vec+RF	0.191	0.106	0.223	0.362	0.193	0.110	0.233	0.315
	OWL2Vec*+RF	0.270	0.160	0.362	0.521	0.284	0.151	0.411	0.534
	BERTSubs	0.299	0.108	0.473	0.613	0.295	0.139	0.472	0.667
NCIT-DOID (Disease)	Word2Vec+RF	0.306	0.206	0.390	0.510	0.363	0.263	0.448	0.566
	OWL2Vec*+RF	0.388	0.285	0.485	0.604	0.422	0.315	0.524	0.647
	BERTSubs	0.601	0.460	0.777	0.877	0.618	0.496	0.758	0.862
SNOMED-FMA (Body)	Word2Vec+RF	0.558	0.415	0.731	0.850	0.629	0.503	0.792	0.886
	OWL2Vec*+RF	0.668	0.540	0.836	0.911	0.743	0.626	0.900	0.944
	BERTSubs	0.589	0.422	0.816	0.939	0.622	0.490	0.788	0.878
SNOMED-NCIT (Pharm)	Word2Vec+RF	0.488	0.335	0.687	0.852	0.526	0.402	0.663	0.834
	OWL2Vec*+RF	0.524	0.364	0.738	0.870	0.579	0.446	0.747	0.893
	BERTSubs	0.504	0.321	0.762	0.920	0.476	0.281	0.715	0.900
SNOMED-NCIT (Neoplas)	Word2Vec+RF	0.512	0.368	0.694	0.834	0.577	0.433	0.773	0.880
	OWL2Vec*+RF	0.603	0.461	0.782	0.860	0.666	0.547	0.827	0.880
	BERTSubs	0.530	0.333	0.786	0.948	0.638	0.463	0.859	0.953

Table 4.9: Partial Results of Subsumption Matching in OAEI Bio-ML 2022.

effectively. Regarding local ranking results, AML's outcomes are not included as it lacks a mechanism for assigning scores to individual concept pairs. BERTMap consistently surpasses both BERTMapLt and LogMap in local ranking metrics (MRR and Hits@1), aligning with expectations due to its competent BERT-based machine learning module. It is worth noting that BERTMapLt utilises a general edit similarity algorithm, i.e., $1 -$ normalised edit distance, for computing mapping scores in ranking tasks, and in fact full string matching is a specific case of edit similarity equal to 1.0.

Subsumption Matching Table 4.9 presents the results for the subsumption matching tasks, where "+RF" indicates the utilisation of a Random Forest classifier in conjunction with embedding techniques for mapping prediction. The results reveal that OWL2Vec*+RF consistently outperforms Word2Vec+RF across all tasks, demonstrating the added value of ontology-specific embeddings in capturing semantic relationships. BERTSubs demonstrates better performance than OWL2Vec*+RF in the OMIM-ORDO (Disease) and NCIT-DOID (Disease) tasks across all metrics. However, in the SNOMED-

		Unsupervised			Semi-supervised		
Task	**System**	**P**	**R**	**F1**	**P**	**R**	**F1**
OMIM-ORDO (Disease)	BERTMap	0.734	0.576	0.646	0.645	0.592	0.617
	SORBETMatcher	0.693	0.635	0.663	0.568	0.652	0.607
	OLaLa	0.735	0.582	0.649	0.655	0.570	0.610
NCIT-DOID (Disease)	BERTMap	0.888	0.878	0.883	0.831	0.883	0.856
	SORBETMatcher	0.920	0.907	0.913	0.925	0.882	0.903
	OLaLa	0.913	0.864	0.888	0.880	0.861	0.870
SNOMED-FMA (Body)	BERTMap	0.979	0.662	0.790	0.970	0.669	0.792
	SORBETMatcher	0.618	0.749	0.677	0.794	0.704	0.746
	OLaLa	0.270	0.348	0.304	0.202	0.339	0.253
SNOMED-NCIT (Pharm)	BERTMap	0.971	0.585	0.730	0.898	0.715	0.796
	SORBETMatcher	0.973	0.607	0.748	0.876	0.604	0.715
	OLaLa	NA	NA	NA	NA	NA	NA
SNOMED-NCIT (Neoplas)	BERTMap	0.557	0.762	0.643	0.562	0.771	0.650
	SORBETMatcher	0.626	0.642	0.634	0.731	0.605	0.662
	OLaLa	0.540	0.546	0.543	0.451	0.545	0.493

Table 4.10: Partial Results of Equivalence Matching (comparing BERTMap to recent LM-based OM systems) in OAEI Bio-ML 2023.

FMA (Body), SNOMED-NCIT (Pharm), and SNOMED-NCIT (Neoplas) tasks, while BERTSubs posts lower MRR and Hits@1 scores, it achieves higher Hits@10 scores compared to OWL2Vec*+RF. The semi-supervised approach generally yields better results, aligning with the premise that a small portion of supervised data can enhance the model's ability to generalise from intra-ontology subsumptions to inter-ontology subsumptions. A notable observation is the marked difference in MRR and Hits@1 scores between subsumption matching with BERTSubs and equivalence matching with BERTMap. This, to some degree, verifies that the subsumption matching is more challenging.

Comparisons with Recent Systems The results presented in Tables 4.8 and 4.9 are based on our work in He et al. (2022b). As discussed in Section 4.1, following BERTMap, several recent studies have explored different aspects of LMs for OM. Notably, the SORBETMatcher (Gosselin and Zouaq, 2023), which utilises BERT-based ontology embedding, and OLaLa (Hertling and Paulheim, 2023), which employs the RAG paradigm,

both participated in OAEI 2023. A comparative analysis of BERTMap with these systems is provided in Table 4.10. The ranking results are not presented here because both SORBETMatcher and OLaLa do not report the relevant scores, and OLaLa does not complete the SNOMED-NCIT (Pharm) task. From these results, we observe that BERTMap is still comparable to the more recent LM-based OM systems, attaining the best F-scores on two out of five tasks in the unsupervised setting and three out of five tasks in the semi-supervised setting. The relatively unstable results of OLaLa may result from its zero-shot or few-shot prompting without fine-tuning LLMs, indicating that a full-fledged framework for LLM-based OM requires further exploration (similar to the findings on Bio-LLM in the next section).

For complete equivalence matching results, which include comparisons across a spectrum of systems including the BERTMap family, the LogMap family (Jiménez-Ruiz and Cuenca Grau, 2011), the Matcha family (Cotovio et al., 2024), AMD (Wang, 2022), ATMatcher (Hertling and Paulheim, 2022) (OAEI 2022 only), LSMatch (Sharma et al., 2022) (OAEI 2022 only), SORBETMatcher (Gosselin and Zouaq, 2023) (since OAEI 2023), and OLaLa (Hertling and Paulheim, 2023) (since OAEI 2023), please refer to the OAEI website[15]. The subsumption matching tasks have seen fewer participants compared to the equivalence matching tasks. Main contributors include BERTSubs (Chen et al., 2023), OWL2Vec* (Chen et al., 2021a), and SORBETMatcher (since OAEI 2023).

4.4.3 Evaluation on Bio-LLM

Finally, we delve into a preliminary testing of large language models (LLMs) in the context of zero-shot OM, evaluating on the Bio-LLM datasets with the corresponding metrics introduced in Section 4.3.5.

The Bio-LLM evaluation is crafted to bypass the candidate selection process, focusing on a carefully chosen yet challenging subset to test the capabilities of LLMs in zero-shot mapping prediction. To this end, we devise the following concept identification process.

[15] https://www.cs.ox.ac.uk/isg/projects/ConCur/oaei/index.html

Concept Identification This is essentially a binary classification task that determines if two concepts, given their names (multiple labels per concept possible) and/or additional structural context, are identical or not. As LLMs typically operate in a conversational style, we need to provide a task prompt that incorporates the available information of two input concepts, and gather classification results from the responses of LLMs. To avoid excessive prompt engineering, we present the task description (as in previous sentences) and the available input information (such as concept labels and structural context) to ChatGPT based on GPT-4[16], and ask it to generate a task prompt for an LLM like itself. The resulting template is as follows:

> Given the lists of names *and hierarchical relationships* associated with two concepts, your task is to determine whether these concepts are identical or not. Consider the following:
>
> Source Concept Names: <list of concept names>
>
> *Parent Concepts of the Source Concept: <list of concept names>*
>
> *Child Concepts of the Source Concept: <list of concept names>*
>
> ... (same for the target concept)
>
> Analyze the names *and the hierarchical information* provided for each concept and provide a conclusion on whether these two concepts are identical or different ("Yes" or "No") based on their associated names *and hierarchical relationships*.

The italicised part is generated in the second round when we inform ChatGPT parent/child context can be considered. Since the prompt indicates a yes/no question, we anticipate the generation of "Yes" or "No" tokens in the LLM responses. For simplicity, we use the generation probability of the "Yes" token as the classification score. Note that this score is proportional to the final mapping score but is not normalised. For ranking-based evaluation, given a source concept, we also consider candidate target concepts with the "No" answer as well as their "No" scores, placing them after the candidate target concepts with the "Yes" answer in an ascending order – a larger "No" score implies a lower rank.

Model Settings In our experiments, we consider two LLMs, namely Flan-T5-XXL (Chung et al., 2022) and GPT-3.5-turbo, while using BERTMap and BERTMapLt as baselines. We examine Flan-T5-XXL under various settings: *(i)* the vanilla setting, where a mapping is deemed true if it is associated with a "Yes" answer; *(ii)* the threshold[17] setting, which

[16] ChatGPT (GPT-4 version): `https://chat.openai.com/?model=gpt-4`

[17] The thresholds are empirically set to 0.650, 0.999, and 0.900 for Flan-T5-XXL, BERTMap, and BERTMapLt in a pioneer experiment concerning small fragments.

filters out the "Yes" mappings with scores below a certain threshold; *(iii)* the parent/child setting, where sampled parent and child concept names are included as additional context; and *(iv)* parent/child+threshold setting, incorporating both structural context and thresholding. We also conduct experiments for GPT-3.5-turbo, the most capable variant in the GPT-3.5 series, using the same prompt. However, only setting *(i)* is reported due to a high cost of this model.

System	Precision	Recall	F-score	Hits@1	MRR	RR
Flan-T5-XXL	0.643	0.720	0.679	0.860	0.927	0.860
+ threshold	**0.861**	0.620	**0.721**	0.860	0.927	**0.940**
+ parent/child	0.597	**0.740**	0.661	0.880	0.926	0.760
+ threshold & parent/child	0.750	0.480	0.585	0.880	0.926	0.920
GPT-3.5-turbo	0.217	0.560	0.313	-	-	-
BERTMap	0.750	0.540	0.628	**0.900**	**0.940**	0.920
BERTMapLt	0.196	0.180	0.187	0.460	0.516	0.920

Table 4.11: Results on the NCIT-DOID (Disease) task of Bio-LLM.

System	Precision	Recall	F-score	Hits@1	MRR	RR
Flan-T5-XXL	0.257	0.360	0.300	0.500	0.655	0.640
+ threshold	0.452	0.280	0.346	0.500	0.655	0.820
+ parent/child	0.387	0.240	0.296	**0.540**	0.667	0.900
+ threshold & parent/child	0.429	0.120	0.188	**0.540**	0.667	0.940
GPT-3.5-turbo	0.075	0.540	0.132	-	-	-
BERTMap	0.485	**0.640**	**0.552**	**0.540**	**0.723**	0.920
BERTMapLt	**0.516**	0.320	0.395	0.340	0.543	**0.960**

Table 4.12: Results on the SNOMED-FMA (Body) task of Bio-LLM.

Results As shown in Tables 4.11 and 4.12, we observe that the Flan-T5-XXL (+threshold) obtains the best F-score among its settings. While it outpaces BERTMap by 9.3 points in F-score on the NCIT-DOID subset but lags behind BERTMap and BERTMapLt by 20.6 and 4.9 points, respectively, on the SNOMED-FMA (Body) subset. Regarding MRR, BERTMap leads on both subsets. Among Flan-T5-XXL settings, using a threshold enhances precision but reduces recall. Incorporating parent/child context does not enhance matching results – this underscores the need for a more in-depth examination

of strategies for leveraging ontology context. GPT-3.5-turbo[18] does not perform well with the given prompt. One possible reason is the model's tendency to furnish extended explanations for its responses, making it challenging to extract straightforward yes/no answers. Besides, no ranking scores are presented for GPT-3.5-turbo because it does not support extracting generation probabilities. The suboptimal performance of BERTMapLt is as expected because we exclude concept pairs that can be string-matched from the extracted datasets while BERTMapLt relies on the edit similarity score.

[18]The experimental trials for text-davinci-003 and GPT-4 also showed suboptimal results.

CHAPTER 5

Language Models for Ontology Completion

Contents

This chapter concerns our contributions to ontology completion, specifically focusing on subsumption inference (defined in Section 2.4.3 of Chapter 2) using language models (LMs). The discussion is framed within the context of "LMs-as-KBs" from NLP research, exploring the capacity of pre-trained LMs to function as knowledge bases (KBs) for knowledge inquiry – in our case, the knowledge of subsumption relationships (He et al., 2023c). We construct a collection of LM probing datasets, termed OntoLAMA, along with a prompt-based probing method for this purpose.

5.1 Overview

The advancements of transformer-based pre-trained language models (LMs) have sparked research interests in investigating how much explicit semantics LMs can learn or infer from knowledge bases (KBs) (AlKhamissi et al., 2022). The LAMA (LAnguage Model Analysis) probe (Petroni et al., 2019a) is among the first works that adopt prompt-based methods to simulate the process of querying relational knowledge from various KBs such as ConceptNet (Speer et al., 2012) and GoogleRE[1]. Some subsequent studies focus on probing specific types of knowledge from sources like commonsense KBs (Da et al., 2021), biomedical KBs (Sung et al., 2021), temporal KBs (Dhingra et al., 2022), and cross-lingual KBs (Liu et al., 2022).

However, the existing "LMs-as-KBs" studies primarily focus on simple, triple-based, relational KBs, but neglect more structured, logic-based, conceptualised KBs. As illustrated in Example 2.4.2, the statement "London is the capital of the UK" can be represented by the triple $(\mathsf{London}, \mathsf{capitalOf}, \mathsf{UK})$; but a statement like "arthritis is a type of arthropathy characterised by inflammation" encapsulates conceptual knowledge that transcends simplistic triple-based representation. Such fine-grained knowledge demands a more formalised and expressive framework for precise definition, a gap well-addressed by OWL ontologies. As detailed in Section 2.3 of Chapter 2, OWL ontologies can be seen as description logic (DL) KBs with rich built-in vocabularies for knowledge representation and various reasoning tools supported. Taking the example of Arthritis, in OWL it can be formally described as $\mathsf{Arthritis} \sqsubseteq \mathsf{Arthropathy} \sqcap \exists \mathsf{hasMorphology}.\mathsf{Inflammation}$.

We take a further step along the "LMs-as-KBs" research line towards more formalised semantics by targeting DL KBs and in particular the OWL ontologies. This, in turn, benefits the task of subsumption inference. As shown in Figure 5.1, our process begins with the extraction of concept pairs (C, D) from an ontology, where these pairs are classified as positive (indicating an entailed subsumption relationship) or negative (suggesting that C and D are *assumed* to be disjoint). This extraction is an automated process, carefully designed to take the syntax and semantics of OWL

[1] https://code.google.com/archive/p/relation-extraction-corpus/

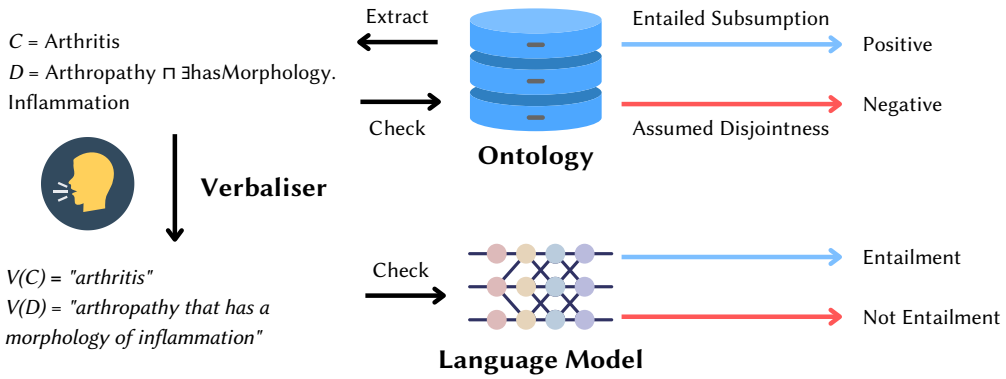


Figure 5.1: OntoLAMA framework.

ontologies into account. To translate the concepts and especially the ones with complex logical expressions into natural language texts, we develop a *recursive concept verbaliser*. We formulate the Subsumption Inference (SI) task similar to the Natural Language Inference (NLI) task, treating the concept pairs as analogous to premise-hypothesis pairs in NLI contexts (Padó and Dagan, 2022). These pairs are then wrapped into a NLI-based template, serving as inputs for LMs.

We created SI datasets from ontologies of various domains and scales, and conducted extensive experiments. Our results demonstrate that LMs perform better on a typical NLI task than the constructed SI tasks under the zero-shot setting, indicating that LMs encode relatively less background knowledge of ontology subsumptions. However, by providing a small number of samples (K-shot settings), the performance on SI is significantly improved. This observation is consistent with the three LMs that were experimented with.

5.2 Related Work

The prompt learning paradigm, as we have introduced in Section 3.3.5 of Chapter 3, has shed light on better usage of pre-trained LMs without, or with minor, supervision. However, LMs are typically pre-trained in a stochastic manner, making it challenging to study what knowledge LMs have implicitly encoded (Petroni et al., 2019a) and how to access LMs in an optimal or controllable way (Gao et al., 2021a; Li et al., 2022).

Therefore, prompt engineering is gaining its importance, as one of the pivotal studies for effectively attaining knowledge from LMs.

Our work is informed by the "LMs-as-KBs" literature, where different prompt-based probes have been designed to test LMs' knowledge of relational data. In Petroni et al. (2019a), the probing task of world knowledge has been formulated as a cloze-style answering task where LMs are required to fill in the [MASK] token given input texts wrapped into a manually designed template. Sung et al. (2021) did a similar work but shift the focus to (biomedical) domain knowledge of domain-specific LMs. Liu et al. (2022) pre-trained LMs with multi-lingual knowledge graphs (KGs) and tested on the cross-lingual tasks. Dhingra et al. (2022) proposed datasets with temporal signals and probed LMs on them with templates generated by the text-to-text transformer T5 (Raffel et al., 2020). In contrast to these existing "LMs-as-KBs" works which mostly focus on relational facts, our work was the first that introduce conceptual knowledge and logical formalism to the realm. It is worth mentioning that at the time of this writing, there were some follow-up works that attempted to address formal semantics more carefully (Poulis et al., 2023) or extended the analysis to large language models (Li et al., 2024).

5.3 Subsumption Inference

5.3.1 Reformulation

As previously discussed in Section 2.4.3 of Chapter 2, subsumption inference (SI) is a pivotal sub-task in ontology completion, aimed at expanding the TBox of an ontology with missing subsumption axioms. We now re-formulate this task within the framework of Natural Language Inference (NLI).

A subsumption axiom, denoted as $C \sqsubseteq D$, implies that "every instance of C is also an instance of D". This logical statement allows us to construct a premise-hypothesis pair: the *premise* being "x is a C" and the *hypothesis* being "x is a D", where x represents an arbitrary individual. While various formulations are possible for these statements, we adopt a straightforward approach for clarity and efficacy (detailed further in Section 5.4).

To render the concept expressions C and D into natural language, we develop an *ontology verbaliser*. Drawing parallels to NLI or Recognising Textual Entailment (RTE)

(Poliak, 2020; Padó and Dagan, 2022), we define the SI task as determining whether the given premise entails the hypothesis (positive) or not (negative). This classification is similar to a two-way RTE task, where we exclude the *neutral* class. In the context of ontological modelling, the neutral class, which indicate unrelatedness between two statements, is less applicable. This is becasue ontologies are invariably underspecified – even when two concepts have not been entailed as a subsumption or non-subsumption, they may still be related in real world due to OWA (discussed on Page 16).

5.3.2 Assumed Disjointness

In an ontology $\mathcal{O}$, we derive positive and negative samples to probe LMs. Positive samples are concept pairs (C, D) where $\mathcal{O} \models C \sqsubseteq D$. However, due to OWA, negative subsumption cannot be directly inferred when $\mathcal{O} \not\models C \sqsubseteq D$. We address this by introducing *assumed disjointness* to generate plausible negative samples, inspired by approaches in (Schlobach, 2005; Solimando et al., 2017) yet adapted for our context:

Definition 5.3.1 (Assumed Disjointness)

For two concepts C and D in $\mathcal{O}$, if they remain satisfiable in $\mathcal{O} \cup \{C \sqcap D \sqsubseteq \bot\}$ (**C1**), and there is no named atomic concept A in $\mathcal{O}$ such that both $\mathcal{O} \models A \sqsubseteq C$ and $\mathcal{O} \models A \sqsubseteq D$ hold (**C2**), then we assume C and D to be disjoint.

Condition **C1** ensures C and D maintain satisfiability when postulated as disjoint, while **C2** confirms no common descendants, preventing them from becoming unsatisfiable. Satisfying these conditions allows treating (C, D) as a valid negative subsumption.

Given the impracticality of verifying satisfiability for each (C, D) in complex ontologies, we propose a practical approach to replace **C1**, termed *assumed satisfiability*:

Definition 5.3.2 (Assumed Satisfiability)

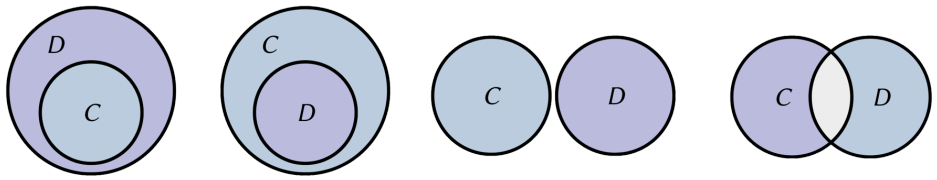
Concepts C and D are assumed to be satisfiable in $\mathcal{O} \cup \{C \sqcap D \sqsubseteq \bot\}$ if they are not in a subsumption relationship, i.e., $\mathcal{O} \not\models C \sqsubseteq D$ and $\mathcal{O} \not\models D \sqsubseteq C$ (**C3**) and share no common instance, i.e., there is no *named* instance a in $\mathcal{O}$ such that $\mathcal{O} \models C(a)$

and $\mathcal{O} \models D(a)$ (**C4**).

Since these two conditions avoid repeated reasoning invoked by adding new (disjointness) axioms, they are much more efficient than iteratively conducting satisfiability check for candidates. The choice between **C1** and **C2** or **C3**, **C4**, and **C2** can depend on the ontology's scale and complexity.

5.3.3 Set-based Semantics

To explain our rationale behind our choices of positive and negative subsumption samples, we delve into the underlying set-based semantics.



(a) $C \sqsubseteq D$ (b) $D \sqsubseteq C$ (c) $C \sqcap D \sqsubseteq \bot$ (d) $C \sqcap D \not\sqsubseteq \bot$ but not (a) or (b)

Figure 5.2: Set-based semantics for relationships between two ontology concepts.

Recall the Description Logic (DL) preliminaries in Section 2.2 of Chapter 2, an ontology $\mathcal{O}$ entails a subsumption axiom $C \sqsubseteq D$ if it holds for every interpretation $\mathcal{I}$ of $\mathcal{O}$. This scenario is visually represented as case (a) in Figure 5.2, where the set corresponding to C is entirely contained within the set for D.

On the contrary, scenarios (b), (c), and (d) depict situations where, for at least one interpretation $\mathcal{I}$, there exists an individual x such that $x^{\mathcal{I}} \in C^{\mathcal{I}}$ but $x^{\mathcal{I}} \notin D^{\mathcal{I}}$. This suggests that the subsumption $C \sqsubseteq D$ is not entailed by $\mathcal{O}$. However, this does not mean non-subsumption is entailed. In scenarios (b) and (d), it is possible to find an interpretation $\mathcal{I}$ where individuals reside only within the overlap of C and D, thus satisfying $\mathcal{I} \models C \sqsubseteq D$. For a non-subsumption to be entailed, it is necessary to prevent case (a) across all interpretations $\mathcal{I}$ of $\mathcal{O}$, or in other words, $\exists x.C(x) \wedge \neg D(x)$ is satisfied in every possible interpretation.

Disjointness, depicted in *(c)*, arises when there is no intersection between the sets of C and D ($\forall x.C(x) \rightarrow \neg D(x)$) in any interpretation, which is more restrictive than non-subsumption. It is common to use disjointness to imply non-subsumption and thus deriving negative subsumption samples. However, ontologies are typically underspecified, meaning that they do not explicitly define all possible disjointness axioms. Therefore, generating an adequate number of negative samples directly from disjointness axioms is impractical.

To find a middle ground, it is reasonable to adopt heuristics. The assumed disjointness we propose in the previous section serves this purpose. In an ideal scenario, verifying **C1** and **C2** avoids *(a)* and *(b)*, and reduces the chance of *(d)*. Similarly, under our practical alternative, **C3** ensures that *(a)* and *(b)* are not entailed, while **C2** and **C4** collectively lower the chances of encountering *(d)*. Thus, our heuristic approach of assumed disjointness provides a viable strategy to approximate non-subsumptions.

5.3.4 Concept Verbalisation

To transform concept expressions (especially complex ones) in OWL, we develop a *recursive concept verbaliser* consisting of a *syntax tree parser* and a set of *recursive rules* (see Table 5.1 that follows).

Pattern	Verbalisation ($\mathcal{V}$)
A (atomic)	the name (`rdfs:label`) of A
r (property)	the name (`rdfs:label`) of r; append "is" to the head if started with a *passive verb*, *adjective*, or *noun*.
$\exists/\forall$	"some"/"only" or none.
$\neg C$	"not $\mathcal{V}(C)$"
$\exists/\forall r.C$	"something that $\mathcal{V}(r)$ $\mathcal{V}(\exists/\forall)$ $\mathcal{V}(C)$"
$C_1 \sqcap ... \sqcap C_n$	*Step 1:* Merge every $C_i = \exists/\forall r.D_i$ and $C_j = \exists/\forall r.D_j$ into $\exists/\forall r.(D_i \sqcap D_j)$, resulting in $C_1 \sqcap \ldots \sqcap C_{n'}$. *Step 2:* Verbalise $C_1 \sqcap ... \sqcap C_{n'}$: $\mathcal{V}_a :=$ "$\mathcal{V}(C_1)$ and ... and $\mathcal{V}(C_{n'})$" if *no* C_i is a restriction. $\mathcal{V}_b :=$ "something that $\mathcal{V}(r_1)$ $\mathcal{V}(\exists/\forall)$ $\mathcal{V}(D_1)$ and ... and $\mathcal{V}(r_{n'})$ $\mathcal{V}(\exists/\forall)$ $\mathcal{V}(D_{n'})$" if *all* C_is are restrictions. $\mathcal{V}_c :=$ "$\mathcal{V}_a(C_1, ..., C_m)$ that $\mathcal{V}_b(C_{m+1}, ..., C_{n'})$" if *some* C_is, for $i = 1, ..., m$, are restrictions.
$C_1 \sqcup ... \sqcup C_n$	Similar to $\mathcal{V}(C_1 \sqcap ... \sqcap C_n)$ but replace "and" with "or" and replace $\mathcal{V}_c$ with $\mathcal{V}_a$.

Table 5.1: Recursive rules for verbalising a concept expression C in OWL ontologies.

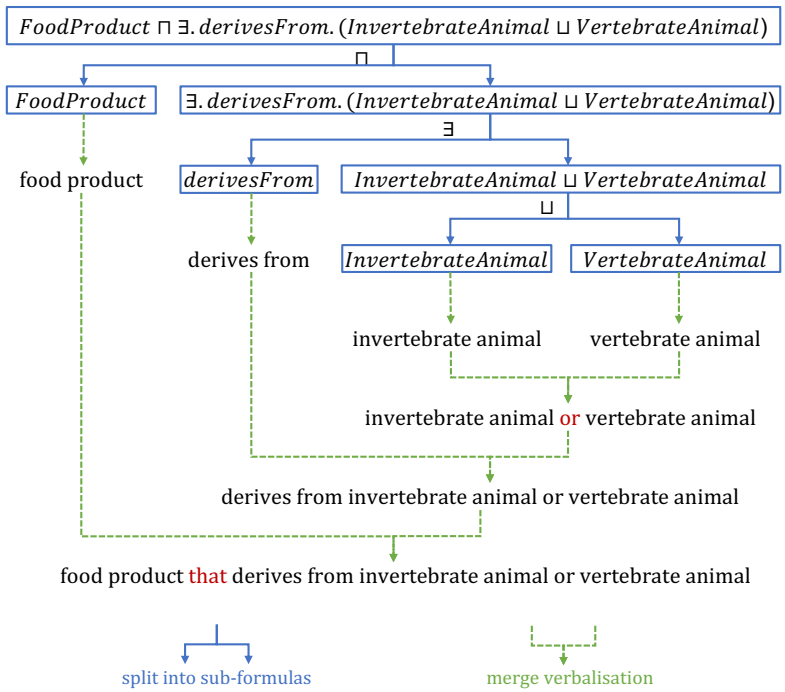


Figure 5.3: Illustration of how the recursive concept verbaliser is applied to an example complex concept expression. The algorithm first splits the complex concept into a sub-formula tree, verbalising the leaf nodes, and then merging the verbalised sub-formulas recursively. The key word associated with the logical operator at each merging step is marked in red.

A concrete example is shown in Figure 5.3, where the complex concept is first split into a sub-formula tree by the syntax parser, then verbalised according to the recursive rules in Table 5.1. The leaf nodes are either atomic concepts or properties which can be verbalised by their names. At each recursive step, verbalised child nodes are merged according to the logical pattern in their parent node.

For named atomic concepts, we simply use its label (in English) for verbalisation. For property names, we enforce some simple rules for a basic grammar fix as shown in Table 5.1. For example, "characteristic of" is changed to "is characteristic of"; "realised in" is changed to "is realised in". Note that the word's part-of-speech tag is automatically determined using the Python library `Spacy`[2].

To enhance clarity and avoid redundancy, we implement a simplification strategy for the conjunction (⊓) and disjunction (⊔) patterns. In the example depicted in Figure 5.3, the term "food product" is positioned as an *antecedent* before "that", with the verbalised

[2] `https://spacy.io/`

$C = \mathsf{BioRegulation} \sqcap \exists \mathsf{negRegulate.ProlineBiosynProc}$ $\mathcal{V}(C) =$ "biological regulation that negatively regulates some proline biosynthetic process"
$C = \mathsf{ApoptoticProc} \sqcap \exists \mathsf{partOf.Luteolysis}$ $\mathcal{V}(C) =$ "apoptotic process that is part of some lutelysis"
$C = \mathsf{ConcnOf} \sqcap \exists \mathsf{charOf.}(\mathsf{Fucose} \sqcap \exists \mathsf{partOf.MaterialEnt})$ $\mathcal{V}(C) =$ "concentration of something that is characteristic of some fucose that is part of some material entity"
$C = \exists \mathsf{derivesFrom.}(\mathsf{TimothyPlant} \sqcup \mathsf{TrifoliumPratense}) \sqcap \mathsf{PlantFoodProd} \sqcap \mathsf{Silage}$ $\mathcal{V}(C) =$ "silage and plant food product that derives from some timothy plant or trifolium pratense"
$C = \mathsf{Apple} \sqcap \neg \exists \mathsf{hasPart.ApplePeel}$ $\mathcal{V}(C) =$ "apple (whole or parts) and not something that has part some apple peel"

Table 5.2: Verbalisation examples of complex concepts from Go and FoodOn.

conjunction of two restrictions forming a *relative clause* after "that". If the concept FoodProduct is not involved in the concept expression, the antecedent can be replaced by the general word "something".

Furthermore, when encountering two restrictions sharing the same quantifier and property and connected by $\sqcap$ or $\sqcup$, we merge these into a single restriction. For instance, $\exists \mathsf{derivesFrom.InvertebrateAnimal} \sqcup \exists \mathsf{derivesFrom.VertebrateAnimal}$ would be merged into $\exists \mathsf{derivesFrom.}(\mathsf{InvertebrateAnimal} \sqcup \mathsf{VertebrateAnimal})$. This approach ensures that the verbalisation is not only faithful to the original logical expression but also presented in a streamlined manner.

For more verbalisation examples, please refer to Table 5.2, where the relevant concept expressions are derived from the Gene Ontology (Go) (Ashburner et al., 2000) and the Food Ontology (FoodOn) (Dooley et al., 2018).

Limitation Despite the aforementioned efforts, it is hard to guarantee a completely natural verbalisation of logical expressions due to the naming variety (e.g., use "director" instead of "directed by" for a property name) in ontologies. In practice, we can use LLMs like ChatGPT to fix any undesirable naming before applying this verbalisation. Note that it is computationally expensive to only use LLMs for the verbalisation. A combination of two (which is again neural-symbolic) can guarantee both efficiency and quality.

In the following sub-sections, we propose two specific SI tasks and their respective subsumption sample generation strategies.

5.3.5 Atomic Subsumption Inference

This task focuses on inferring subsumption axioms that involve only *named atomic concepts*, which are typically abundant in ontologies and can be easily verbalised using concept labels.

The positive samples are drawn from the entailed subsumption axioms of the target ontology. We consider two types of negative samples: *(i) soft* negative samples, which consist of two randomly selected concepts; *(ii) hard* negative samples, which consist of pairs of *sibling* concepts. Siblings in an ontology share a common parent, implying a closer semantic relationship, but are often considered (or assumed to be) disjoint.

Note that the definition of hard negatives is similar to Chapter 4 when we discussed BERTMap, except that here the selection of the negative samples adheres to the criteria of assumed disjointness discussed in Section 5.3.2. To ensure a balanced class distribution, we initially gather equal counts of soft and hard negatives and subsequently trim the combined set to match the size of the positive sample set.

5.3.6 Complex Subsumption Inference

This task considers subsumption axioms that involve *complex concepts* with patterns that can be verbalised by rules presented in Table 5.1. Particularly, we choose equivalence axioms of the form $A \equiv C$[3] (where A is atomic and C is complex) as anchors, and equivalently transform them into subsumption axioms of the forms $A \sqsubseteq C$ and $C \sqsubseteq A$, through which complex concepts can appear on both the premise and hypothesis sides.

We extract equivalence axioms in the form of $A \equiv C$ from the target ontology. Taking each such axiom as an anchor, we can obtain positive complex subsumption axioms of the form $\mathsf{SubClassOf}(A) \sqsubseteq C$ or $C \sqsubseteq \mathsf{SuperClassOf}(A)$. To derive challenging negative samples, we first randomly replace a named concept or a property in $A \equiv C$ to generate either *(i)* $A' \equiv C$ (if A is replaced by A') or *(ii)* $A \equiv C'$ (if C is corrupted). Without loss of generality, we assume the random replacement leads to case *(ii)*. We then check if A and C' satisfy the assumed disjointness as

[3]Equivalence axioms of this form are referred to as the *definition* of the named concept, and are common in OWL.

described in Section 5.3.1. In the affirmative case, we can have either $A \sqsubseteq C'$ or $C' \sqsubseteq A$ as the final negative subsumption; otherwise, we skip this sample. For example, given SunflowerSeed $\equiv$ Seed $\sqcap$ $\exists$DerivesFrom.HelianthusAnnuus, a possible negative subsumption is SunflowerSeed $\sqsubseteq$ Fruit $\sqcap$ $\exists$DerivesFrom.HelianthusAnnuus if Seed in C is replaced by Fruit to create C'.

5.4 Prompt-based Inference

To perform the inference task using prompt-based approaches, we encapsulate the verbalised subsumption axioms alongside a [MASK] token within a *template* to create input sequences for LMs. Drawing on Definition 3.3.7 from Chapter 3, a template is a function that integrates contextual prompts with allocated slots for inputs and expected answers. In this context, the template receives the verbalised concept pairs $\mathcal{V}(C)$ and $\mathcal{V}(D)$ as inputs, and the answer slot is represented by a [MASK] token, which the LM aims to predict. It is important to note that while Definition 3.3.7 describes a broader framework for prompt-based language modelling where the answer slot could accommodate a sequence of tokens, our focus here is specifically on masked LMs. These models typically predict a single token for the [MASK] slot. In addition, when the expected answer consists of multiple tokens, a common workaround is to use the mean embedding as the overall representation.

For specific choices of subsumption template functions, we opt to use those have achieved promising results from NLI tasks (Schick and Schütze, 2021; Gao et al., 2021a), with minor modifications to fit the SI context. Specifically, we prepend "It is [A]" to make the input sequence complete. The modified templates are as follows:

$$T_1 := \underbrace{\text{It is [A] } \mathcal{V}(C)}_{\text{premise}} \text{? [MASK], } \underbrace{\text{it is [A] } \mathcal{V}(D)}_{\text{hypothesis}}.$$

$$T_2 := \text{“}\underbrace{\text{It is [A] } \mathcal{V}(C)}_{\text{premise}}\text{”? [MASK], “}\underbrace{\text{it is [A] } \mathcal{V}(D)}_{\text{hypothesis}}\text{”.}$$

where [A] serves as a placeholder for the articles "a", "an", or no article, chosen based on the subsequent word for grammatical correctness. Specifically, "an" is used when the

following word begins with a vowel sound, and no article is used if the next word is "something".

To map the LM's predictions to their corresponding classifications, we define a set of *label words*. This involves comparing the predicted embedding for the [MASK] token with these label words to deduce the final binary classification: "positive" or "negative".

$L_1 :=$ {"positive": ["Yes"], "negative": ["No"]}

$L_2 :=$ {"positive": ["Right"], "negative": ["Wrong"]}

$L_3 :=$ {"positive": ["Yes", "Right"], "negative": ["No", "Wrong"]}

These label words have also been proven effective in NLI tasks. The predicted class y ("positive" or "negative") for a given verbalised sample $x = (\mathcal{V}(C), \mathcal{V}(D))$ is determined by the probability:

$$P(y \mid x) = P([\mathsf{MASK}] \in L_j[y]) \mid T_i(\mathcal{V}(C), \mathcal{V}(D))) = \frac{\sum_{w \in L_j[y]} \exp(\mathbf{w} \cdot \mathbf{m})}{\sum_{v \in L_j[\cdot]} \exp(\mathbf{v} \cdot \mathbf{m})}$$

Here, $L_j[\cdot]$ and $L_j[y]$ represent the complete set of label words and those associated with class y in L_j, respectively. $T_i(\mathcal{V}(C), \mathcal{V}(D))$ signifies the transformation of verbalised concepts $\mathcal{V}(C)$ and $\mathcal{V}(D)$ via template T_i. Vectors $\mathbf{w}$ and $\mathbf{v}$ correspond to the embeddings for label words w and v, and $\mathbf{m}$ refers to the embedding of the [MASK] token. From this equation, we can see that the probability of predicting a class y is computed based its labels words and normalised over all defined label words. For prompt-based fine-tuning, we can fit this probability term into the cross-entropy loss.

It is important to note that the templates T_1 and T_2 are tailored for the SI tasks. To gauge the relative challenge posed by SI tasks compared to conventional NLI, we considered a binary NLI dataset, details of which are elaborated in the following section. For this NLI task, the premise and hypothesis in templates T_1 and T_2 are directly adopted from the original NLI dataset, albeit with the removal of trailing punctuation.

5.5 Datasets

Dataset Sources To construct the SI datasets, we selected ontologies across various domains and sizes, using their latest versions at the experiment time:

- **Schema.org** (Guha et al., 2016) (version: `2022-03-17`): A general ontology that provides a schema for structured data on the Web.
- **Disease Ontology (DOID)** (Schriml et al., 2012) (version: `2022-09-29`): An ontology dedicated to human disease categorisation.
- **Food Ontology (FoodOn)** (Dooley et al., 2018) (version: `2022-08-12`): An ontology that covers a spectrum of food-related concepts, including products, sources, and nutrition.
- **Gene Ontology (GO)** (Ashburner et al., 2000) (version: `2022-11-03`): A detailed and commonly referenced ontology in biomedicine focusing on gene and gene functions.

Preprocessing To ensure data quality and relevance, we performed universal preprocessing steps across all ontologies, followed by specific adjustments for each:

- **Universal Steps**:
  - Removal of obsolete concepts tagged with the `owl:deprecated` property.
  - Elimination of evidently irrelevant concepts, like `foodOn:stupidType`.
  - Utilisation of `rdfs:label` for concept names, defaulting to lowercase and removing underscores.
- **Schema.org**: Transformed Java-identifier-style names (e.g., "APIReference") into more readable forms (e.g., "API Reference").
- **DOID**: Excluded the general `doid:Disease` concept to prevent trivial subsumptions in the form of $C \sqsubseteq \mathsf{Disease}$.

- **FoodOn**: Adjusted labels to exclude non-natural language text patterns, preserving meaningful content and substituted `rdfs:label` with `obo:hasSynonym` or `obo:hasExactSynonym` when necessary due to empty labels.

- **GO**: Required no specific adjustments.

Dataset Construction For each selected ontology, we constructed an Atomic SI dataset, while Complex SI datasets were specifically constructed for FoodOn and GO due to their abundance of equivalence axioms. To mitigate redundancy stemming from multiple samples generated by a single equivalence axiom in the Complex SI datasets, we capped the sampling at a maximum of 4 positive and 4 negative samples per axiom. For class balance, the quantity of negative samples was aligned with that of positive samples across all data splits.

The datasets were generally partitioned into training, validation, and testing sets with an 8:1:1 ratio. However, for the smaller Atomic SI dataset for Schema.org and the Complex SI dataset for FoodOn, a 2:1:7 split was applied. Given the probing study's emphasis on K-shot settings, the training and validation datasets were kept intentionally small.

In addition to SI datasets, we created a modified version of the Multi-Genre Natural Language Inference (MNLI) corpus (Williams et al., 2018), termed biMNLI. This version *(i)* omits the neutral class, *(ii)* combines Matched and Mismatched test sets into a singular testing set, *(iii)* allocates 10% of training data to validation, *(iv)* and ensures a balanced distribution between entailment and contradiction classes by discarding extra samples from the dominant class.

Table 5.3 provides a detailed breakdown of the dataset statistics, including the number of concepts, equivalence axioms, and the distribution of samples across the different splits for each SI dataset, along with the statistics for the biMNLI dataset. Table 5.4 presents examples of data points from the constructed datasets.

Source	#Concepts	#EquivAxioms	#Dataset (Train/Val/Test)
Schema.org	894	-	Atomic SI: 808/404/2,830
DOID	11,157	-	Atomic SI: 90,500/11,312/11,314
FoodOn	30,995	2,383	Atomic SI: 768,486/96,060/96,062 Complex SI: 3,754/1,850/13,080
Go	43,303	11,456	Atomic SI: 772,870/96,608/96,610 Complex SI: 72,318/9,040/9,040
MNLI	-	-	biMNLI: 235,622/26,180/12,906

Table 5.3: Dataset statistics of OntoLAMA.

Dataset	Example
Go's Atomic SI	Sub-concept: "ctpase activity" Super-concept: "ribonucleoside triphosphate phosphatase activity" Label: positive
FoodOn's Complex SI	Sub-concept: "ham and cheese sandwich that derives from some lima bean (whole)" Super-concept: "lima bean substance" Label: negative
biMNLI	Premise: "At the turn of the 19th century Los Angeles and Salt Lake City were among the burgeoning metropolises of the new American West." Hypothesis: "Salt Lake City was booming in the early 19th century." Label: positive

Table 5.4: Examples of data points in OntoLAMA.

5.6 Evaluation

5.6.1 Experiment Settings

Our evaluation primarily focuses on the K-shot performance of LMs on SI tasks and the modified biMNLI task. The zero-shot scenario, where $K = 0$, aims to evaluate the intrinsic knowledge encoded by LMs, offering insights into their immediate understanding of subsumption relationships without any task-specific training. Furthermore, we explore few-shot learning scenarios with K set to $\{4, 8, 16, 32, 64, 128\}$,[4] aiming to observe incremental improvements in model performance as a modest number of examples are introduced. The value of K refers to the number of samples per class ("positive" or

[4]For small ontologies like Schema.org, $K = 128$ may not be considered as few-shot; however, we still report the relevant results to maintain the integrity.

"negative") randomly drawn from training and validation sets. This approach aligns with common practice in prompt-based probing, where the primary goal is to maximise the use the pre-existing knowledge within LMs, rather than extensively fine-tuning them on a specific downstream task.

Across each K-shot setting, we explore all the combinations ($2 \times 3 = 6$) of templates T_i and label word sets L_j. For $K > 0$, we additionally consider 3 random seeds (thus 18 experiments per K-shot setting). For $K = 0$, random seeds do not affect the results. Moreover, a fully supervised setting is adopted an oracle, establishing the performance ceiling; here, we opt for a singular template and label word set combination (T_1 and L_1) based on preliminary findings that fine-tuning on large samples minimises variance brought by these factors.

Evaluation Metrics Our primary metric is *mean accuracy*, complemented by *standard deviation* to capture variability across template/label word combinations and random seeds. Note that this is different from ranking-based metrics in Defintion 2.4.6 of Chapter 2 that are commonly adopted in evaluating ontology completion. As we reformulate the SI task similar to NLI, we adhere to NLI's evaluation convention.

Implementation Details For each $K > 0$ experiment, the models are trained for 10 epochs, contrasting with a single epoch for the fully supervised setting, given its expansive training dataset size.[5] Model selection is epoch-wise, favouring the best-performing checkpoint on the validation set for subsequent test set evaluation. We employ the AdamW optimiser (Loshchilov and Hutter, 2018), with a learning rate of 10^{-5}, weight decay at 10^{-2}, and an initial warm-up of 50 steps. All our experiments are conducted on two `Quadro RTX 8000` GPUs.

Models We select the RoBERTa language model family (Liu et al., 2019), which are frequently introduced in cloze-style probing tasks (Liu et al., 2021b; Sung et al., 2021; Kavumba et al., 2022). The main results concern the roberta-large and roberta-base variants. Additionally, we delve into a domain-specific evaluation using a biomedical variant of roberta-large.

[5]Exceptions are made for the Atomic SI from Schema.org and Complex SI from FoodOn, where the full supervision setting extends to 10 epochs due to their smaller training sizes.

For more comparisons, we include a simple majority vote baseline (majority), which maintains a 50.0% accuracy score due to the deliberate class balance in our datasets. Another baseline under consideration is a word2vec model (Mikolov et al., 2013) pre-trained on the GoogleNews corpus[6], paired with mean pooling and a logistic regression classifier. This baseline offers a lens through which to evaluate the capabilities of a traditional non-contextual word embedding model on SI tasks. For the word2vec baseline, we only report results for $K \in \{4, 128\}$ as the increase of K does not bring significant change and results of $K = 128$ are roughly comparable to results of $K = 4$ for roberta-large. This suggests that the SI task has a basic level of challenge and cannot be easily solved using word2vec.

5.6.2 Results and Analysis

The main results are presented in Table 5.3. For clearer observation of how the performance varies as K increases, we also provide illustrative visualisations in Figure 5.4 for roberta-large and 5.5 for roberta-base. The following paragraphs delve into specific findings.

SI vs. biMNLI We first observe that both roberta-large and roberta-base achieve better zero-shot results on biMNLI than on all the SI datasets by at least 7.0% and 6.1%, respectively, showing that under our prompt settings, both LMs encode better background knowledge on biMNLI than SI. However, as K grows, the performances on both biMNLI and SI improve consistently and significantly (while the standard deviation generally reduces), and we can see at $K = 32$, the mean accuracy scores on the Atomic SI tasks have surpassed biMNLI for roberta-base. At $K = 64$, the mean accuracy scores on biMNLI and all the Atomic SI tasks converge to around 90.0%; the scores on the two Complex SI tasks are also above 80.0% for both LMs. Across all settings, roberta-large consistently outperforms roberta-base.

[6] https://code.google.com/archive/p/word2vec/

		Atomic SI				Complex SI	
Setting	biMNLI	Schema.org	DOID	FoodOn	GO	FoodOn	GO
majority	50.0	50.0	50.0	50.0	50.0	50.0	50.0
word2vec							
K=4	51.5 (0.2)	54.9 (2.9)	64.6 (2.6)	63.5 (1.0)	60.1 (4.1)	56.8 (4.2)	56.9 (5.4)
K=128	52.1 (0.4)	73.0 (0.4)	70.8 (1.7)	71.4 (1.0)	66.3 (0.9)	63.8 (0.6)	66.4 (1.3)
roberta-base							
K=0	62.5 (6.5)	56.4 (3.6)	53.3 (4.0)	54.6 (4.4)	49.0 (2.4)	55.9 (3.6)	48.7 (3.1)
K=4	67.6 (5.2)	62.9 (5.2)	61.8 (6.7)	62.1 (4.2)	65.2 (5.0)	62.4 (3.2)	52.2 (7.1)
K=8	70.7 (4.5)	71.2 (4.5)	72.9 (5.7)	69.0 (5.2)	70.4 (5.1)	66.0 (4.4)	63.0 (5.0)
K=16	74.3 (3.3)	79.7 (4.2)	83.4 (2.5)	79.8 (3.0)	78.3 (3.0)	70.2 (5.5)	73.8 (4.0)
K=32	78.8 (1.1)	84.3 (2.0)	89.0 (1.4)	85.0 (1.1)	84.6 (2.5)	77.0 (1.5)	76.4 (2.5)
K=64	80.9 (1.5)	88.3 (1.5)	91.2 (0.7)	88.2 (0.7)	87.3 (0.8)	80.0 (2.0)	81.7 (1.4)
K=128	85.1 (1.0)	91.1 (0.7)	92.4 (0.7)	90.0 (0.7)	89.0 (0.8)	85.5 (1.3)	86.9 (1.5)
roberta-large							
K=0	68.7 (6.2)	61.7 (7.2)	59.8 (5.4)	60.1 (8.8)	54.6 (1.9)	56.1 (1.9)	50.4 (0.6)
K=4	78.1 (6.6)	69.4 (5.4)	74.0 (5.5)	71.6 (4.4)	67.6 (3.4)	64.1 (5.1)	56.9 (5.7)
K=8	83.0 (5.2)	78.5 (3.0)	84.4 (3.8)	77.0 (6.0)	75.3 (3.2)	71.3 (3.1)	64.2 (7.6)
K=16	87.5 (2.4)	84.4 (2.4)	87.6 (2.3)	83.4 (3.5)	82.8 (1.9)	76.2 (2.5)	76.4 (3.3)
K=32	89.9 (1.2)	87.3 (1.9)	92.3 (0.7)	88.9 (1.6)	87.7 (1.6)	80.8 (3.8)	81.6 (2.2)
K=64	90.8 (1.4)	90.4 (0.8)	92.6 (0.7)	90.9 (1.2)	90.1 (0.7)	84.1 (1.4)	86.2 (2.2)
K=128	93.0 (0.8)	92.9 (0.8)	93.4 (0.5)	92.2 (0.5)	91.0 (0.7)	88.4 (1.1)	90.2 (1.0)
full	97.5	95.4	97.8	98.7	98.1	95.8	98.8

Table 5.5: Results for the biMNLI, Atomic SI, and Complex SI tasks with each cell stating *"mean accuracy (standard deviation)"* except for majority vote and the fully supervised settings where standard deviation is not available.

Intra-SI Task Comparisons A notable trend is the relative difficulty of Complex SI compared to Atomic SI. For instance, at $K = 0$, roberta-large's performance on the Complex SI dataset of GO is nearly equivalent to the majority vote at 50.4%. At $K = 128$, roberta-large reaches an 88.4% accuracy on the Complex SI dataset of FoodOn, while surpassing 90% on other datasets. Among the Atomic SI tasks, the GO dataset poses the most significant challenge, aligning with expectations given GO's fine-grained, expert-level structure. Interestingly, despite DOID's domain-specific nature, it records an impressive 92.3% accuracy at $K = 32$, outperforming all other tasks at this sample size.

Domain-specific SI We extend our exploration to domain-specific LMs and their performance on domain-specific SI tasks. In this context, we evaluate roberta-large-pm-m3-voc,

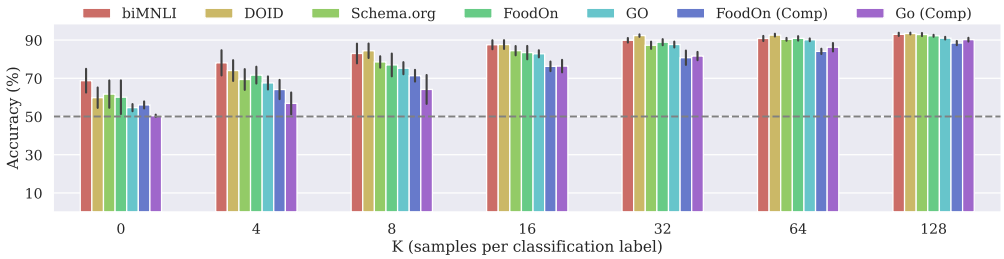


Figure 5.4: Visualisation of K-shot results (for roberta-large) on the biMNLI, Atomic SI, and Complex SI tasks, where the dotted horizontal line indicates majority vote.

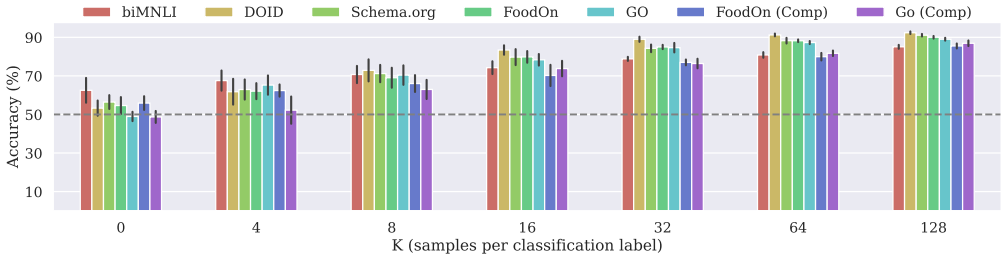


Figure 5.5: Visualisation of K-shot results (for roberta-base) on the biMNLI, Atomic SI, and Complex SI tasks, where the dotted horizontal line indicates majority vote.

a variant of RoBERTa fine-tuned on biomedical texts from PubMed abstracts, PMC full-text articles, and MIMIC-III clinical notes, with a vocabulary tailored to PubMed data (Lewis et al., 2020a). The K-shot performance of roberta-large-pm-m3-voc on the biomedical-oriented SI tasks for DOID and GO is detailed in Table 5.6.

Initially, the zero-shot performance of roberta-large-pm-m3-voc closely aligns with the majority vote baseline. However, as we introduce a small number of training samples (K increases), this domain-specific model exhibits a more pronounced performance improvement on the Atomic SI tasks for DOID and GO compared to the general roberta-large model. This suggests a heightened sensitivity of roberta-large-pm-m3-voc to the biomedical context within these tasks. Nevertheless, the GO Complex SI task presents a notable challenge for roberta-large-pm-m3-voc. For instance, at $K = 4$, there is no significant improvement compared to $K = 0$.

Templates and Label Words The choice of templates and label words significantly influences LM performance in zero-shot or few-shot settings. For instance, on the Atomic SI task of FoodOn, roberta-large exhibits a considerable standard deviation of

K	DOID	GO	GO (Comp)
0	49.7 (0.4)	50.1 (0.2)	50.0 (0.0)
4	64.8 (7.9)	66.2 (6.5)	50.0 (0.7)
32	94.7 (1.3)	93.5 (1.1)	73.5 (3.6)
128	96.3 (0.4)	95.2 (0.5)	90.5 (1.8)

Table 5.6: Results for roberta-large-pm-m3-voc on biomedical SI tasks.

8.8% at $K = 0$, indicating substantial variation in performance across different template and label word combinations. This variability underscores the importance of selecting an effective template and label words for the specific task and dataset. On the Complex SI task of GO, roberta-large demonstrates a low standard deviation of 0.6% at $K = 0$, but the corresponding accuracy of 50.4% suggests that none of the template-label word combinations are particularly effective. Moreover, the suitability of a template-label word combination appears to be model-specific. For example, roberta-large-pm-m3-voc underperforms in zero-shot settings on biomedical ontology SI tasks, implying that effective template and label word choices for one model may not necessarily transfer well to another.

5.7 Conclusion and Future Work

As a work that introduces ontologies to the "LMs-as-KBs" collection, this paper emphasises on how to establish a meaningful adaptation from logical expressions to natural language expressions while retaining their formal semantics. Utilising the NLI framework, we have crafted the SI task, carefully differentiating between textual and logical entailment. We also develop a recursive concept verbaliser for OWL ontologies as an auxiliary tool. Our results demonstrate that with our SI set-ups, LMs can successfully learn to infer both atomic and complex subsumptions when a small number of annotated samples are provided. This immediately helps ontology completion and futher paves the way for investigating more complex reasoning tasks with LMs or guiding LMs using ontology semantics with limited data. Moreover, while our SI approach is promising, alternative methods for probing ontological subsumption knowledge could be explored. For instance, directly framing $C \sqsubseteq D$ as "$\mathcal{V}(C)$ is a kind of $\mathcal{V}(D)$" and approaching it as

a fact-checking or inference task with no premises could offer different insights, despite our pilot experiments suggesting less efficacy compared to the current SI methodology.

Several avenues for future exploration arise from our work:

- **Enhanced Verbalisation:** Our ontology verbaliser does not yet cover all possible complex concept patterns (e.g., cardinality restrictions, nominals, etc.). We have integrated this tool into `DeepOnto`[7] and upgraded it to support more logical expressions.
- **More Textual Information:** Many ontologies include rich textual information such as synonyms, definitions, and comments. Integrating these textual cues could provide more context to enhance the LM's prediction.
- **Beyond TBox:** We have focused on TBox (terminological) axioms, but incorporating ABox (assertional) axioms could lead to new tasks like membership prediction, determining the conceptual category to which an individual belongs.
- **Training LMs with Ontological Knowledge:** A future direction could involve training LMs directly with ontologies, taking their logical semantics into account. Such an LM could be more directly applicable to ontology engineering tasks and potentially require fewer fine-tuning samples for adaptation.

[7] `https://krr-oxford.github.io/DeepOnto/deeponto/onto/verbalisation/`

CHAPTER 6

Language Models for Hierarchy Embedding

Contents

This chapter mainly discusses how to explicitly encode hierarchies with transformer encoder-based language models (LMs). We demonstrate effective hierarchy understanding of the resulting models, which we term Hierarchy Transformer encoders (HITs), through tasks designed to evaluate the capability of generalising from asserted subsumptions to inferred and unseen ones. The HIT models are expected to serve as more viable options of pre-trained LMs in a diverse array of hierarchy-aware applications.

6.1 Overview

Despite the proven efficacy of transformer-based LMs, a key limitation of these models is to interpret hierarchical structures latent in language and other complex symbolic data. This limitation has been highlighted by several studies, including those by Hanna and Mareček (2021) and He et al. (2023c), which employed prompt-based probes to reveal the limited hierarchical knowledge in pre-trained LMs, and the work by Lin and Ng (2022), which demonstrated these models' struggles with capturing the transitivity of hierarchical relationships.

Prior research has explored various methods to infuse hierarchical information into LM training. Common approaches include classification-based fine-tuning using sentence head embedding with a classification layer (Chen et al., 2023) or few-shot prompting with an answer mapping to classification labels (He et al., 2023c). To pre-train, or re-train[1] LMs on a corpus constructed from hierarchical data, Liu et al. (2020a) converted structural representations into textual formats to align the masked language modelling objective. Others, like Liu et al. (2021a) and Gosselin and Zouaq (2023), have focused on extracting analogous and contrasting examples from hierarchical structures for a similarity-based contrastive learning objective. The aforementioned studies leveraged hierarchical information as implicit signals to augment LMs, yet no existing works specifically targeted the explicit encoding of hierarchies with LMs.

To bridge this gap, we introduce a novel approach to re-train transformer encoder-based LMs as **Hi**erarchy **T**ransformer encoders (HITs). Inspired by the efficacy of hyperbolic geometry in representing hierarchical structures (Nickel and Kiela, 2017; Ganea et al., 2018a), we propose the hyperbolic clustering and centripetal losses tailored for LM re-training. As illustrated in Figure 6.1, transformer encoder-based LMs typically use a $\tanh$ activation function in the last layer, which maps each embedding dimension to the range $[-1, 1]$. Consequently, the output embeddings of LMs are confined within a unit d-dimensional hyper-cube. Leveraging this characteristic, we utilise a Poincaré

[1] The term *re-train* refers to further pre-train LMs on another corpus; it is distinguished from (standard) *fine-tune* as it does not modify the LM's architecture, while the latter often involves adding task-specific layers which lead to additional learnable parameters.

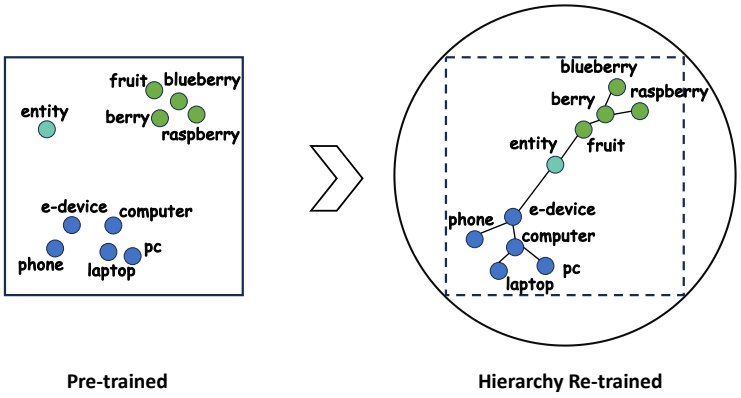


Figure 6.1: Illustration of how hierarchies are explicitly encoded in HIT after re-training. The square (d-dimensional hyper-cube) refers to the output embedding space of transformer encoder-based LMs whose final activation function is $\tanh$, and the circumscribed circle (d-dimensional hyper-sphere) refers to the Poincaré ball of radius $\sqrt{d}$. The distance and norm metrics involved in our re-training hyperbolic losses are defined w.r.t. this manifold.

ball of radius $\sqrt{d}$, whose boundary circumscribes[2] the output embedding space of LMs. The metrics for distance and norm used in our hyperbolic losses are defined w.r.t. this specific manifold. After re-training, entities are not only clustered according to their relatedness but also hierarchically organised.

In evaluating HITs, we compare their performance against both pre-trained and fine-tuned LMs, as well as typical hyperbolic approaches, in the Multi-hop Inference and Mixed-hop Prediction tasks. The Multi-hop Inference task, following the setting in Ganea et al. (2018a), involves training models on all asserted (i.e., one-hop) subsumptions and assessing their ability to infer transitive (i.e., multi-hop) subsumptions. The Mixed-hop Prediction task is designed to mirror real-world scenarios, where models trained on incomplete hierarchy are applied to predict unknown subsumption relationships between arbitrary entity pairs. Additionally, we introduce a transfer learning setting for the Mixed-hop Prediction task, where models trained on one hierarchy are tested on another. Our experiments utilise datasets derived from WordNet's noun hierarchy (Miller, 1995) and the SNOMED CT ontology (Stearns et al., 2001), and transfer evaluation datasets from Schema.org (Guha et al., 2016), Food Ontology (FoodOn) (Dooley et al., 2018), and Disease Ontology (DOID) (Schriml et al., 2012). The results show that HITs

[2]We ignore the vertices of the hyper-cube, as they are on the boundary and thus are undefined in the open Poincaré ball.

significantly surpass all baselines in these tasks, demonstrating their robustness to generalise from asserted to inferred and unseen subsumptions, and a promising potential in hierarchy-based semantic search.

6.2 Preliminaries

This section introduces preliminaries of hyperbolic geometry, which were not covered in Part II Foundations. Furthermore, we revisit the definition of hierarchies as outlined in Section 2.4.4 of Chapter 2.

6.2.1 Hyperbolic Geometry

Hyperbolic geometry, a kind of non-Euclidean geometry, is characterised by its constant negative Gaussian curvature, a fundamental aspect that differentiates it from the flat, zero curvature of Euclidean geometry. In hyperbolic space, distances between points increase exponentially as one moves towards the boundary, making it inherently suitable for embedding hierarchical structures. This intuition aligns with the tree embedding theorem based on δ-hyperbolicity, as discussed in Gromov (1987) and Ganea et al. (2018a).

Among the various models[3] of hyperbolic geometry that are isometric[4] to each other, the Poincaré ball is chosen for its capacity to contain the the output embedding space of LMs directly, as explained in the second last paragraph of Section 6.1.

Definition 6.2.1 (Poincaré Ball)

The d-dimensional Poincaré ball with a negative curvature $-c$ (where $c > 0$) is defined by the open ball $\mathbb{B}_c^d = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|^2 < \frac{1}{c}\}$. The Poincaré distance between two vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{B}_c^d$ is given by:

$$d_c(\mathbf{u}, \mathbf{v}) = \frac{2}{\sqrt{c}} \tanh^{-1}(\sqrt{c}\|-\mathbf{u} \oplus_c \mathbf{v}\|) \tag{6.1}$$

where $\|\cdot\|$ denotes the Euclidean norm, and $\oplus_c$ denotes the Möbius addition.

[3] E.g., the Poincaré ball model, the Poincaré half plane model, and the hyperboloid model.
[4] An isometry is a bijective distance-preserving transformation.

Definition 6.2.2 (Möbius Addition)

The Möbius addition of two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{B}_c^d$ is defined as:

$$\mathbf{u} \oplus_c \mathbf{v} = \frac{(1 + 2c\langle \mathbf{u}, \mathbf{v} \rangle + c\|\mathbf{v}\|^2)\mathbf{u} + (1 - c\|\mathbf{u}\|^2)\mathbf{v}}{1 + 2c\langle \mathbf{u}, \mathbf{v} \rangle + c^2\|\mathbf{u}\|^2\|\mathbf{v}\|^2} \tag{6.2}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product in Euclidean space.

Note that for flat curvature $c = 0$, $\mathbb{B}_c^d$ will be $\mathbb{R}^d$ and $\oplus_c$ will be the Euclidean addition.

6.2.2 Hierarchy

As presented in Definition 2.4.7, we define a hierarchy as a directed acyclic graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ represents the set of entities (nodes), and $\mathcal{E}$ represents the set of subsumption relationships (edges). The asserted subsumptions in $\mathcal{E}$ are direct (one-hop) edges, while the transitively inferred subsumptions are indirect (multi-hop) edges, denoted as $\mathcal{T}$.

This work adopts a closed-world assumption, under which a pair of entities (e_1, e_2) is classified as negative if it is not included in either $\mathcal{E}$ or $\mathcal{T}$. In this context, a *hard negative* arises when two entities, e_1 and e_2, are siblings, indicating that they share a common parent in the hierarchy, yet no direct or indirect subsumption relationship is established between them.

Explicit hierarchies can often be derived from structured data sources such as taxonomies, ontologies, and knowledge graphs. A taxonomy and the TBox of an ontology intrinsically define subsumptions, whereas in knowledge graphs, hierarchical relationships are defined in a more customised manner. For instance, in WordNet, the *hypernym* relationship corresponds to subsumption.

6.3 Related Work

Prompt-based probing is widely used for extracting knowledge from LMs. Studies like Hanna and Mareček (2021) utilised cloze-style prompts for hypernym detection, while He et al. (2023c) approached subsumption prediction similar to Natural Language Inference. Lin and Ng (2022) examined if LMs, when correctly predicting "A is a

B" and "B is a C", can consistently infer the transitive relationship "A is a C". These studies collectively highlight the limited capacity of pre-trained LMs in understanding hierarchical structures. Other research efforts, such as those by Liu et al. (2021a) and Gosselin and Zouaq (2023), have aimed to incorporate structural semantics into LMs for entity encoding. However, these largely focus on entity equivalence or similarity, with less emphasis on hierarchical organisation.

Regarding hyperbolic embeddings, methods like the Poincaré embeddings (Nickel and Kiela, 2017) and the hyperbolic entailment cone (Ganea et al., 2018a) have shown effectiveness in representing hierarchies. However, these techniques are inherently static, constrained by a fixed, pre-defined set of entities, and thus lacking inductive prediction capability. A promising solution to tackle this limitation, as suggested by our work in this chapter, is to integrate with pre-trained LMs, utilising their language processing ability to facilitate inductive predictions.

Moreover, there were explorations like Poincaré GloVe (Tifrea et al., 2018a) and hyperbolic text embeddings (Dhingra et al., 2018) that adapted word embeddings to hyperbolic spaces. However, these approaches, akin to Word2Vec's supersedence by transformer-based LMs, grapple with limitations due to their non-contextual word representations and inadequate handling of the out-of-vocabulary (OOV) problem.

Notably, Chen et al. (2020) stands out as a pioneering investigation into the integration of transformer-based LMs with hyperbolic losses. Their research incorporated learnable layers to project LM embeddings into hyperbolic spaces, facilitating the tasks of syntax parsing and sentiment analysis. Our methodology diverges from theirs by aiming to re-train LMs as general hierarchy encoders without the need for learnable parameters. Preserving the encoder nature of LMs is pivotal in maintaining the geometric interpretability of their embeddings. Our goal is to ensure that the hierarchy re-trained LMs can serve as new pre-trained models for a broad spectrum of hierarchy-aware tasks.

6.4 Hierarchy Transformer Encoder

We intend to propose a general and effective strategy to re-train transformer encoder-based LMs as Hierarchy Transformer encoders (HITs). To deal with arbitrary input

lengths of entity names, we employ an architecture similar to sentence transformers (Reimers and Gurevych, 2019), incorporating a mean pooling layer over token embeddings to produce sentence embeddings for entities. Note that some of the sentence transformer models have a normalisation layer after pooling; we exclude this layer because its presence will constrain the embeddings' Euclidean norms to one, thus hindering hierarchical organisation of entity embeddings. It is also worth mentioning that these changes do not add in learnable parameters besides the ones already in LMs, thus retaining the original architectures of LMs as encoders.

As aforementioned, the output embedding space of these LMs is typically a d-dimensional hyper-cube because of the $\tanh$ activation function in the last layer. Thus, we can construct a Poincaré ball of radius $\sqrt{d}$ (or equivalently, curvature value $c = \frac{1}{d}$) whose boundary circumscribes the hyper-cube. While an alternative approach of scaling each dimension of the LM embeddings by $\sqrt{d}$ to fit within a unit Poincaré ball was considered, this method proved challenging for loss convergence. Consequently, the output embedding space of LMs is a subset of this Poincaré ball if we exclude the vertices of the hyper-cube. Our empirical results (see Section 6.5) support the assumption that leveraging the full volume of the Poincaré ball is not necessary for capturing hierarchies in a high-dimensional space. Based on the constructed manifold, we propose the following two losses for hierarchy re-training.

Definition 6.4.1 (Hyperbolic Clustering Loss)

Given input data $\mathcal{D}$ consisting of triplets of the form (e, e^+, e^-), where e^+ is a parent of e, and e^- is a negative parent of e, the hyperbolic clustering loss is defined as:

$$\mathcal{L}_{\text{cluster}} = \sum_{(e,e^+,e^-)\in\mathcal{D}} \max(d_c(\mathbf{e}, \mathbf{e}^+) - d_c(\mathbf{e}, \mathbf{e}^-) + \alpha, 0) \tag{6.3}$$

where $d_c(\cdot, \cdot)$ refers to the hyperbolic distance function defined in equation 6.1, and α is the hyperbolic distance margin; the bold letters denote the embeddings of the corresponding entities.

This loss aims at clustering related entities and distancing unrelated ones in the Poincaré ball. We formulate it in the form of triplet loss (see further discussion in Section 3.3.4 of Chapter 3) because related entities are not equivalent but their semantic distances should be smaller than those between unrelated entities.

Definition 6.4.2 (Hyperbolic Centripetal Loss)

Given input data $\mathcal{D}$ consisting of triplets of the form (e, e^+, e^-), where e^+ is a parent of e, and e^- is a negative parent of e, the hyperbolic centripetal loss is defined as:

$$\mathcal{L}_{\text{centri}} = \sum_{(e,e^+,e^-)\in\mathcal{D}} \max(\|\mathbf{e}^+\|_c - \|\mathbf{e}\|_c + \beta, 0) \tag{6.4}$$

where $\|\cdot\|_c := d_c(\cdot, 0)$ refers to the hyperbolic norm and β is the hyperbolic norm margin.

This loss ensures parent entities are positioned closer to the Poincaré ball's origin than their child counterparts, reflecting the natural expansion of hierarchies from the origin to the boundary of the manifold. The term *centripetal* is used to imply that the manifold's origin represents an **imaginary root entity** for everything. Notice that only the child and parent entities (the positive subsumptions) are used to calculate this loss.

Definition 6.4.3 (Hierarchy Re-training Loss)

The overall hierarchy re-training loss is defined as:

$$\mathcal{L}_{\text{HiT}} = \mathcal{L}_{\text{cluster}} + \mathcal{L}_{\text{centri}} \tag{6.5}$$

where $\mathcal{L}_{\text{cluster}}$ and $\mathcal{L}_{\text{centri}}$ are defined above.

In Figure 6.2, we demonstrate the impact of $\mathcal{L}_{\text{HiT}}$ on entity embeddings. The entity E-device, being most general, is nearest to the origin. Sibling entities, such as Phone and Computer, Laptop and PC, are closer to their common parent than to each

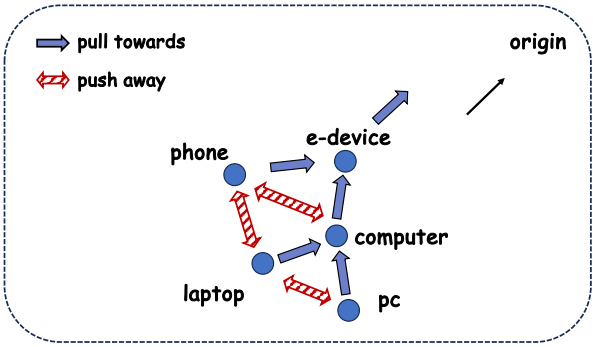


Figure 6.2: Illustration of the impact of hyperbolic clustering and centripetal losses during training. In Euclidean space, it seems contradictory that both Phone and Computer are pulled towards E-device but are also pushed away from each other. However, in principle this is not a problem in hyperbolic space, where distances increase exponentially relative to Euclidean distances as one moves from the origin to the boundary of the manifold.

other, illustrating the effect of re-training to cluster related entities while maintain hierarchical relationships.

As the HIT model functions as an encoder like pre-trained LMs, it does not inherently support direct predictions of subsumption relationships. To address this, we devise an empirical scoring function that probes the hierarchy re-trained entity embeddings. This function aims to predict the subsumption relationship for any given pair of entities (e_1, e_2), incorporating both the clustering and centripetal heuristics:

Definition 6.4.4 (Subsumption Prediction for HIT)

The subsumption prediction probing function for HIT models is defined as:

$$s(e_1 \sqsubseteq e_2) = -(d_c(\mathbf{e}_1, \mathbf{e}_2) + \lambda(\|\mathbf{e}_2\|_c - \|\mathbf{e}_1\|_c)) \tag{6.6}$$

where $\lambda > 0$ represents a weighting factor applied to the centripetal heuristic component.

This subsumption score is structured to increase as the hyperbolic distance between $\mathbf{e}_1$ and $\mathbf{e}_2$ decreases, and/or as the relative difference in their hyperbolic norms ($\|\mathbf{e}_1\|_c - \|\mathbf{e}_2\|_c$) increases. Essentially, for the model to predict $e_1 \sqsubseteq e_2$, it is expected that $\mathbf{e}_1$ and $\mathbf{e}_2$ are relatively closer in the Poincaré ball, with $\mathbf{e}_1$ positioned further from the

manifold's origin compared to $\mathbf{e}_2$. The value for λ and the overall scoring threshold are to be ascertained through hyperparameter tuning on the validation set.

6.5 Evaluation

6.5.1 Task Formulation

In our evaluation, we propose the following two tasks to assess a model's ability to generalise from asserted to inferred and unseen subsumptions.

Multi-hop Inference This task, following the setting in Ganea et al. (2018a), aims to evaluate the model's ability in deducing indirect, multi-hop subsumptions $\mathcal{T}$ from direct, one-hop subsumptions $\mathcal{E}$, so as to simulate transitive inference. We split $\mathcal{T}$ for validation and testing, denoted as $\mathcal{T}_{val}$ and $\mathcal{T}_{test}$, respectively. For each positive subsumption (e, e^+) involved, we sampled 10 negative parents e^- for e, leading to 10 training triplets[5]. Following the criteria in Section 6.2.2, (e, e^-) is a valid negative if $(e, e^-) \notin \mathcal{E} \cup \mathcal{T}$. We further split the task into two settings: one with **random negatives** and another with **hard negatives**, the latter mainly comprising sibling entities. Since not every entity has enough siblings, we supplemented with random negatives that have been sampled to maintain a consistent positive-to-negative ratio of $1:10$. The percentages of true hard negatives are also reported in our data statistics table at the end of this section.

Mixed-hop Prediction This task aims to evaluate the model's capability in determining the existence of subsumption relationships between arbitrary entity pairs, where the entities are not necessarily seen during training. We propose a challenging setting where models are trained on incomplete direct subsumptions and examined on a mix of hold-out, unseen direct and indirect (mixed-hop) subsumptions. We split $\mathcal{E}$ into training, validation, and test sets, denoted as $\mathcal{E}_{train}$, $\mathcal{E}_{val}$, and $\mathcal{E}_{test}$, respectively. The final training, validataion, and test sets for this task are $\mathcal{E}_{train}$, $\mathcal{E}_{val} \cup \mathcal{T}_{val}$, and $\mathcal{E}_{test} \cup \mathcal{T}_{test}$, respectively, where $\mathcal{T}_{val}$ and $\mathcal{T}_{test}$ are re-used from the previous task. Again, each positive subsumption in these sets is paired with 10 negative samples, either randomly chosen or from sibling entities. Furthermore, an important factor that reflects the model's

[5] 10 training triplets are constructed from 11 entity pairs.

generalisability is to examine the **transfer ability across hierarchies**. To this end, we extend the mixed-hop prediction task with a transfer setting where models trained on asserted training edges of one hierarchy are tested on arbitrary entity pairs of another.

Evaluation Metrics For both Multi-hop Inference and Mixed-hop Prediction tasks, we utilise Precision, Recall, and F1 score (abbreviated as F-score in latter discussion) as our primary metrics of evaluation. We have opted not to include Accuracy, as preliminary testing indicated a potential bias in this metric, with a misleadingly high score resulting from the much larger volume of negative compared to positive samples. It is important to note that, although the training phase uses entity triplets, the evaluation only involves entity pairs.

6.5.2 Dataset Construction

Dataset Sources We select WordNet (Miller, 1995) and SNOMED CT (Stearns et al., 2001) as the primary hierarchies for our models to train on. WordNet is chosen for its comprehensive and well-structured representation of linguistic hierarchies. SNOMED CT, on the other hand, is a widely recognised knowledge base in the biomedical field, offering a rich taxonomy of medical and clinical terms.

To evaluate the transferability and robustness of our models across different domains, we consider datasets derived from ontologies that demonstrate varied semantic granularities and cover diverse fields. These include Schema.org (Guha et al., 2016), which provides schemas for structured data on the Internet, Food Ontology (FoodOn) (Dooley et al., 2018), which details food-related terms, and Disease Ontology (DOID) (Schriml et al., 2012), which categorises human diseases.

Preprocessing We retrieve WordNet from `NLTK`[6] and adopted pre-processing steps similar to Nickel and Kiela (2017), utilising the *hypernym* relations between noun *synsets* to construct the hierarchy. Using the tool provided by the SNOMED CT team,[7] we convert the latest version (released in December 2023) of SNOMED CT into the OWL ontology format. For Schema.org, FoodOn, and DOID, our pre-processing parallel that in

[6] `https://www.nltk.org/`
[7] `https://github.com/IHTSDO/snomed-owl-toolkit`

He et al. (2023c) (also introduced in Section 5.5 of Chapter 5). We use the pre-processed versions of FoodOn and DOID and apply the same pre-processing methodology to the latest version of Schema.org. Careful considerations are required for constructing named entity hierarchies from ontologies due to the existence of complex concepts. To maintain the flow of reading, further specifics on these preprocessing steps are provided in Appendix 6.A.

Entity Lexicon To accommodate the textual input requirements of LMs, we construct an entity lexicon using the `name` attribute in WordNet and the `rdfs:label` property in the ontologies, selecting the first English label if multiple labels for one entity are available. For the entity lexicon of SNOMED CT, we address potential information leakage during testing. Typically, an SNOMED CT entity is named in the format of "[EntityName] ([BranchName])", e.g., "virus (organism)". The [BranchName] denotes the top ancestor and is propagated through all its descendants. To prevent this information from biasing the model, we remove the branch name from each entity.

Dataset Splitting On WordNet (Noun), SNOMED CT, FoodOn, and DOID, we adopt a consistent splitting ratio for the validation and testing sets. Specifically, we allocate two separate 5% portions of the indirect subsumptions $\mathcal{T}$ to form $\mathcal{T}_{val}$ and $\mathcal{T}_{test}$, respectively. Similarly, two distinct 5% portions of the direct subsumptions $\mathcal{E}$ are used as $\mathcal{E}_{val}$ and $\mathcal{E}_{test}$. As Schema.org is significantly smaller than the other hierarchies and only used for transfer evaluation, we split its entire $\mathcal{E}$ and $\mathcal{T}$ sets into halves for validation and testing, respectively.

Table 6.1 presents the extracted hierarchies' statistics and the resulting datasets for the Multi-hop Inference and Mixed-hop Prediction tasks. Note that the Multi-hop Inference follows previous evaluation settings in Ganea et al. (2018a) and the dataset is constructed only on WordNet in order to compare with previous static hyperbolic embedding approaches.

Source	#Entity	#Sub	#ISub	#Dataset (Train/Val/Test)
WordNet	74,401	75,850	587,658	multi: 834K (64.8%)/323K (62.3%)/323K (61.8%) mixed: 751K (64.7%)/365K (62.7 %)/365K (62.2%)
Schema.org	903	950	1,978	mixed: -/15K (65.6%)/15K (64.9%)
FoodOn	30,963	36,486	438,266	mixed: 361K (67.0%)/261K (62.1%)/261K (62.2%)
DOID	11,157	11,180	45,383	mixed: 122K (63.6%)/31K (59.1%)/31K (60.8%)
SNOMED	364,352	420,193	2,775,696	mixed: 4,160K (79.5%)/1,758K (65.3%)/1,758K (65.2%)

Table 6.1: Statistics of WordNet (Noun), SNOMED CT, Schema.org, and FoodOn, including the numbers of entities (**#Entity**), direct subsumptions (**#Sub**), indirect subsumptions (**#ISub**), and the dataset splittings (**#Dataset**) for Multi-hop Inference and Mixed-hop Prediction tasks. Note that the numbers in **#Dataset** are counts of entity pairs rather than entity triplets, and the percentages in the parentheses indicate the portions of real hard negatives in the hard negative setting.

6.5.3 Baselines

Naive Prior We first introduce a naive baseline that utilises the prior probability of positive subsumptions in the training set for prediction. Given that each positive sample is paired with 10 negatives, the prior probability of a positive prediction stands at $\frac{1}{11}$. Consequently, Precision, Recall, and F-score on the test set are all $\frac{1}{11}$.

Pre-trained LMs We consider pre-trained LMs as baselines to illustrate their limitations in capturing hierarchical structure semantics. The probe for pre-trained LMs typically needs to follow their pre-training objectives for optimal performance. Since in this work we focus on LMs based on the sentence transformer architecture (Reimers and Gurevych, 2019), which are primarily pre-trained on sentence pairs using cosine similarity losses, we devise the following probe for evaluation: for each entity pair (e_1, e_2), we compute the cosine similarity between the masked reference sentence "e_1 is a $[MASK]$." and the sample sentence "e_1 is a e_2.". These similarity scores serve as the subsumption scores, with thresholds identified via grid search on the validation set. Note that although these LMs originate from masked language models, they cannot be easily probed via mask filling logits or perplexities as in Petroni et al. (2019b) and Salazar et al. (2020) because their mask filling layers are not preserved in the released versions.

The pre-trained LMs selected from the sentence transformer library include all-MiniLM-L6-v2, all-MiniLM-L12-v2, and all-mpnet-base-v2, each re-trained on 1B sentence

pairs from a spectrum of NLP tasks.[8] The all-MiniLM models, derived from the MiniLM distilled from BERT (Wang et al., 2020), are noted for their efficiency and performance, with all-MiniLM-L6-v2 and all-MiniLM-L12-v2 being the variants of 6 and 12 layers, respectively. The embedding dimension of them is 384. The all-mpnet-base-v2 model, based on MPNet (Song et al., 2020), is selected for its top performance in the sentence transformer collection[9]. It has an embedding dimension of 768. While the selected models are not specifically designed for hierarchy re-training, their optimisation for clustering and semantic search tasks makes them suitable initial checkpoints.

Fine-tuned LMs Fine-tuned LMs are used as baselines to demonstrate that despite the efficacy in various tasks, standard fine-tuning struggles to address this specific challenge. Following the BERTSubs approach outlined in Chen et al. (2023), we employ pre-trained LMs with an added linear layer for binary classification, and optimising on the Softmax loss. Chen et al. (2023) have shown that this method outperforms various structure-based embedding techniques, such as TransE (Bordes et al., 2013) and DistMult (Yang et al., 2014), and also surpasses OWL2Vec* (Chen et al., 2021a), which integrates both structural and textual embeddings, in the task of predicting subsumptions.

Hyperbolic Approaches Previous static hyperbolic embedding models are typically evaluated using the Multi-hop Inference task. In our study, we select the Poincaré Embedding (PoincaréEmbed) (Nickel and Kiela, 2017) and the Hyperbolic Entailment Cone (HyperbolicCone) (Ganea et al., 2018a) as baselines on this task. However, their lack of inductive prediction capabilities prevents their evaluation on the Mixed-hop Prediction task and its transfer setting. Additionally, we include the hyperbolic GloVe embedding (PoincaréGloVe) as a baseline in our transfer evaluation. We select the best-performing PoincaréGloVe (50×2D with an initial trick) pre-trained on a 1.4B token English Wikipedia dump. While PoincaréGloVe supports inductive prediction, its effectiveness is limited by word-level tokenisation, rendering it less effective at handling unknown words. To address this, we employ NaivePrior as a fallback method for entities that involve unknown words and cannot be predicted by PoincaréGloVe.

[8] https://discuss.huggingface.co/t/train-the-best-sentence-embedding-model-ever-with-1b-training-pairs/7354

[9] https://www.sbert.net/docs/pretrained_models.html

6.5.4 Implementation

The code implementation of this work primarily depends on `DeepOnto` (He et al., 2023b) for processing hierarchies and constructing datasets, `Geoopt` (Kochurov et al., 2020) for Poincaré ball, `Sentence-Transformers` (Reimers and Gurevych, 2019) and `Huggingface-Transformers` (Wolf et al., 2020) for training and evaluation of LMs. All our experiments are conducted on a single Quadro RTX 8000 GPU.

In the hierarchy re-training of our HiT models, we configure the hyperbolic clustering loss margin (α in Equation 6.3) at 5.0 and the hyperbolic centripetal loss margin (β in Equation 6.4) at 0.1. An exception is made for all-mpnet-base-v2 with hard negatives, where α is adjusted to 3.0, based on validation. The models are trained for 20 epochs, with a training batch size of 256, 500 warm-up steps and an initial learning rate of 10^{-5}, using the AdamW optimiser (Loshchilov and Hutter, 2018). Model selection is conducted after each epoch, guided by performance on the validation set.

For standard fine-tuning, we largely adhere to the standard settings of the Huggingface Trainer[10], but maintain the same training batch size as used in the hierarchy re-training. Notably, our preliminary testing reveals that fine-tuning is more prone to overfitting. Consequently, we adopt a more frequent model selection interval, performing this assessment every 500 training steps rather than epoch-wise.

For our hyperbolic baselines, we trained PoincaréEmbed with an embedding dimension of 200 for 200 epochs, a training batch size of 256, 10 warm-up epochs, and a constant learning rate of 0.01, using the Riemannian Adam optimiser (Becigneul and Ganea, 2018). According to (Ganea et al., 2018a), HyperbolicCone benefits from a robust initialisation such as the one provided by a pre-trained PoincaréEmbed. Consequently, we utilised the same hyperparameters to train HyperbolicCone except that we initialised it with the weights from our pre-trained PoincaréEmbed. As outlined in the main paper, we selected the optimal pre-trained PoincaréGloVe model reported by (Tifrea et al., 2018b) (50$\times$2D with an initial trick) as a baseline for our transfer evaluation.

[10] `https://huggingface.co/docs/transformers/main_classes/trainer`

	Random Negatives			Hard Negatives		
Model	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
NaivePrior	0.091	0.091	0.091	0.091	0.091	0.091
Multi-hop Inference (WordNet)						
PoincaréEmbed	0.862	0.866	0.864	0.797	0.867	0.830
HyperbolicCone	0.817	0.996	0.898	0.243	0.902	0.383
all-MiniLM-L6-v2	0.160	0.442	0.235	0.132	0.507	0.209
+ fine-tune	0.800	0.513	0.625	0.764	0.597	0.670
+ HiT	0.864	0.879	0.871	0.905	0.908	0.907
all-MiniLM-L12-v2	0.127	0.585	0.209	0.108	0.740	0.188
+ fine-tune	0.811	0.515	0.630	0.819	0.530	0.643
+ HiT	0.880	0.927	0.903	0.910	0.906	0.908
all-mpnet-base-v2	0.281	0.428	0.339	0.183	0.359	0.242
+ fine-tune	0.796	0.501	0.615	0.758	0.628	0.687
+ HiT	0.897	0.936	0.916	0.886	0.912	0.899
Mixed-hop Prediction (WordNet)						
all-MiniLM-L6-v2	0.160	0.438	0.235	0.131	0.504	0.208
+ fine-tune	0.747	0.575	0.650	0.769	0.578	0.660
+ HiT	0.835	0.877	0.856	0.882	0.843	0.862
all-MiniLM-L12-v2	0.127	0.583	0.209	0.111	0.625	0.188
+ fine-tune	0.794	0.517	0.627	0.859	0.515	0.644
+ HiT	0.875	0.895	0.885	0.886	0.857	0.871
all-mpnet-base-v2	0.287	0.439	0.347	0.197	0.344	0.250
+ fine-tune	0.828	0.536	0.651	0.723	0.622	0.669
+ HiT	0.892	0.910	0.900	0.869	0.858	0.863

Table 6.2: Multi-hop Inference and Mixed-hop Prediction test results on WordNet.

6.5.5 Results on WordNet

The effectiveness of our hierarchy re-training approach is evident from the results of the Multi-hop Inference and Mixed-hop Prediction tasks on WordNet (see Table 6.2), and the transfer Mixed-hop Prediction task on Schema.org, FoodOn, and DOID for pre-trained models and models trained on WordNet (see Table 6.3). In the following, we present several pivotal findings based on these results.

Performance of HiTs The HiT models, re-trained from LMs of various sizes, consistently outperform their pre-trained and fine-tuned counterparts across all evaluation tasks.

	Random Negatives			Hard Negatives		
Model	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
NaivePrior	0.091	0.091	0.091	0.091	0.091	0.091
Transfer Mixed-hop Prediction (WordNet → Schema.org)						
PoincaréGloVe	0.485	0.403	0.441	0.436	0.415	0.425
all-MiniLM-L12-v2	0.312	0.524	0.391	0.248	0.494	0.330
+ fine-tune	0.391	0.433	0.411	0.597	0.248	0.351
+ HiT	0.503	0.613	0.553	0.408	0.583	0.480
Transfer Mixed-hop Prediction (WordNet → FoodOn)						
PoincaréGloVe	0.192	0.224	0.207	0.189	0.200	0.195
all-MiniLM-L12-v2	0.135	0.656	0.224	0.099	0.833	0.176
+ fine-tune	0.436	0.382	0.407	0.690	0.177	0.282
+ HiT	0.690	0.463	0.554	0.741	0.385	0.507
Transfer Mixed-hop Prediction (WordNet → DOID)						
PoincaréGloVe	0.265	0.314	0.287	0.283	0.318	0.299
all-MiniLM-L12-v2	0.342	0.451	0.389	0.159	0.455	0.235
+ fine-tune	0.585	0.621	0.603	0.868	0.179	0.297
+ HiT	0.696	0.711	0.704	0.810	0.435	0.566

Table 6.3: Transfer Mixed-hop Prediction test results on Schema.org, FoodOn, and DOID.

In the Multi-hop Inference task, HiTs exhibit exceptional performance with F-scores ranging from 0.871 to 0.916. This indicates a strong capability in generalising from asserted to transitively inferred entity subsumptions. In the Mixed-hop Prediction task, F-scores ranging from 0.856 to 0.900 highlight the effectiveness of HiTs in generalising from asserted to arbitrary entity subsumptions.

For the Transfer Mixed-hop Prediction tasks, we selected all-MiniLM-L12-v2 as the pre-trained model because all-MiniLM-L12-v2+HiT attains comparable performance to all-mpnet-base-v2+HiT while it is more computationally efficient owing to a smaller parameter size. Notably, all-MiniLM-L12-v2+HiT performs better than pre-trained and fine-tuned all-MiniLM-L12-v2 on these transfer tasks by at least 0.150 and 0.101 in F-scores, respectively.

Limited Hierarchical Knowledge in Pre-trained LMs For the tasks on WordNet, all-mpnet-base-v2 achieves the highest F-scores among all the pre-trained models, yet

these scores (e.g., 0.347 and 0.250 on the Mixed-hop Prediction task with random negatives and hard negatives, respectively) are considerably lower compared to their fine-tuned (lagging by 0.304 and 0.419) and hierarchy re-trained (lagging by 0.553 and 0.613) counterparts. This disparity confirms findings from LM probing studies such as those by Hanna and Mareček (2021) and He et al. (2023c), demonstrating the limited hierarchical knowledge in pre-trained LMs.

Limited Generalisation in Fine-tuned LMs The research by Chen et al. (2023) illustrates that fine-tuned LMs perform well on single-hop subsumptions. Our observations concur, showing that fine-tuned LMs achieve comparable performance as HITs when assessed on just single-hop test samples. However, their effectiveness wanes when applied to arbitrary entity subsumptions. For the tasks on WordNet, fine-tuned LMs underperform HITs by 0.194 to 0.301 in F-scores. In the transfer task from WordNet to Schema.org, the fine-tuned all-MiniLM-L12-v2 model only marginally outperforms its initial state, with an increase of around 0.02 in F-scores across both negative settings.

Performance of Hyperbolic Baselines The Multi-hop Inference task with random negatives follows the evaluation in (Ganea et al., 2018a). In this setup, both PoincaréEmbed and HyperbolicCone significantly outperform the pre-trained and standard fine-tuned LMs, and perform comparably to the HIT models. However, HyperbolicCone exhibits substantially worse performance in the hard negative setting; its low precision and high recall suggest difficulties in differentiating sibling entities that are closely positioned in the embedding space. In the transfer evaluation, PoincaréGloVe shows improved performance over pre-trained and standard fine-tuned models on Schema.org. However, it does not demonstrate a similar advantage on FoodOn and DOID, primarily due to its limited vocabulary, which allows it to predict almost all test samples on Schema.org but substantially fewer on the others.

A crucial advantage of HIT over these hyperbolic approaches is its capability for inductive prediction, enabling its application in tasks like Mixed-hop Prediction and transfer learning scenarios, where previous hyperbolic embedding methods are not applicable due to their static vocabulary for entities.

Comparison of Random and Hard Negatives For the tasks on WordNet, hard negative settings present greater challenges compared to random negative settings for all pre-trained LMs. This increased difficulty, however, is not as pronounced in fine-tuned LMs and HITs. A plausible explanation is that while hard negatives pose challenges, they simultaneously act as high-quality adversarial examples, potentially leading to more robust training outcomes. In the Transfer Mixed Prediction task, hard negative settings are generally more challenging than random negative settings. For instance, in the WordNet-to-DOID transfer task, both fine-tuned and hierarchy re-trained all-MiniLM-L12-v2 models exhibit significantly higher F-scores in the random negative setting, with differences of 0.306 and 0.138 respectively, compared to the hard negative setting.

Case Analysis on WordNet-to-DOID Transfer In the WordNet-to-DOID transfer task, the disparity in the Disease category is notable: WordNet contains only 605 entities that are descendants of Disease, compared to over $10K$ in DOID. Despite this significant difference, HIT models effectively transfer knowledge, achieving F-scores of 0.704 (random negatives) and 0.566 (hard negatives).

6.5.6 Results on SNOMED CT

Table 6.4 details the Mixed-hop Prediction results on SNOMED CT, along with the Transfer Mixed-hop Prediction results on Schema.org, FoodOn, and DOID for pre-trained LMs and models trained on SNOMED CT. Building on the demonstrated effectiveness of HIT in simulating transitive inference from the previous section, we present only the results for inductive subsumption prediction. The transfer evaluation incorporates the same set of hierarchies as those used in the evaluation on WordNet (Noun), facilitating a meaningful comparison of model performance across different training hierarchies.

In the Mixed-hop Prediction task on SNOMED CT, all models – pre-trained, fine-tuned, and hierarchy re-trained – outperform their counterparts on WordNet (Noun) in terms of F-scores. This enhanced performance is likely attributed to SNOMED CT's more specific entity naming, reducing homonyms and thus ambiguity. In the transfer results, models trained on SNOMED CT exhibit notably better F-scores on DOID, which aligns with expectations given DOID's focus on diseases and SNOMED CT's comprehensive

	Random Negatives			Hard Negatives		
Model	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
MajorityPrior	0.091	0.091	0.091	0.091	0.091	0.091
Mixed-hop Prediction (SNOMED)						
all-MiniLM-L12-v2	0.224	0.443	0.297	0.145	0.398	0.213
+ fine-tune	0.919	0.859	0.888	0.894	0.635	0.743
+ HɪT	0.941	0.967	0.954	0.905	0.894	0.899
Transfer Mixed-hop Prediction (SNOMED → Schema.org)						
all-MiniLM-L12-v2	0.312	0.524	0.391	0.248	0.494	0.330
+ fine-tune	0.198	0.864	0.322	0.431	0.288	0.345
+ HɪT	0.432	0.580	0.495	0.274	0.608	0.378
Transfer Mixed-hop Prediction (SNOMED → FoodOn)						
all-MiniLM-L12-v2	0.135	0.656	0.224	0.099	0.833	0.176
+ fine-tune	0.378	0.540	0.445	0.638	0.371	0.469
+ HɪT	0.700	0.500	0.583	0.594	0.442	0.506
Transfer Mixed-hop Prediction (SNOMED → DOID)						
all-MiniLM-L12-v2	0.342	0.451	0.389	0.159	0.455	0.235
+ fine-tune	0.547	0.912	0.684	0.831	0.795	0.812
+ HɪT	0.836	0.864	0.850	0.739	0.748	0.744

Table 6.4: Mixed-hop Prediction test results on SNOMED, comparing LMs of different sizes and different settings (pre-trained, fine-tuned, and hierarchy re-trained), and Transfer Mixed-hop Prediction results on Schema.org, FoodOn, and DOID.

biomedical scope. Conversely, their performance on Schema.org, a common-sense ontology, is comparatively worse. Notably, the fine-tuned all-MiniLM-L12-v2 performs even worse than its pre-trained version on Schema.org, suggesting an overfitting to biomedical domain knowledge in SNOMED CT.

6.5.7 Analysis of HɪT Embeddings

Distribution Figure 6.3 illustrates the distribution of WordNet entity embeddings generated by HɪT w.r.t. their hyperbolic norms. These norms effectively capture the natural expansion of the hierarchical structure, evidenced by an exponential rise in the number of child entities. A notable observation is the sharp decline in the number of entities when hyperbolic norms exceed 23, suggesting that few entities reside at these

higher levels. Additionally, the range of entity hyperbolic norms, approximately from 8 to 24, indicates that a relatively small region of the high-dimensional manifold suffices to accommodate all entities in WordNet, conforming our assumption stated in Section 6.4.

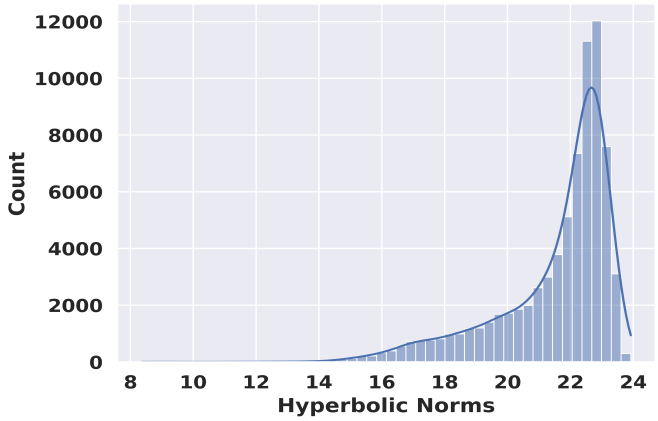


Figure 6.3: Visualisation of the distribution of WordNet entity embeddings generated by HiT (re-trained from all-MiniLM-L12-v2) w.r.t. their hyperbolic norms.

Correlation In Table 6.5, we further compare the Pearson correlation coefficients across different hyperbolic models that measures the linear relationship between entities' hyperbolic norms and their depths[11] We can observe that all the hyperbolic models lead to a positive correlation between norms and depths as expected, but HiT presents a stronger correlation than both PoincaréEmbed and HyperbolicCone.

HiT	PoincaréEmbed	HyperbolicCone
0.346	0.130	0.245

Table 6.5: Statistical correlations between WordNet entities' depths and their hyperbolic norms in HiT, PoincaréEmbed, and HyperbolicCone, respectively.

Case Study In Table 6.6, we showcase the effectiveness of HiT using selected entities: Computer, PC[12], Fruit, and Berry). The table presents the hyperbolic distances between these entities' embeddings, their individual hyperbolic norms, and their depths in the WordNet hierarchy. We can observe that: *(i)* closely related entities, such as Fruit and

[11]Depth of an entity is the minimum number of hops to the root node. For hierarchies that do not have a root node, we set up an imaginary root node when calculating the depth. in the WordNet hierarchy.

[12]The full name "personal computer" is used for embedding.

	Computer	PC	Fruit	Berry
Computer	0.0	5.9	22.5	24.9
PC	5.9	0.0	25.2	27.2
Fruit	22.5	25.2	0.0	6.72
Berry	24.9	27.2	6.72	0.0
h-norm	17.5	19.1	15.3	16.6
depth	9	11	9	10

Table 6.6: Hyperbolic distances between the embeddings of selected entities (Computer, PC, Fruit, Berry), along with their individual hyperbolic norms (**h-norm**) and depths in WordNet.

Berry, are significantly nearer to each other compared to more distant pairs; *(ii)* more specific entities like PC and Berry are positioned further from the origin of the manifold than their ancestor entities; *(iii)* the disparity in hyperbolic norms between PC and Computer is greater compared to that between Fruit and Berry, reflecting the hierarchical depth where PC is a grandchild of Computer, while Berry is a direct child of Fruit.

6.6 Conclusion and Future Work

This work tackles the challenge of enabling language models to interpret and encode hierarchies. We devise the hierarchy re-training approach that involves a joint optimisation on both the hyperbolic clustering and hyperbolic centripetal losses, aiming to cluster and organise entities according to their hierarchical relationships. The resulting HyT models demonstrate proficiency in simulating transitive inference and predicting subsumptions within and across hierarchies. Additionally, the analysis of HyT embeddings highlights their interpretability in geometric space.

Looking forward, we have identified several directions for expanding the capabilities and applications of HyT models:

- **Hierarchy-based Semantic Search:** A promising direction is to enhance current semantic search methodologies, which primarily focus on similarity, to include hierarchy-based queries. This would enable searches that specifically target parent or child entities.

- **Fine-tuning for HyT Models:** Drawing inspiration from the custom fine-tuning methods used in sentence transformers (Reimers and Gurevych, 2019), we plan

to explore hyperbolic entity embedding features for fine-tuning HiT models. This adaptation could potentially replace the empirical subsumption prediction function outlined in Equation 6.6.

- **Addressing Catastrophic Forgetting:** To enhance the versatility of HiT models for broader applications, we need to address the issue of catastrophic forgetting – where a model loses previously learned information upon training for new tasks. As our current approach does not modify the architecture of LMs, it is possible to explore *continual pre-training* (Ke et al., 2022) strategies for retaining knowledge across multiple learning phases.

- **Multi-relations:** While this research focuses on singular hierarchical relationships, real-world data often involve multiple, and sometimes complex, inter-entity relationships. Future work will delve into the integration of multi-relations to further refine and advance HiT model capabilities.

6.A Ontology Taxonomy

As introduced in Chapter 2, the TBox of an OWL ontology defines entity relationships using subsumption axioms ($C \sqsubseteq D$) and equivalence axioms ($C \equiv D$), where C and D are atomic or complex concept expressions defined in the description logic $\mathcal{SROIQ}$. The concept hierarchy, or *ontology taxonomy*, is defined as a directed acyclic graph with nodes representing named atomic concepts and edges representing direct subsumption relationships between them. To avoid misplacing implicit direct child of certain concepts under the root node `owl:Thing`, we utilised an ontology reasoner to deduce *direct*[13] subsumptions and included these as edges in the hierarchy. This approach fully considers both subsumption and equivalence axioms for all potential edges. Considering an atomic concept Beef, which is defined by $\mathsf{Beef} \equiv \mathsf{Meat} \sqcap \exists \mathsf{derivesFrom}.\mathsf{Cattle}$, as an example, the reasoning will lead to an edge between Beef and Meat (an implicit but direct subsumption). Without reasoning, Beef will be misplaced under `owl:Thing`, if no

[13]Refer to the definition of DirectSubClassOf at `https://owlcs.github.io/owlapi/apidocs_4/org/semanticweb/owlapi/reasoner/OWLReasoner.html`.

other subsumptions about Beef are asserted in the ontology. Tools like Protégé[14] (Musen, 2015) also demonstrate similar considerations in presenting ontology taxonomies.

[14] https://protege.stanford.edu/

PART III

Language Models for Ontology Engineering

Implementations

CHAPTER 7

Ontology Engineering with DeepOnto

Contents

This chapter introduces `DeepOnto` (He et al., 2023b), a Python package we developed to support ontology engineering with deep learning techniques, particularly language models (LMs). This tool represents a practical contribution to this thesis, complementing the research advancements discussed in the preceding chapters. The homepage of `DeepOnto` is `https://krr-oxford.github.io/DeepOnto/`, where detailed tutorials and code references are provided.

7.1 Overview

Integrating deep learning techniques, particularly language models (LMs), with knowledge representation techniques like ontologies has raised widespread attention, urging the need of a platform that supports both paradigms.

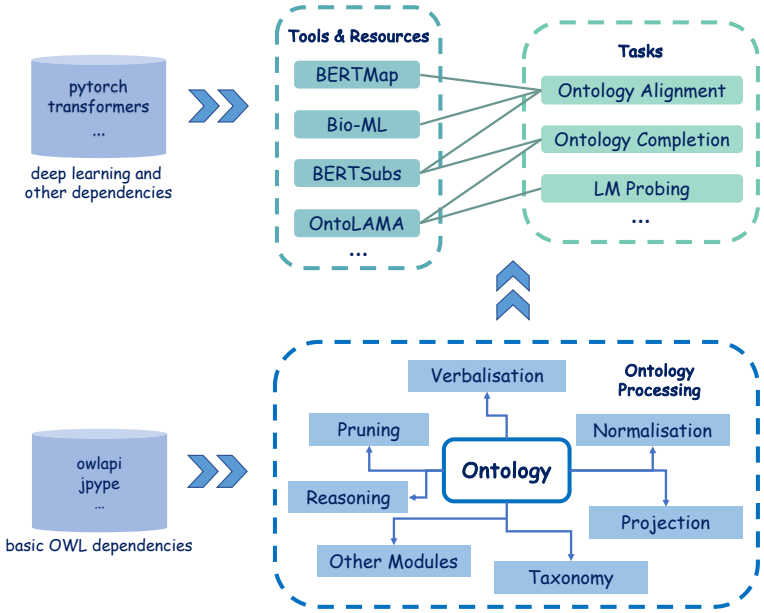


Figure 7.1: Illustration of `DeepOnto`'s architecture, with the lower half depicting the core ontology processing module, and the upper half presenting various tools and resources for diverse ontology engineering tasks. The thick arrow signs indicate dependency support and the thin arrow signs in the core ontology processing module point to sub-modules related to different functionalities.

Despite the availability of tools like OWL API (Horridge and Bechhofer, 2011) and Jena (Carroll et al., 2004) that offer robust support for basic ontology processing, particularly with OWL ontologies, there is a gap in solutions that convert various types of ontology information into formats to facilitate deep learning applications. This gap is further widened by the disparity between the programming languages of popular deep learning frameworks, which are predominantly Python-based, whereas the majority of ontology APIs are Java-based. While mOWL toolkit (Zhapa-Camacho et al., 2023) attempts to bridge this gap by leveraging JPype[1] to integrate Python and Java, its primary focus is on ontology embeddings and their biomedical applications, rather than essential ontology processing capabilities and several other critical tasks in ontology engineering. To address this, we introduce `DeepOnto`, a Python package designed to support deep learning-driven ontology engineering.

As illustrated in Figure 7.1, `DeepOnto`'s core is its ontology processing module,

[1] `https://jpype.readthedocs.io/en/latest/install.html`

enabling fundamental operations like loading, saving, retrieval, ancestor/descendant querying, and entity/axiom modification. It also supports more advanced functions such as reasoning, verbalisation, normalisation, and projection, detailed in Section 7.3. Built upon this foundation, `DeepOnto` offers an array of tools and resources for tasks like ontology alignment and ontology completion, including BERTMap, BERTSubs, Bio-ML, and OntoLAMA that we have mentioned in the previous chapters. `DeepOnto` provides a fairly flexible and extensible interface for further implementations. The incorporation of new tools and resources may lead to the integration of additional foundational features into the core module. This demonstrates the progressive development cycle of `DeepOnto`.

7.2 Design Principle

7.2.1 Dependencies

For our backend dependency, we select the OWL API, renowned for its stability, reliability, and broad adoption in notable projects and tools like Protégé (Musen, 2015), ROBOT (Jackson et al., 2019) and HermiT (Glimm et al., 2014). `DeepOnto` was initially built on Owlready2 (Lamy, 2017), a Python-based ontology API, but we found limitations in its functionality and error handling, particularly when editing or managing multiple ontologies. Although RDFLib[2], on which Owlready2 was built, allows for direct RDF triple manipulation, extending it to fully support OWL ontologies would demand substantial development.

Thus, we opt to integrate the OWL API via JPype, a bridge between Python and the Java Virtual Machine (JVM), following the approach used in the mOWL library[3]. Specifically, the Java dependency files are compiled into the Java Archive (JAR) format and will be shipped upon package installation. JVM can then be launched within a Python program, establishing a connection to the JAR files. We constrain the direct import of the Java dependencies within the central ontology processing module, maintaining them in a relatively static version. This strategy ensures long-term stability and streamlines the updating and management processes of the codebase.

[2] `https://rdflib.dev/`

[3] mOWL is not used as a direct dependency but parts of its functionalities (e.g., normalisation and projection) encapsulated in Java were migrated to `DeepOnto` to avoid duplicated implementations.

DeepOnto adopts PyTorch (Paszke et al., 2019) as the backbone for deep learning dependencies. PyTorch is characterised by its dynamic computation graph, which enables runtime modification of the model's architecture, providing flexibility and ease of use for users. Also, significant efforts are invested to ensure backward compatibility. Currently, ontology engineering modules in DeepOnto primarily focus on applications of LMs, for which the `transformers` library (Wolf et al., 2019) is indispensable. This library, supporting both Tensorflow and PyTorch, provides an extensive collection of pre-trained LMs and relevant utilities. DeepOnto leverages the PyTorch variant of `transformers` as a dependency.

7.2.2 Architecture

The architecture of DeepOnto is designed to be clear and efficient. As shown in Figure 7.1, the basis of DeepOnto a core ontology processing module that integrates and simplifies[4] the capabilities of the OWL API, and extends beyond. This module is anchored by the `Ontology` class[5], a comprehensive interface that offers streamlined access to ontology operations such as entity manipulation, hierarchy exploration, axiom modification, and annotation retrieval. In addition to these basic processing features, DeepOnto incorporates a suite of sub-modules to enhance the core's functionality. These sub-modules, including reasoning, pruning, verbalisation, normalisation, and projection, are tailored to convert ontological information into various formats conducive to deep learning applications. For example, the verbalisation module translates ontology entities into human-readable text, while the projection module transforms ontologies into RDF graphs.

The core ontology processing module of DeepOnto paves the way for the development and integration of individual tools and resources. Currently, DeepOnto is primarily focused on leveraging transformer-based LMs to facilitate a variety of ontology engineering tasks, including ontology alignment, subsumption-based ontology

[4]For instance, when using the OWL API, retrieving subsumption axioms for different entity types requires distinct codes. We have consolidated these similar functions into a single Python method for easier use.

[5]The term *"class"* refers to a blueprint for creating objects – encapsulating attributes and methods in Python programming.

completion, and ontology-informed LM probing. Looking ahead, `DeepOnto` is planned to expand its collection with new tools for ontology embedding and concept insertion, among others. `DeepOnto` can also be extended to a broader context of knowledge engineering, facilitating tasks like entity linking, entity alignment, and link prediction for knowledge graphs.

7.3 Ontology Processing

In this section, we delve into the components of the core ontology processing module, which is central to the functionality of `DeepOnto`.

Ontology At the heart of `DeepOnto` is the `Ontology` class, which serves as the gateway to fundamental ontology manipulation and access operations. Instantiating an `Ontology` object is straightforward: it requires just the path to the ontology file. This simplicity is demonstrated in the Python snippet below, which shows how to load an ontology using `DeepOnto`:

```
from deeponto.onto import Ontology
onto = Ontology("path_to_ontology.owl")
```

Figure 7.2: Code snippet for loading an ontology using `DeepOnto`.

Through the `Ontology` class, users gain the ability to perform a variety of operations. These include accessing named entities (like concepts and properties) via their IRIs, fetching asserted parent and child entities, and manipulating ontology axioms. When implemented in Java using the OWL API, even simple tasks like these can be cumbersome and verbose. `DeepOnto`'s encapsulation in Python significantly enhances code cleanliness and readability.

Ontology Reasoning Every instance of `Ontology` is accompanied by an instance of `OntologyReasoner` as its attribute. It is used for conducting reasoning activities, including obtaining inferred subsumers and subsumees, as well as checking entailment and consistency. By encapsulating these basic reasoning functions, we can implement more complex or specific reasoning algorithms. For example, we have implemented the

assumed disjointness proposed in Section 5.3.2 of Chapter 5, which can be particularly useful in the negative sampling process frequently employed in various machine learning-driven ontology curation tasks, including but not limited to, subsumption prediction. Currently, `DeepOnto` supports three types of reasoners:

- **HermiT** (Glimm et al., 2014), a sound and complete OWL reasoner based on hypertableau calculus.
- **ELK** (Kazakov et al., 2012), a highly optimised reasoner tailored to the OWL 2 EL profile.
- **Structural**[6], a straightforward structural reasoner offering basic reasoning capabilities over entity hierarchies.

Notice that `DeepOnto` can be easily extended to incorporate other reasoners[7] that inherit the reasoner interface of OWL API.

Ontology Pruning Real-world ontologies frequently exhibit a large-scale nature, leading to diminished efficiency during system evaluation. To extract a scalable subset from an ontology, we often prune the ontology by removing its concepts based on certain criteria, e.g., semantic types. In order to maintain the hierarchical structure during the pruning process, we implement the pruning algorithm proposed in Section 4.3.2 of Chapter 4 in the `OntologyPruner` class, which introduces subsumption axioms between the asserted (atomic or complex) parents and children of the concept targeted for removal (see Figure 4.2 for visual illustration). For future development, we aim to incorporate more pruning approaches. A key addition will be ontology modularisation, a technique that seeks to extract a (small) sub-ontology that entails a given axiom or is sufficient to answer a specific concept of queries (d'Aquin et al., 2007). We also aim to explore its various variants that involve approximation, catering to different scenarios such as the construction of personalised data graphs.

[6] `https://owlcs.github.io/owlapi/apidocs_4/org/semanticweb/owlapi/reasoner/structural/StructuralReasoner.html`

[7] E.g., `https://owlapi.sourceforge.net/reasoners.html`

Ontology Verbalisation Verbalising an entity into natural language text that follows the semantics of its original OWL expression can improve an ontology's accessibility and support many ontology engineering tasks. For example, ontology alignment systems, as we have mentioned in Chapter 4, often rely on lexical matching to achieve promising results. While a named entity can be easily verbalised using its labels, complex expressions that involve logical operators need a more sophisticated algorithm. In `DeepOnto`, we implement the recursive concept verbaliser discussed in Section 5.3.4 of Chapter 5 in the `OntologyVerbaliser` class, and we separate the syntax parsing utility to the `OntologySyntaxParser` class.

Ontology Normalisation Normalisation in OWL 2 EL refers to the transformation of axioms into one of the following *normal forms*: $C \sqsubseteq D$, $C \sqcap C' \sqsubseteq D$, $C \sqsubseteq \exists r.D$, $\exists r.C \sqsubseteq D$, $r \sqsubseteq s$, and $r \circ r' \sqsubseteq s$, with no loss of semantics (Baader et al., 2005). Here, C and C' can be named concepts or $\top$, D can be a named concept or $\bot$; and r, r', and s represent roles. The normalisation algorithm has been implemented in several $\mathcal{EL}$ embedding models (Kulmanov et al., 2019; Heindorf et al., 2022) and this feature is also implemented in `DeepOnto` (`OntologyNormaliser`) for easy access. Normalisation can facilitate the training deep learning-based ontology engineering models because it simplifies the set of axiom patterns. For example, a complex subsumption $C \sqsubseteq D \sqcap \exists r.E$ can be normalised into $C \sqsubseteq D$ and $C \sqsubseteq \exists r.E$ such that a model trained on normalised axioms can make independent predictions for each axiom and then combine the results for a joint prediction.

Ontology Taxonomy An ontology defines a concept hierarchy through asserted subsumption axioms, resulting in a taxonomy. This taxonomy simplifies the ontology's structure, allowing easier navigation and retrieval by focusing exclusively on hierarchical relationships. This, in turn, facilitates graph-based deep learning models. In `DeepOnto`, the ontology taxonomy (`OntologyTaxonomy`) is implemented as a directed acyclic graph with named concepts as nodes and subsumption relations as directed edges (following Definition 2.4.7 in Chapter 2). The top concept `owl:Thing` is used as the root node. To establish the subsumption edges, we employ an ontology reasoner to infer *direct*

subsumers for each named concept to prevent misplacement under the root node. The rationale behind this has been detailed in Appendix 6.A of Chapter 6.

Ontology Projection `DeepOnto` offers the capability to transform an OWL ontology into a set of RDF triples. A default method for this transformation, which adheres to the W3C standard[8], is provided by OWL API. This method preserves the ontology's semantics and is effective for storing or exchanging ontologies. However, as it may introduce many blank nodes for representing complex logical expressions, its utility for ontology visualisation or applying graph-based algorithms, such as Random Walk and Graph Neural Networks, is limited. In such situations, a simplified graph representation is often needed, an approach sometimes termed as *ontology projection*[9]. `DeepOnto` has implemented the algorithm (`OntologyProjector`) originally used in the ontology visualisation system OptiqueVQS (Soylu et al., 2018), to transform axioms into a set of simplified RDF triples. Briefly, a concept subsumption axiom $C \sqsubseteq D$ is transformed into $(C, \texttt{rdfs:subClassOf}, D)$, an individual membership axiom $D(d)$ is transformed into $(C, \texttt{rdf:type}, D)$, a role assertion axiom $r(a, b)$ is transformed into (a, r, b), a restriction axiom in the form of $C \sqsubseteq \exists r.D$ or $C \sqsubseteq \forall r.D$ is transformed into (C, r, D). Here, C and D are concepts; a, b, and d denote individuals.

7.4 Tools and Resources

The current version of `DeepOnto` has integrated several tools and resources for ontology alignment and ontology completion, including those mentioned in the previous chapters. For ontology alignment, `DeepOnto` has BERTMap (He et al., 2022a) for concept equivalence matching, BERTSubs (Inter) (Chen et al., 2023) for concept subsumption matching, Bio-ML (He et al., 2022b) and its sub-track Bio-LLM (He et al., 2023a) for benchmarking (all in Chapter 4). For ontology completion, `DeepOnto` has BERTSubs (Intra), and the prompt-based inference approach proposed in OntoLAMA (He et al., 2023c). As

[8] `https://www.w3.org/TR/owl2-mapping-to-rdf/`

[9] Note that ontology taxonomy can be seen as a special case of projection, but we separate the functionalities into different modules because the former is purely for concept hierarchy and the latter is for triples.

introduced in Chapter 5, OntoLAMA also involves a collection of subsumption inference datasets for language model probing.

We also developed the `HierarchyTransformers` library that follows the layout of `SentenceTransformers` (Reimers and Gurevych, 2019), providing an easy-to-use interface for loading, training, and evaluating the HiT models along with other baselines discussed in Chapter 6. This library depends on `DeepOnto` for data processing, especially the ontology taxonomy module mentioned in Section 7.3.

7.5 Impact and Use

`DeepOnto` has been gaining attention from both the industry and academia. On the industrial front, notable adoptions of `DeepOnto` include its utilisation by Samsung Research UK (SRUK) for Digital Health Coaching[10] and Madrid Digital[11], where it played a crucial role in their Proof of Concept process. Feeback from our users indicates `DeepOnto`'s primary applications within the health and biomedical sectors. From the academia side, `DeepOnto` contributes to the OAEI Bio-ML track for extensive OM benchmarking, and our tools and resources have facilitated numerous research works, evident in existing scholarly citations. For example, BERTMap serves as a typical baseline in subsequent BERT-based ontology alignment studies, including the studies by Amir et al. (2023), Wang (2023), and Wang et al. (2023), to name a few.

7.6 Future Work

We envision `DeepOnto` evolving into a valuable toolkit for the ontology engineering community. As we move forward, our plan is to continuously enhance the toolkit by expanding its features (e.g., ontology modularisation), and by integrating new tools and resources to facilitate more automated tasks in ontology engineering. These include, but are not limited to, embedding concepts and properties with both formal semantics and literals considered, placing new concepts derived from text mentions into an ontology, generating descriptions for concepts based on their contexts in an ontology,

[10] `https://research.samsung.com/sruk`
[11] `https://www.comunidad.madrid/en/servicios/sede-electronica/madrid-digital`

and identifying, generating, and placing appropriate common parents for concepts clustered under certain criteria. As illustrated in the progressive development circle of `DeepOnto`, these enhancements will elicit requirements of more ontology processing features, all of which will be seamlessly integrated into our core API.

CHAPTER 8

Conclusion

This thesis explores automating ontology engineering with language models (LMs). Comprehensive background information on ontologies and language models is provided in Part I, encompassing Chapters 2 and 3. Our methodologies are detailed in Part II, which includes Chapters 4, 5, and 6. Lastly, the practical contributions of this research are discussed in Part III, specifically in Chapter 7.

In Chapter 2, we introduce an important standard for computational ontologies, namely the Web Ontology Language (OWL). OWL supports two different semantic frameworks: Direct Semantics, which aligns with the $\mathcal{SROIQ}$ Description Logic (DL) formalism, and RDF-based Semantics, which is associated with RDF graphs. Our discussion mainly centers on Direct Semantics, through which we introduce the essential basics of DL to aid in explaining the $\mathcal{SROIQ}$ DL that underpins OWL, providing a clearer understanding of its structure and capabilities. Subsequent to exploring OWL ontologies, we discuss the specific ontology engineering tasks that this thesis addresses: ontology alignment, ontology completion, and hierarchy embedding. For each of these tasks, we provide detailed definitions and discussions concerning both the nature of the tasks and their respective evaluation methods.

In Chapter 3, we go through the evolution of language models (LMs) from early statistical N-gram models to contemporary transformer-based LMs. We introduce essential preliminaries of typical LMs along the path, presenting detailed definitions

through the lens of probability theory. The discussion then progresses to the architecture of transformers and transformer-based LMs. Various learning paradigms of transformer-based LMs are introduced, including pre-training, fine-tuning, contrastive learning, and prompt learning. We also discuss recent large language models (LLMs) and highlight the Retrieval-Augmented Generation (RAG) as a promising approach for future applications in knowledge engineering. With this background, we proceed to review specific literature on the utilisation of LMs in knowledge engineering tasks. A summary of these works is systematically presented in Table 3.2.

In Chapter 4, we discuss our contributions to ontology alignment, or ontology matching (OM), including the BERTMap system and the Bio-ML resource. BERTMap represents one of the pioneering efforts to effectively integrate transformer-based LMs for OM tasks. Given the critical role of lexical matching in OM, leveraging the advanced language processing capabilities of LMs offers a promising and viable approach. Our review of related work reveals that while symbolic (or rule-based) systems still shine in OM, particularly in equivalence matching, their predictive limitations restrict their application in more complex settings such as subsumption matching. Prior to BERTMap, machine learning-based solutions in OM mainly focused on supervised learning approaches, which are often impractical for OM due to the lack of labelled data and extreme class imbalance. Unsupervised methods have been explored less and typically do not incorporate contextualised language understanding. BERTMap addresses many of the mentioned challenges. It primarily operates in an unsupervised mode, although it also supports semi-supervised and data-augmented modes to accommodate various real-world scenarios. The system begins by constructing text semantics corpora derived from input ontologies, followed by fine-tuning a BERT model on these corpora. This fine-tuned BERT model serves as a synonym classifier, which, along with a string-matching module, assists in predicting ontology mappings. The initial mappings generated by BERTMap are then subjected to a series of refinements, including threshold filtering, mapping extension, and mapping repair, to enhance the precision and recall of the alignment results.

Regarding Bio-ML, it is a biomedical OM resource consisting of five distinct OM tasks, each offering settings for both equivalence and subsumption matching. Bio-ML stands out from previous OM resources by providing a comprehensive and adaptable evaluation framework that accommodates both rule-based and machine learning-based OM systems. Over the past two years, Bio-ML has been incorporated into one of the tracks of the Ontology Alignment Evaluation Initiative (OAEI), serving as a benchmark for evaluating the performance of various OM systems. The 2023 update to Bio-ML introduced a special sub-track for LLM-based OM solutions, named Bio-LLM. In our evaluation, we conduct experiments of BERTMap on both the OAEI LargeBio and Bio-ML benchmarks. Additionally, we discuss preliminary findings on the application of LLMs to OM.

In Chapter 5, we explore how the task of ontology completion, specifically subsumption inference (SI), can be adapted to the capabilities of LMs. There is growing interest in assessing the extent to which LMs have encoded knowledge and their potential to function as knowledge bases (KBs). However, most of the studies in this literature focus on simple, triple-based, relational KBs, while omitting more structured, logic-based, conceptualised KBs. To compensate this, we introduce OntoLAMA, a collection of LM probing datasets for the task of SI. We reformulate SI within the framework of natural language inference (NLI), adapting it to better align with the operational paradigms of LMs. To facilitate this, we develop heuristics to extract subsumption samples from ontologies, and a recursive concept verbaliser to transform concept expressions in these samples into natural language texts, which are then used to probe the LMs. We also discuss a prompt-based probing method as an effective technique for eliciting stored knowledge from LMs. Our findings demonstrate that LMs, with the aid of a limited number of training samples, can effectively perform subsumption inference. Although our primary focus in this chapter is on LM probing, we categorise this work within the broader context of LMs for ontology completion to streamline the reading of the thesis.

In Chapter 6, we investigate how to explicitly encode hierarchical structures with LMs. We introduce a novel method to re-train transformer encoder-based LMs as Hierarchy Transformer Encoders (HITs), leveraging the expansive nature of hyperbolic

geometry. As the output embedding space of LMs is typically constrained to a d-dimensional hyper-cube because of the $\tanh$ activation function in the output layer, we can construct a Poincaré ball of radius $\sqrt{d}$ such that its boundary circumscribes the LM's output space. Based on this manifold, we propose hyperbolic clustering and hyperbolic centripetal losses for effectively clustering related entities while organising them hierarchically. We demonstrate the effectiveness of our HiT models through tasks that assess their ability to generalise from asserted entity subsumptions to inferred and unseen ones. While there have been prior efforts to incorporate hyperbolic losses into word embeddings, HiT represents the first initiative to apply such principles to transformer-based LMs for explicit hierarchy encoding. Moreover, previous hyperbolic embedding approaches are typically static and limited by a fixed vocabulary – this means they are not capable of predicting inductively. In contrast, the inherent sub-word tokenisation of transformer-based LMs allows HiT adapt to new and evolving vocabularies, thereby enhancing its applicability in real-world scenarios.

In Chapter 7, we discuss the `DeepOnto` package, which stands as the practical contribution of this thesis. The core module of `DeepOnto` simplifies and enhances the functionality of the OWL API, making basic ontology processing more accessible while extending its capabilities to include advanced features such as reasoning, pruning, verbalisation, normalisation, and projection. These enhancements are designed to support the data transformation requirements in deep learning-based ontology engineering applications. Building on this core module, `DeepOnto` provides polished implementations of BERTMap, BERTSubs, OntoLAMA, and Bio-ML. Additionally, it serves as a data processing dependency for `HierarchyTransformer`, the library developed for our HiT models. We can deem `DeepOnto` as the code repository for this thesis.

Overall, this thesis serves as a compelling example of Neural-Symbolic integration, showcasing the diverse applications of LMs across various aspects of ontology engineering. Looking ahead, our focus will be on maximising the utilisation of information and preserving more semantics. This is crucial for advancing towards our ultimate goal of automating the construction of knowledge bases from a variety of information sources and paving the way for more sophisticated systems in knowledge engineering.

References

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

R. Abboud, I. Ceylan, T. Lukasiewicz, and T. Salvatori. Boxe: A box embedding model for knowledge base completion. *Advances in Neural Information Processing Systems*, 33:9649–9661, 2020.

B. AlKhamissi, M. Li, A. Celikyilmaz, M. Diab, and M. Ghazvininejad. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*, 2022.

E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, W. Redmond, and M. B. McDermott. Publicly Available Clinical BERT Embeddings. *NAACL HLT 2019*, page 72, 2019.

M. Amir, M. Baruah, M. Eslamialishah, S. Ehsani, A. Bahramali, S. Naddaf-Sh, and S. Zarandioon. Truveta Mapper: A Zero-shot Ontology Alignment Framework. *arXiv preprint arXiv:2301.09767*, 2023.

D. Araci. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*, 2019.

S. Arora, Y. Liang, and T. Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*, 2017.

M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the 12th international joint conference on Artificial intelligence-Volume 1*, pages 452–457, 1991.

F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 364–369, 2005.

F. Baader, I. Horrocks, and U. Sattler. Description logics. *Foundations of Artificial Intelligence*, 3:135–179, 2008.

S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein, et al. OWL web ontology language reference. *W3C recommendation*, 2004.

G. Becigneul and O.-E. Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2018.

O. Bodenreider. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi. COMET: Commonsense Transformers for Automatic Knowledge Graph Construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, 2019.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83, 2004.

J. Caufield, H. Hegde, V. Emonet, N. Harris, M. Joachimiak, N. Matentzoglu, H. Kim, S. Moxon, J. Reese, M. Haendel, et al. Structured Prompt Interrogation and Recursive Extraction of Semantics (SPIRES): a method for populating knowledge bases using zero-shot learning. *Bioinformatics (Oxford, England)*, pages btae104–btae104, 2024.

B. Chen, Y. Fu, G. Xu, P. Xie, C. Tan, M. Chen, and L. Jing. Probing BERT in Hyperbolic Spaces. In *International Conference on Learning Representations*, 2020.

J. Chen, P. Hu, E. Jimenez-Ruiz, O. M. Holter, D. Antonyrajah, and I. Horrocks. OWL2Vec*: Embedding of OWL ontologies. *Machine Learning*, 110(7):1813–1845, 2021a.

J. Chen, E. Jiménez-Ruiz, I. Horrocks, D. Antonyrajah, A. Hadian, and J. Lee. Augmenting ontology alignment by semantic embedding and distant supervision. In *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18*, pages 392–408. Springer, 2021b.

J. Chen, Y. He, Y. Geng, E. Jiménez-Ruiz, H. Dong, and I. Horrocks. Contextual semantic embeddings for ontology subsumption prediction. *World Wide Web*, 26(5):2569–2591, 2023.

K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, 2014.

H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

P. G. Cotovio, L. Ferraz, D. Faria, L. Balbi, M. C. Silva, and C. Pesquita. Matcha-DL a tool for supervised ontology alignment. *preprint*, 2024.

J. Da, R. Le Bras, X. Lu, Y. Choi, and A. Bosselut. Analyzing Commonsense Emergence in Few-shot Knowledge Models. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

B. Dan and R. Guha. RDF vocabulary description language 1.0: RDF schema. *W3C recommendation*, 2004.

G. K. D. De Vries, S. De Rooij, et al. A Fast and Simple Graph Kernel for RDF. *DMoLD*, 1082, 2013.

G. Delétang, A. Ruoss, P.-A. Duquenne, E. Catt, T. Genewein, C. Mattern, J. Grau-Moya, L. K. Wenliang, M. Aitchison, L. Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

B. Dhingra, C. Shallue, M. Norouzi, A. Dai, and G. Dahl. Embedding Text in Hyperbolic Spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 59–69, 2018.

B. Dhingra, J. R. Cole, J. M. Eisenschlos, D. Gillick, J. Eisenstein, and W. W. Cohen. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273, 2022.

H. Dong, J. Chen, Y. He, and I. Horrocks. Ontology enrichment from texts: A biomedical dataset for concept discovery and placement. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 5316–5320, 2023a.

H. Dong, J. Chen, Y. He, Y. Liu, and I. Horrocks. Reveal the unknown: Out-of-knowledge-base mention discovery with entity linking. In *Proceedings of the 32nd*

ACM International Conference on Information and Knowledge Management, pages 452–462, 2023b.

D. M. Dooley, E. J. Griffiths, G. S. Gosal, P. L. Buttigieg, R. Hoehndorf, M. C. Lange, L. M. Schriml, F. S. Brinkman, and W. W. Hsiao. FoodOn: a harmonized food ontology to increase global food traceability, quality control and data integration. *npj Science of Food*, 2(1):23, 2018.

M. Dragoni, T. Bailoni, R. Maimone, and C. Eccher. Helis: An ontology for supporting healthy lifestyles. In *The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part II 17*, pages 53–69. Springer, 2018.

M. d'Aquin, A. Schlicht, H. Stuckenschmidt, and M. Sabou. Ontology modularization for knowledge selection: Experiments and evaluations. In *Database and Expert Systems Applications: 18th International Conference, DEXA 2007, Regensburg, Germany, September 3-7, 2007. Proceedings 18*, pages 874–883. Springer, 2007.

D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, and F. M. Couto. The agreementmakerlight ontology matching system. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences: Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings*, pages 527–541. Springer, 2013.

M. Freitag and Y. Al-Onaizan. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, 2017.

O. Ganea, G. Bécigneul, and T. Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*, pages 1646–1655. PMLR, 2018a.

O. Ganea, G. Bécigneul, and T. Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*, pages 1646–1655. PMLR, 2018b.

T. Gao, A. Fisch, and D. Chen. Making Pre-trained Language Models Better Few-shot Learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, 2021a.

T. Gao, X. Yao, and D. Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, 2021b.

T. Gao, H. Yen, J. Yu, and D. Chen. Enabling Large Language Models to Generate Text with Citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488, 2023.

A. d. Garcez, S. Bader, H. Bowman, L. C. Lamb, L. de Penning, B. Illuminoo, H. Poon, and C. G. Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *Neuro-Symbolic Artificial Intelligence: The State of the Art*, 342(1): 327, 2022.

B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. Hermit: an OWL 2 reasoner. *Journal of Automated Reasoning*, 53:245–269, 2014.

A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media, 2006.

F. Gosselin and A. Zouaq. SORBET: A Siamese Network for Ontology Embeddings Using a Distance-Based Regression Loss and BERT. In *International Semantic Web Conference*, pages 561–578. Springer, 2023.

B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler. A logical framework for modularity of ontologies. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 298–303, 2007.

M. Gromov. Hyperbolic groups. In *Essays in group theory*, pages 75–263. Springer, 1987.

T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928, 1995.

N. Guarino, D. Oberle, and S. Staab. What is an ontology? *Handbook on ontologies*, pages 1–17, 2009.

R. V. Guha, D. Brickley, and S. Macbeth. Schema. org: evolution of structured data on the web. *Communications of the ACM*, 59(2):44–51, 2016.

B. Gunel, J. Du, A. Conneau, and V. Stoyanov. Supervised Contrastive Learning for Pre-trained Language Model Fine-tuning. In *International Conference on Learning Representations*, 2020.

M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

M. Haendel, N. Vasilevsky, D. Unni, C. Bologa, N. Harris, H. Rehm, A. Hamosh, G. Baynam, T. Groza, J. McMurry, et al. How many rare diseases are there? *Nature reviews drug discovery*, 19(2):77–78, 2020.

A. Hamosh, A. F. Scott, J. S. Amberger, C. A. Bocchini, and V. A. McKusick. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic acids research*, 33(suppl_1):D514–D517, 2005.

K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022.

M. Hanna and D. Mareček. Analyzing BERT's knowledge of hypernymy via prompting. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 275–282, 2021.

Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

I. Harrow, E. Jiménez-Ruiz, A. Splendiani, M. Romacker, P. Woollard, S. Markel, Y. Alam-Faruque, M. Koch, J. Malone, and A. Waaler. Matching disease and phenotype ontologies in the ontology alignment evaluation initiative. *Journal of biomedical semantics*, 8:1–13, 2017.

Y. He, J. Chen, D. Antonyrajah, and I. Horrocks. BERTMap: A BERT-Based Ontology Alignment System. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5): 5684–5691, Jun. 2022a. doi: 10.1609/aaai.v36i5.20510. URL `https://ojs.aaai.org/index.php/AAAI/article/view/20510`.

Y. He, J. Chen, H. Dong, E. Jiménez-Ruiz, A. Hadian, and I. Horrocks. Machine learning-friendly biomedical datasets for equivalence and subsumption ontology matching. In *International Semantic Web Conference*, pages 575–591. Springer, 2022b.

Y. He, J. Chen, H. Dong, and I. Horrocks. Exploring large language models for ontology alignment. *arXiv preprint arXiv:2309.07172*, 2023a.

Y. He, J. Chen, H. Dong, I. Horrocks, C. Allocca, T. Kim, and B. Sapkota. DeepOnto: A Python package for ontology engineering with deep learning. *arXiv preprint arXiv:2307.03067*, 2023b.

Y. He, J. Chen, E. Jimenez-Ruiz, H. Dong, and I. Horrocks. Language Model Analysis for Ontology Subsumption Inference. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3439–3453, 2023c.

Y. He, Z. Yuan, J. Chen, and I. Horrocks. Language Models as Hierarchy Encoders. *arXiv preprint arXiv:2401.11374*, 2024.

S. Heindorf, L. Blübaum, N. Düsterhus, T. Werner, V. N. Golani, C. Demir, and A.-C. Ngonga Ngomo. Evolearner: Learning description logics with evolutionary algorithms. In *Proceedings of the ACM Web Conference 2022*, pages 818–828, 2022.

S. Hertling and H. Paulheim. Atbox results for OAEI 2022. In *OM@ ISWC*, 2022.

S. Hertling and H. Paulheim. OLaLa: Ontology matching with large language models. In *Proceedings of the 12th Knowledge Capture Conference 2023*, pages 131–139, 2023.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic web*, 2(1):11–21, 2011.

I. Horrocks, B. Parsia, and U. Sattler. OWL 2 web ontology language direct semantics. *World Wide Web Consortium*, pages 42–65, 2012.

N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

S. Hu, N. Ding, H. Wang, Z. Liu, J. Wang, J. Li, W. Wu, and M. Sun. Knowledgeable Prompt-tuning: Incorporating Knowledge into Prompt Verbalizer for Text Classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2225–2240, 2022.

V. Iyer, A. Agarwal, and H. Kumar. VeeAlign: multifaceted context representation using dual attention for ontology alignment. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10780–10792, 2021.

R. C. Jackson, J. P. Balhoff, E. Douglass, N. L. Harris, C. J. Mungall, and J. A. Overton. ROBOT: a tool for automating ontology workflows. *BMC bioinformatics*, 20:1–10, 2019.

E. Jiménez-Ruiz and B. Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *The Semantic Web–ISWC 2011: 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I 10*, pages 273–288. Springer, 2011.

E. Jiménez-Ruiz, B. C. Grau, I. Horrocks, and R. Berlanga. Logic-based assessment of the compatibility of UMLS ontology sources. *Journal of biomedical semantics*, 2:1–16, 2011.

E. Jiménez-Ruiz, C. Meilicke, B. C. Grau, and I. Horrocks. Evaluating Mapping Repair Systems with Large Biomedical Ontologies. *Description Logics*, 13:246–257, 2013.

E. Jiménez-Ruiz, A. Agibetov, J. Chen, M. Samwald, and V. Cross. Dividing the Ontology Alignment Task with Semantic Embeddings and Logic-Based Modules. In *ECAI 2020*, pages 784–791. IOS Press, 2020.

P. Kavumba, R. Takahashi, and Y. Oda. Are Prompt-based Models Clueless? In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2333–2352, 2022.

Y. Kazakov, M. Krötzsch, and F. Simančík. ELK: a reasoner for OWL EL ontologies. System description, University of Oxford, 2012. available from `http://code.google.com/p/elk-reasoner/wiki/Publications`.

Z. Ke, Y. Shao, H. Lin, T. Konishi, G. Kim, and B. Liu. Continual Pre-training of Language Models. In *The Eleventh International Conference on Learning Representations*, 2022.

J. D. M.-W. C. Kenton and L. K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2, 2019.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

M. Kochurov, R. Karimov, and S. Kozlukov. Geoopt: Riemannian optimization in pytorch. *arXiv preprint arXiv:2005.02819*, 2020.

P. Kolyvakis, A. Kalousis, and D. Kiritsis. DeepAlignment: Unsupervised ontology matching with refined word vectors. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 787–798, 2018.

M. Kulmanov, W. Liu-Wei, Y. Yan, and R. Hoehndorf. El embeddings: Geometric construction of models for the description logic el++. *arXiv preprint arXiv:1902.10499*, 2019.

J.-B. Lamy. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine*, 80:11–28, 2017.

O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. *W3C Recommendation*, 1999.

Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

P. Lewis, M. Ott, J. Du, and V. Stoyanov. Pretrained language models for biomedical and clinical tasks: understanding and extending the state-of-the-art. In *Proceedings of the 3rd clinical natural language processing workshop*, pages 146–157, 2020a.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020b.

N. Li, T. Bailleux, Z. Bouraoui, and S. Schockaert. Ontology Completion with Natural Language Inference and Concept Embeddings: An Analysis. *arXiv preprint arXiv:2403.17216*, 2024.

X. Li, J. Thickstun, I. Gulrajani, P. S. Liang, and T. B. Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35: 4328–4343, 2022.

R. Lin and H. T. Ng. Does BERT Know that the IS-A Relation Is Transitive? In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 94–99, 2022.

F. Liu, E. Shareghi, Z. Meng, M. Basaldella, and N. Collier. Self-Alignment Pretraining for Biomedical Entity Representations. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4228–4238, 2021a.

H. Liu, Y. Perl, and J. Geller. Concept placement using BERT trained by transforming and summarizing biomedical ontology structure. *Journal of Biomedical Informatics*, 112:103607, 2020a.

L. Liu and M. T. Özsu. *Encyclopedia of database systems*, volume 6. Springer New York, 2009.

L. Liu, X. Li, R. He, L. Bing, S. Joty, and L. Si. Enhancing Multilingual Language Model with Massive Multilingual Knowledge Triples. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6878–6890, 2022.

P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang. K-bert: Enabling language representation with knowledge graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2901–2908, 2020b.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Z. Liu, Y. Wang, J. Kasai, H. Hajishirzi, and N. A. Smith. Probing Across Time: What Does RoBERTa Know and When? In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 820–842, 2021b.

I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2018.

L. LUO, Y.-F. Li, R. Haf, and S. Pan. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In *The Twelfth International Conference on Learning Representations*, 2023.

M.-T. Luong, H. Pham, and C. D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. ISBN 9781139472104. URL `https://books.google.co.uk/books?id=t1PoSh4uwVcC`.

I. Melnyk, P. Dognin, and P. Das. Grapher: Multi-stage knowledge graph construction using pretrained language models. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

T. Mikolov et al. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 80(26), 2012.

G. A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38 (11):39–41, 1995.

B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, et al. OWL 2 web ontology language profiles. *W3C recommendation*, 2009.

N. Mrkšić, D. Ó. Séaghdha, B. Thomson, M. Gasic, L. M. R. Barahona, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young. Counter-fitting Word Vectors to Linguistic Constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, 2016.

C. J. Mungall, S. Koehler, P. Robinson, I. Holmes, and M. Haendel. k-BOOM: A Bayesian approach to ontology structure inference, with applications in disease ontology construction. *bioRxiv*, page 048843, 2016.

M. A. Musen. The protégé project: a look back and a look forward. *AI matters*, 1(4):4–12, 2015.

J. Nash. The imbedding problem for riemannian manifolds. *Annals of mathematics*, 63 (1):20–63, 1956.

S. Neutel and M. H. de Boer. Towards Automatic Ontology Alignment using BERT. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 2021.

V. Nguyen, H. Y. Yip, and O. Bodenreider. Biomedical vocabulary alignment at scale in the UMLS metathesaurus. In *Proceedings of the Web Conference 2021*, pages 2672–2683, 2021.

M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

I. Nkisi-Orji, N. Wiratunga, S. Massie, K.-Y. Hui, and R. Heaven. Ontology alignment based on word embedding and random forest classification. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*, pages 557–572. Springer, 2019.

OpenAI. GPT-4 Technical Report. *ArXiv*, abs/2303.08774, 2023.

L. Otero-Cerdeira, F. J. Rodríguez-Martínez, and A. Gómez-Rodríguez. Ontology matching: A literature review. *Expert Systems with Applications*, 42(2):949–971, 2015.

S. Padó and I. Dagan. 679Textual Entailment. In *The Oxford Handbook of Computational Linguistics*. Oxford University Press, 06 2022. ISBN 9780199573691.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

C. Pesquita, D. Faria, E. Santos, and F. M. Couto. To repair or not to repair: reconciling correctness and coherence in ontology reference alignments. In *Proc. 8th ISWC ontology matching workshop (OM), Sydney (AU), page this volume*, volume 3, 2013.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In M. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://aclanthology.org/N18-1202`.

F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller. Language Models as Knowledge Bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, 2019a.

F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller. Language Models as Knowledge Bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, 2019b.

A. Poliak. A survey on Recognizing Textual Entailment as an NLP Evaluation. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 92–109, 2020.

J. Portisch and H. Paulheim. ALOD2Vec matcher. *OM@ ISWC*, 2288:132–137, 2018.

J. Portisch, M. Hladik, and H. Paulheim. Wiktionary matcher. In *CEUR Workshop Proceedings*, volume 2536, pages 181–188. RWTH Aachen, 2020.

A. Poulis, E. Tsalapati, and M. Koubarakis. Reasoning over Description Logic-based Contexts with Transformers. *arXiv preprint arXiv:2311.08941*, 2023.

A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI*, 2018.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

P. Ristoski and H. Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In *The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part I 15*, pages 498–514. Springer, 2016.

C. Rosse and J. L. Mejino Jr. The foundational model of anatomy ontology. In *Anatomy ontologies for bioinformatics: principles and practice*, pages 59–117. Springer, 2008.

D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al. Learning internal representations by error propagation, 1985.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

J. Salazar, D. Liang, T. Q. Nguyen, and K. Kirchhoff. Masked Language Model Scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, 2020.

G. Sarti, N. Feldhus, L. Sickert, and O. Van Der Wal. Inseq: An Interpretability Toolkit for Sequence Generation Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 421–435, 2023.

T. Schick and H. Schütze. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, 2021.

S. Schlobach. Debugging and semantic clarification by pinpointing. In *European Semantic Web Conference*, pages 226–240. Springer, 2005.

M. Schneider, J. Carroll, I. Herman, P. F. Patel-Schneider, et al. OWL 2 web ontology language rdf-based semantics. *W3C Recommendation*, 2009.

L. M. Schriml, C. Arze, S. Nadendla, Y.-W. W. Chang, M. Mazaitis, V. Felix, G. Feng, and W. A. Kibbe. Disease Ontology: a backbone for disease semantic integration. *Nucleic acids research*, 40(D1):D940–D946, 2012.

C. E. Shannon. Prediction and entropy of printed English. *Bell system technical journal*, 30(1):50–64, 1951.

A. Sharma, A. Patel, and S. Jain. Lsmatch and lsmatch-multilingual results for OAEI. *OM@ ISWC*, 2022.

K. A. Shefchek, N. L. Harris, M. Gargano, N. Matentzoglu, D. Unni, M. Brush, D. Keith, T. Conlin, N. Vasilevsky, X. A. Zhang, et al. The Monarch Initiative in 2019: an integrative data and analytic platform connecting phenotypes to genotypes across species. *Nucleic acids research*, 48(D1):D704–D715, 2020.

P. Shvaiko and J. Euzenat. Ontology matching: state of the art and future challenges. *IEEE Transactions on knowledge and data engineering*, 25(1):158–176, 2011.

E. Simperl and M. Luczak-Rösch. Collaborative ontology engineering: a survey. *The Knowledge Engineering Review*, 29(1):101–131, 2014.

N. Sioutos, S. de Coronado, M. W. Haber, F. W. Hartel, W.-L. Shaiu, and L. W. Wright. NCI Thesaurus: a semantic model integrating cancer-related clinical and molecular information. *Journal of biomedical informatics*, 40(1):30–43, 2007.

A. Solimando, E. Jimenez-Ruiz, and G. Guerrini. Minimizing conservativity violations in ontology alignments: Algorithms and evaluation. *Knowledge and Information Systems*, 51:775–819, 2017.

K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33: 16857–16867, 2020.

A. Soylu, E. Kharlamov, D. Zheleznyakov, E. Jimenez-Ruiz, M. Giese, M. G. Skjæveland, D. Hovland, R. Schlatte, S. Brandt, H. Lie, et al. OptiqueVQS: A visual query system over ontologies for industry. *Semantic Web*, 9(5):627–660, 2018.

R. Speer, C. Havasi, et al. Representing general relational knowledge in conceptNet 5. In *LREC*, volume 2012, pages 3679–86, 2012.

M. Q. Stearns, C. Price, K. A. Spackman, and A. Y. Wang. SNOMED clinical terms: overview of the development process and project status. In *Proceedings of the AMIA Symposium*, page 662. American Medical Informatics Association, 2001.

R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data & knowledge engineering*, 25(1-2):161–197, 1998.

M. Sung, J. Lee, S. Yi, M. Jeon, S. Kim, and J. Kang. Can Language Models be Biomedical Knowledge Bases? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4723–4734, 2021.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

X. Tang, J. Zhang, B. Chen, Y. Yang, H. Chen, and C. Li. BERT-INT: a BERT-based interaction model for knowledge graph alignment. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3174–3180, 2021.

E. Thiéblin, O. Haemmerlé, N. Hernandez, and C. Trojahn. Survey on complex ontology matching. *Semantic Web*, 11(4):689–727, 2020.

A. Tifrea, G. Becigneul, and O.-E. Ganea. Poincare Glove: Hyperbolic Word Embeddings. In *International Conference on Learning Representations*, 2018a.

A. Tifrea, G. Becigneul, and O.-E. Ganea. Poincare glove: Hyperbolic word embeddings. In *International Conference on Learning Representations*, 2018b.

B. C. Van Fraassen and V. Fraassen. *Formal semantics and logic*, volume 214. Macmillan New York, 1971.

D. Vasant, L. Chanas, J. Malone, M. Hanauer, A. Olry, S. Jupp, P. N. Robinson, H. Parkinson, and A. Rath. Ordo: an ontology connecting rare disease, epidemiology and genetic data. In *Proceedings of ISMB*, volume 30. researchgate. net, 2014.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

O. Vinyals and Q. Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

W3C OWL Working Group. OWL 2 web ontology language document overview. *W3C recommendation*, 2009.

L. L. Wang, C. Bhagavatula, M. Neumann, K. Lo, C. Wilhelm, and W. Ammar. Ontology Alignment in the Biomedical Domain Using Entity Definitions and Context. *BioNLP 2018*, page 47, 2018.

Q. Wang, Z. Gao, and R. Xu. Exploring the In-context Learning Ability of Large Language Model for Biomedical Concept Linking. *arXiv preprint arXiv:2307.01137*, 2023.

W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations*, 2022.

Z. Wang. Amd results for OAEI 2022. In *OM@ ISWC*, pages 145–152, 2022.

Z. Wang. Contextualized Structural Self-supervised Learning for Ontology Matching. *arXiv preprint arXiv:2310.03840*, 2023.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

E. W. D. Whittaker. *Statistical language modelling for automatic speech recognition of Russian and English.* PhD thesis, University of Cambridge, 2000.

A. Williams, N. Nangia, and S. Bowman. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, 2018.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer. Scalable Zero-shot Entity Linking with Dense Entity Retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, 2020.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

C. Xiang, T. Jiang, B. Chang, and Z. Sui. Ersom: A structural ontology matching approach using automatically learned entity representation. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2419–2429, 2015.

B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

H. Ye, N. Zhang, S. Deng, X. Chen, H. Chen, F. Xiong, X. Chen, and H. Chen. Ontology-enhanced Prompt-tuning for Few-shot Learning. In *Proceedings of the ACM Web Conference 2022*, pages 778–787, 2022.

X. Yue, B. Wang, Z. Chen, K. Zhang, Y. Su, and H. Sun. Automatic Evaluation of Attribution by Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4615–4635, 2023.

Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, 2019.

F. Zhapa-Camacho, M. Kulmanov, and R. Hoehndorf. mOWL: Python library for machine learning with biomedical ontologies. *Bioinformatics*, 39(1):btac811, 2023.