Active learning for axiom discovery

Ali Ballout

Apprentissage Actif pour la Découverte d'Axiomes

Ali BALLOUT

I3S, CNRS, Centre Inria d'Université Côte d'Azur, équipe WIMMICS

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur
Dirigée par : Andrea TETTAMANZI, Université Côte d'Azur, CNRS, Inria, I3S
Co-encadrée par : Célia DA COSTA PEREIRA, Université Côte d'Azur, I3S
Soutenue le : 24 juin 2024

Devant le jury, composé de :
Fabien GANDON, Inria
Dario MALCHIODI, Università degli Studi di Milano
Federico ULLIANA, Université de Montpellier, LIRMM
Salem BENFERHAT, Université d'Artois, CRIL

Apprentissage Actif pour la Découverte d'Axiomes

COMPOSITION DU JURY

Président du jury

Fabien GANDON, Directeur de Recherche, Inria

Rapporteurs

Dario Malchiodi, Professeur Associé.,Università degli Studi di Milano

Federico ULLIANA, Professeur Associé, Université de Montpellier, LIRMM

Salem Benferhat, Professeur des Universités, Université d'Artois, CRIL

Examinateurs

Fabien Gandon, Directeur de Recherche, Inria

Directeur de thèse

Andrea Tettamanzi, Professeur des Universités, Université Côte d'Azur, CNRS, Inria, I3S

Célia da Costa Pereira, Professeur Associé, HDR, Université Côte d'Azur, I3S

RÉSUMÉ

Cette thèse aborde le défi de l'évaluation des formules logiques candidates, avec un accent particulier sur les axiomes, en combinant de manière synergique l'apprentissage automatique et le raisonnement symbolique. Cette approche innovante facilite la découverte automatique d'axiomes, principalement dans la phase d'évaluation des axiomes candidats générés. La recherche vise à résoudre le problème de la validation efficace et précise de ces candidats dans le contexte plus large de l'acquisition de connaissances sur le Web sémantique.

Reconnaissant l'importance des heuristiques de génération existantes pour les axiomes candidats, cette recherche se concentre sur l'avancement de la phase d'évaluation de ces candidats. Notre approche consiste à utiliser ces candidats basés sur des heuristiques, puis à évaluer leur compatibilité et leur cohérence avec les bases de connaissances existantes. Le processus d'évaluation, qui nécessite généralement beaucoup de calculs, est révolutionné par le développement d'un modèle prédictif qui évalue efficacement l'adéquation de ces axiomes en tant que substitut du raisonnement traditionnel. Ce modèle innovant réduit considérablement les exigences en matière de calcul, en utilisant le raisonnement comme un "oracle" occasionnel pour classer les axiomes complexes lorsque cela est nécessaire.

L'apprentissage actif joue un rôle essentiel dans ce cadre. Il permet à l'algorithme d'apprentissage automatique de sélectionner des données spécifiques pour l'apprentissage, améliorant ainsi son efficacité et sa précision avec un minimum de données étiquetées. La thèse démontre cette approche dans le contexte du Web sémantique, où le raisonneur joue le rôle d'"oracle" et où les nouveaux axiomes potentiels représentent des données non étiquetées.

Cette recherche contribue de manière significative aux domaines du raisonnement automatique et au-delà, en ouvrant de nouvelles possibilités dans des domaines tels

que la bio-informatique et la preuve automatique de théorèmes. En mariant efficacement l'apprentissage automatique et le raisonnement symbolique, ces travaux ouvrent la voie à des processus de découverte de connaissances plus sophistiqués et autonomes, annonçant un changement de paradigme dans la manière dont nous abordons et exploitons la vaste étendue de données sur le web sémantique.

Mots-clés

apprentissage automatique, raisonnement symbolique, axiomes, bottleneck de l'acquisition des connaissances, Web sémantique, apprentissage actif, ontologies, processus heuristiques, bases de connaissances, modèle prédictif, raisonnement computationnel, efficacité des algorithmes, données étiquetées, raisonnement automatisé, démonstration automatisée de théorèmes, découverte de connaissances.

ABSTRACT

This thesis addresses the challenge of evaluating candidate logical formulas, with a specific focus on axioms, by synergistically combining machine learning with symbolic reasoning. This innovative approach facilitates the automatic discovery of axioms, primarily in the evaluation phase of generated candidate axioms. The research aims to solve the issue of efficiently and accurately validating these candidates in the broader context of knowledge acquisition on the semantic Web.

Recognizing the importance of existing generation heuristics for candidate axioms, this research focuses on advancing the evaluation phase of these candidates. Our approach involves utilizing these heuristic-based candidates and then evaluating their compatibility and consistency with existing knowledge bases. The evaluation process, which is typically computationally intensive, is revolutionized by developing a predictive model that effectively assesses the suitability of these axioms as a surrogate for traditional reasoning. This innovative model significantly reduces computational demands, employing reasoning as an occasional "oracle" to classify complex axioms where necessary.

Active learning plays a pivotal role in this framework. It allows the machine learning algorithm to select specific data for learning, thereby improving its efficiency and accuracy with minimal labeled data. The thesis demonstrates this approach in the context of the semantic Web, where the reasoner acts as the "oracle," and the potential new axioms represent unlabeled data.

This research contributes significantly to the field of automated reasoning, and beyond, opening up new possibilities in areas like bioinformatics and automated theorem proving. By effectively marrying machine learning with symbolic reasoning, this work paves the way for more sophisticated and autonomous knowledge discovery processes, heralding a paradigm shift in how we approach and leverage the vast expanse

of data on the semantic Web.

Keywords

machine learning, symbolic reasoning, axioms, knowledge acquisition bottleneck, semantic Web, active learning, ontologies, heuristic processes, knowledge bases, predictive model, computational reasoning, algorithm efficiency, labeled data, automated reasoning, automated theorem proving, knowledge discovery.

ACKNOWLEDGMENTS

First and foremost i would like to thank my supervisors, Andrea G.B. Tettamanzi and Célia da Costa Pereira, for their patience, trust, advice, and support throughout this thesis. Thank you Andrea for your guidance, and helping me make this research the best it can be. The effort you put throughout this thesis is the reason I am able to consider myself a researcher. I appreciate you having given me this opportunity and always being there whenever my knowledge or experience was lacking. I thank you Célia, for every piece of advice you have given me and every note you placed on my papers. Nothing I wrote ever felt right to me until it got your approval.

I would like to express gratitude to each member of the jury, Fabien Gandon, Federico Ulliana, Salem Benferhat, and Natalie Hernandez for taking the time to review this thesis. I would like to thank Dario Malchiodi for taking the time introducing me to his work and painstakingly explaining things to me during my first year.

I am thankful for every member of the WIMMICS team I have met since I joined. I want to thank every member who has ever answered my many questions. I would also like to thank the team developing CORESE, and RDFMiner. This thesis would not have been possible without them.

Lastly, I thank my mother and father who are the reason I am who I am and where I am today. I thank my brother for being there, always. And I thank my wife and child for believing in me. Thank you to every one in my family who called me "Dr." before I even deserved it.

Contents

Page

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 Navigating the Logic Labyrinth: Why Axioms Matter

The ever-increasing reliance on structured knowledge representations, such as knowledge graphs and ontologies, has accentuated the need for automated evaluation of logical formulas. These formulas, including a special class known as axioms, define the rules, relationships, and constraints of a domain [204]. While substantial strides have been made in the automated handling of such formulas, the task remains predominantly manual and presents challenges in scalability, reliability, and efficiency [172, 280, 165].

Evaluating logical formulas, to acquire knowledge, is fundamental to the fields of robotics, artificial intelligence, formal logic, and knowledge representation [243, 135]. These evaluations are crucial in a variety of applications, ranging from automated theorem proving to database management [327, 169]. Despite its importance, the process often lacks transparency and relies heavily on non scalable exhaustive computational techniques or expert judgment [206, 87].

Among these logical formulas, axioms hold a unique position. They serve as foundational elements in structured knowledge systems and often require a higher degree of scrutiny and validation [339, 293]. Existing machine learning techniques for formula evaluation have shown promise but often fall short in integrating semantic understanding, which is particularly crucial for the nuanced evaluation of axioms [284, 281].

As knowledge structures continue to find new applications and grow in complexity,

the need for automated and scalable methods for formula evaluation, including axioms, becomes increasingly urgent [17, 206]. This thesis aims to address this pressing need by employing active learning techniques for the discovery and evaluation of logical formulas, with a special emphasis on axioms. Our approach seeks to contribute to more robust, transparent, and adaptive systems.

1.2 Evaluating Logical Formulas Through Machine Learning

1.2.1 Problem Statement

In the realm of semantic Web and knowledge acquisition, the efficient evaluation of candidate logical formulas, particularly axioms, remains a significant challenge. Traditional approaches to validating these axioms often involve computationally intensive reasoning processes, which can be impractical for large-scale knowledge bases. The integration of machine learning with symbolic reasoning proposes a promising avenue to revolutionize this process. This research seeks to address the bottleneck in knowledge acquisition by developing a method to efficiently and accurately evaluate candidate axioms using observed facts, thereby facilitating their automatic discovery.

1.2.2 Research Questions

The study is driven by the following key research questions:

1. *How can machine learning be effectively integrated with symbolic reasoning to improve the evaluation process of candidate logical formulas, especially axioms?*

    - This question explores the synergy between machine learning algorithms and symbolic reasoning techniques to create a more efficient evaluation process for axioms.

2. *What role do observed facts play in the evaluation of candidate axioms, and how can they be optimally utilized in this context?*

    - This question aims to understand the significance of observed facts in the evaluation process and seeks to identify the best practices for their application.

3. *Can a predictive model be developed to assess the compatibility and consistency of candidate axioms with existing knowledge bases, and how can this model reduce the computational demands of traditional reasoning?*

    - The focus here is on creating a predictive model that can accurately assess candidate axioms, thereby reducing the reliance on computationally heavy traditional reasoning methods.

4. *How can we ensure the scalability of these models, especially when dealing with large and complex knowledge representations?*

    - This question delves into optimizing machine learning models for scalability, ensuring they effectively handle large and complex knowledge bases without performance loss. The aim is to develop robust models that maintain efficiency and accuracy as the scale and intricacy of data increase.

5. *How can active learning be employed to enhance the efficiency and accuracy of the machine learning algorithm in the context of axiom evaluation?*

    - This question investigates the application of active learning strategies to improve the machine learning algorithm, particularly in terms of efficiency and accuracy in the evaluation of axioms.

6. *What are the implications of this research for the fields of automated reasoning, natural language processing, bioinformatics, and automated theorem proving?*

    - This question seeks to explore the broader impact and potential applications of the research in various related fields.

The answers to these questions aim to contribute significantly to the field of knowledge discovery, particularly in the semantic Web, by paving the way for more sophisticated and autonomous axiom evaluation processes.

1.3 Contributions of this Thesis

This thesis is organized as follows; Chapter 2 provides a comprehensive overview of all foundational elements used in this thesis. Chapter 3 presents the state of the art on formula evaluation using machine learning techniques. Chapter 4 introduces our novel propositional logic formula evaluating model that is built using our novel model-theoretic similarity measure. Chapter 5 focuses on predicting the score of atomic candidate OWL class axioms, while Chapter 6 extends this focus to predicting the acceptability of these axioms. Chapter 7 introduces the concept of scalability, offering a vector-space dimension-reduced approach for formula evaluation. Chapter 8 presents an OWL Class Axiom Score Predictor that incorporates active learning techniques and handles complex concepts. The final technical chapter, Chapter 9, provides a novel and powerful model based on our model-theoretic similarity measure and capable of dealing with complex axioms with the ability of being extended. Chapter 10 concludes the thesis, summarizing the contributions and suggesting avenues for future research.

1.4 Published results

The results of this thesis were published or submitted in several international venues:

- Ballout, A., da Costa Pereira, C., Tettamanzi, A.G.B.: Learning to classify logical formulas based on their semantic similarity. In: Aydogan, R., Criado, N., Lang, J., Sánchez-Anguix, V., Serramia, M. (eds.) PRIMA 2022: Principles and Practice of Multi-Agent Systems - 24th International Conference, Valencia, Spain, November 16-18, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13753, pp. 364–380. Springer (2022). doi: 10.1007/978-3-031-21203-1_22

- Ballout, A., Tettamanzi, A.G.B., Da Costa Pereira, C.: Predicting the score of atomic candidate OWL class axioms. In: 2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). pp. 72–79 (2022). doi: 10.1109/WI-IAT55865.2022.00020

- Ballout, A., da Costa Pereira, C., Tettamanzi, A.G.B.: Predicting the acceptability of atomic candidate OWL class axioms. In: 2023 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). pp. 339–345 (2023). doi: 10.1109/WI-IAT59888.2023.00055

- Ballout, A., da Costa Pereira, C., Tettamanzi, A.: Scalable prediction of atomic candidate OWL class axioms using a vector-space dimension reduced approach. In: Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART. pp. 347–357. INSTICC, SciTePress (2024). doi: 10.5220/0012384200003636

- Ballout, A., da Costa Pereira, C., Tettamanzi, A.: OCASP OWL Class Axiom Score Predictor with Active learning. (2024)

Chapter 2

BACKGROUND

The field of automated logical formula evaluation intersects multiple disciplines, each contributing vital methods and perspectives that inform current approaches. This chapter aims to provide a comprehensive overview of the key areas of knowledge and techniques that underpin this interdisciplinary domain. We begin by exploring the foundational concepts in knowledge representation, including Knowledge Graphs, RDF, Ontologies, and OWL. Next, we delve into the logical frameworks that serve as the basis for formula representation and evaluation, such as propositional and description logics. Vector space approaches, often used for comparing and evaluating formulas, are discussed subsequently. The chapter also covers various formula generation methods, followed by traditional formula evaluation techniques. The use of machine learning has recently emerged as a promising approach to formula evaluation, offering the benefits of scalability and adaptability. We review general machine learning techniques commonly used in the field. This overview serves as a prelude to the challenges and contributions presented in the remainder of this thesis.

2.1 Knowledge Representation

Knowledge representation serves as core constituent of any intelligent system, providing the foundational structures upon which complex reasoning tasks are performed. Over the years, various frameworks and paradigms have been developed to efficiently represent, store, and manipulate knowledge. This section aims to cover some of the most widely-used knowledge representation schemes in the context of automated formula evaluation.

We begin by discussing Knowledge Graphs, a flexible and intuitive representation model that has found applications in a wide range of domains from natural language processing to search engines [9, 43, 321]. Following this, we delve into the Resource Description Framework (RDF), a standard model for data interchange that has been endorsed by the World Wide Web Consortium (W3C) [296, 272].

Ontologies represent another critical aspect of knowledge representation, often serving as the backbone for complex knowledge-based systems. We examine the role of ontologies in providing a shared and common understanding of a domain [271]. Finally, we explore the Web Ontology Language (OWL), a semantic markup language for publishing and sharing ontologies on the World Wide Web [367, 366].

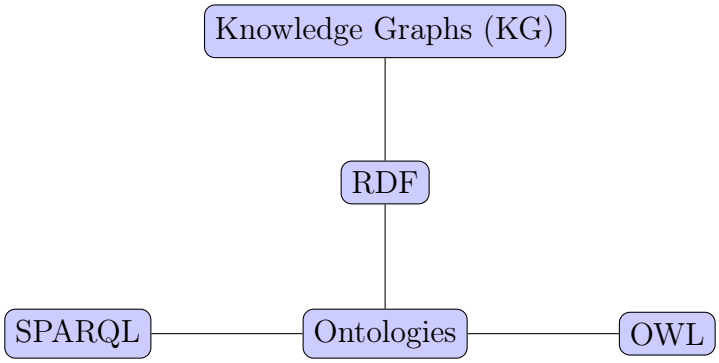
Each of these knowledge representation methods offers unique advantages and limitations, especially when it comes to the evaluation of logical formulas. The subsequent subsections provide a detailed examination of these paradigms, setting the stage for their application in automated formula evaluation. Figure 2.1 represents the components of this section and their interactions.

2.1.1 Knowledge Graphs and RDF

Knowledge Graphs (KGs) have emerged as a powerful tool for organizing and utilizing information in a multitude of applications, ranging from enhancing search engines to powering question-answering systems and product recommendation engines [167, 174]. At their core, KGs represent facts in the form of triples, specifically *(subject, predicate, object)*. In this representation, the *subject* and *object* denote entities from the real world, while the *predicate* serves as the relationship connecting these entities.

The essence of a KG lies in its graph-based structure, where nodes represent entities and edges represent relationships between them. This allows for a highly

Figure 2.1: Detailed components and interactions of the Knowledge Representation section.



Node	**Description**
Knowledge Graphs (KG)	Graphical representation of structured knowledge
RDF	Resource Description Framewor a standard for web-based KGs
Ontologies	Formal naming and definition of types, properties, and interrelationships
OWL	Web ontology language based on Description Logic
SPARQL	Query language for RDF

flexible and rich representation of complex interconnected data. Unlike traditional databases, the relationships between nodes in a KG can vary, offering the capacity to capture the nuanced interactions between entities [166].

The Resource Description Framework (RDF) [89] serves as one of the foundational technologies for representing KGs on the web. It provides a standardized model for data interchange and has been widely adopted for constructing KGs, especially in the context of the Semantic Web. The Semantic Web itself is an extension of the World Wide Web and aims to make data across the Web understandable and usable by machines. It uses a variety of standards, including RDF, to achieve this goal [368].

Linked Data, a subset of KGs, offers a specific method for publishing structured

data on the web, making it both human-readable and machine-readable [42]. One of the core principles of Linked Data is that it should be freely accessible and non-proprietary, thereby promoting open access to information. This philosophy aligns well with the broader goals of the Semantic Web and has implications for how KGs are used in various applications [3, 120, 294].

The ability of KGs to represent complex relationships in a structured yet flexible manner makes them highly valuable for tasks that require a deep understanding of entities and their interconnections. This includes, but is not limited to, natural language processing [315], social network analysis [108], and, pertinent to this thesis, the evaluation of logical formulas [30, 72, 181].

2.1.2 Ontologies and OWL

Ontologies serve as a cornerstone in the realm of knowledge representation, providing a formal framework for organizing and sharing domain knowledge [361]. Unlike databases or Knowledge Graphs, ontologies go beyond merely storing data; they capture the semantics, or the "meaning," of the data. This is achieved by defining the relationships, constraints, and rules for the entities within a particular domain [84].

An ontology typically consists of two main components: the TBox and the ABox. The TBox (Terminological Box) contains the terminological data, defining the classes and the relationships between them. It specifies the ontology's schema, including class hierarchies and property constraints [93, 304]. The ABox (Assertional Box), on the other hand, contains the assertional data that populates the ontology. It consists of instances of the classes defined in the TBox and specifies the relationships between these instances [93, 304].

Together, the TBox and ABox allow ontologies to capture complex relationships and rules, making them highly expressive and adaptable to various domains [369].

They offer advanced features like class inheritance, property restrictions, and logical reasoning, which are essential for tasks that require a deep understanding of entity relationships and constraints, such as the automated evaluation of logical formulas [207, 204].

The power of ontologies is perhaps best demonstrated through the Web Ontology Language (OWL). OWL is a semantic markup language designed to represent rich and complex knowledge about things, groups of things, and the relations between them [94, 389]. It extends the foundational principles of RDF and builds upon them to offer a more expressive framework, allowing for characteristics like data types, classes, properties, and more [57].

Due to the evolving needs of complex applications and to address limitations in the original OWL standard, OWL 2 was introduced [365]. OWL 2 provides several enhancements over its predecessor, including new data types, property chain inclusions, and richer expressivity features like keys and qualified cardinality restrictions [366, 254]. These extensions make OWL 2 more adaptable to a wider range of problems and more efficient in terms of reasoning [298, 332].

Both OWL and OWL 2 offer various profiles tailored for different use-cases. For example, OWL DL is optimized for reasoning, whereas OWL Lite aims for simpler classifications and constraints. Similarly, OWL 2 has profiles like OWL 2 EL, which is designed for ontologies with large numbers of properties and classes [195].

The integration of ontologies and OWL standards into knowledge systems provides a robust framework for semantically rich knowledge representation. This is especially valuable in scenarios requiring an in-depth understanding of complex entity relationships, such as the evaluation of logical formulas [91]. Their role in capturing complex semantics makes them indispensable tools in the research and development of automated formula evaluation methods.

2.1.3 Querying with SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) serves as the standard query language for querying RDF-based data stores, including Knowledge Graphs and ontologies [372]. It provides the means to extract, manipulate, and modify data within RDF triple stores, making it an indispensable tool in the realm of knowledge representation and Semantic Web technologies [157, 317].

The language offers various forms of queries, such as 'SELECT', 'ASK', 'DESCRIBE', and 'CONSTRUCT', each designed for specific types of data retrieval and manipulation [157]. SPARQL also supports more advanced features like federation, allowing for queries distributed over multiple data stores [8, 82, 121].

One of the significant advantages of SPARQL is its ability to perform semantic queries. This means it can understand the underlying semantics defined in ontologies and can execute complex queries that take into account the relationships, constraints, and rules defined in the ontology. This capability is particularly useful for tasks like automated logical formula evaluation, where understanding the semantic context can be critical.

As RDF and OWL constructs are often used in the creation and representation of Knowledge Graphs and ontologies, SPARQL serves as a natural fit for querying these structures. Its flexibility and expressiveness make it an ideal choice for a wide range of applications, from simple data retrieval to complex semantic reasoning tasks [50, 60, 136, 147, 251, 385].

Alongside the core functionalities of SPARQL, the effectiveness of this query language is further amplified through various SPARQL engines and endpoints. These engines are specialized software systems designed to process SPARQL queries and return results from RDF data stores. Some notable SPARQL engines include Apache

Jena Fuseki, Virtuoso, and Blazegraph, each offering unique features and optimizations. Apache Jena Fuseki is renowned for its compliance with SPARQL standards and ease of integration with Jena-based applications [19, 61]. Virtuoso, on the other hand, is recognized for its high performance and scalability, particularly in dealing with large datasets [115]. Blazegraph has gained attention for its high-speed query processing, especially in graph analytics and network data [347, 351].

SPARQL endpoints play a crucial role in facilitating access to RDF data over the web. These endpoints, essentially Web services, allow users to submit SPARQL queries and retrieve results via standard HTTP requests. Public SPARQL endpoints have become a valuable resource in the Semantic Web community. Examples include DBpedia's SPARQL endpoint, which provides access to a vast array of structured data extracted from Wikipedia [214], and Wikidata Query Service, offering an extensive query interface for Wikidata's open knowledge base [364]. The Linked Open Data cloud also hosts numerous SPARQL endpoints, enabling users to explore and interlink a variety of datasets across different domains [46, 226].

These engines and endpoints exemplify the practical implementation of SPARQL in diverse contexts. They not only serve as crucial tools for querying RDF data but also demonstrate the adaptability and reach of SPARQL in handling web-based, distributed, and large-scale datasets. Their continuous evolution and improvement reflect the growing importance of SPARQL in the realm of data querying and the Semantic Web.

In summary, SPARQL provides a robust framework for querying RDF-based knowledge structures, supporting everything from basic data retrieval to advanced semantic queries. Its role is crucial for any research or application that relies on intricate querying of Knowledge Graphs, ontologies, or any RDF-based data structures, including the field of automated formula evaluation [225].

2.2 Formal Logic and Logical Formulas

Formal logic plays an instrumental role in the Computer Science and Artificial Intelligence, offering a rigorous mathematical foundation for reasoning about truth and falsehood [292]. While it originates from the philosophical study of reasoning and argumentation, formal logic has found extensive applications in modern computational systems, including knowledge representation, database querying, and automated theorem proving [45, 378, 137, 250, 327]. It provides the building blocks for constructing logical formulas, which are mathematical expressions that encapsulate specific statements or conditions [227, 311, 112, 113, 323].

In the context of this research, understanding the nuances of formal logic is crucial, as the primary focus revolves around the automated evaluation of logical formulas. These formulas can range from simple propositional statements to more complex quantified expressions, each with its own set of rules, syntax, and semantics. The ability to accurately evaluate these formulas is at the core of various technologies and applications, including but not limited to, knowledge graphs, ontologies, and intelligent systems [77, 253, 327].

The following subsections will offer a detailed examination of various types of formal logic systems, such as propositional logic, first-order logic, and description logic. Table 2.1 provides an overview of Formal Logic Systems used in this thesis including Extensions, Operators, and Sample Formulas. Special attention will be given to OWL profiles, which are subsets of the OWL language optimized for specific computational and reasoning tasks. We will also introduce Herbrand semantics, a key concept in understanding the semantics of predicate logic, and its implications for logical formula evaluation. Finally, the section will explore the critical role of logical formulas and axioms in structured knowledge systems [338, 179, 118, 117].

By the end of this section, the reader should have a robust understanding of the various logical systems and how they relate to the task of automated formula evaluation. This foundational knowledge will be pivotal for appreciating the challenges and methods discussed in later chapters.

Logic System	**Extensions**	**Operators**	**Sample Formula**
Propositional Logic	N/A	$\wedge, \vee, \neg$	$p \wedge \neg q$
First-Order Logic	N/A	$\wedge, \vee, \neg, \forall, \exists$	$\forall x : P(x) \vee \neg Q(x)$
Description Logic	SHOIQ, SHOIN	$\neg, \sqcap, \sqcup, \sqsubseteq, \equiv$, $\forall R.C, \exists R.C, etc.$	$A \sqcap \exists R.B$
OWL (Class Axioms)	OWL2	Object Properties, Class Expressions, etc.	Human $\equiv$ Animal $\sqcap$ $\exists$hasLegs.Two

Table 2.1: Overview of Formal Logic Systems used in this thesis including Extensions, Operators, and Sample Formulas.

2.2.1 Propositional Logic

Propositional logic, also known as zeroth-order logic, forms the foundation upon which more complex logical systems are built [297]. At its core, propositional logic deals with propositions, which are statements that can be either true or false, but not both. These propositions are manipulated using logical connectives such as 'AND' ($\wedge$), 'OR' ($\vee$), 'NOT' ($\neg$), 'IMPLIES' ($\rightarrow$), and 'EQUIVALENT' ($\leftrightarrow$) to form compound propositions or logical formulas [158].

One of the key merits of propositional logic is its simplicity. The syntax and

semantics are straightforward, making it easier to implement automated reasoning systems based on it [130]. However, the simplicity comes at a cost: propositional logic is not expressive enough to capture more complex relationships between entities or to represent statements that involve quantifiers, such as 'for every' or 'there exists' [386].

In automated formula evaluation, propositional logic often serves as a starting point or a simplified model for testing algorithms and methods [158]. While it may not capture the richness of the domain being modeled, it offers a computationally less intensive way to evaluate the validity and soundness of logical formulas [354].

Despite its limitations, propositional logic provides the building blocks for higher-order logics and specialized logical systems like description logic, thereby maintaining its relevance in the study and application of formal logic [384].

2.2.2 First-Order Logic

First-Order Logic (FOL), often referred to as predicate logic or predicate calculus, extends the capabilities of propositional logic by introducing quantifiers and predicates [67, 309]. While propositional logic deals with atomic propositions that are either true or false, FOL provides a means to express more complex relationships by allowing quantification over variables and the use of predicates to represent properties of or relationships between entities [160].

The primary symbols introduced in FOL are the universal quantifier $\forall$ (read as "for all") and the existential quantifier $\exists$ (read as "there exists"). These quantifiers allow for the expression of general statements like "for all humans, they are mortal" or specific claims like "there exists a human who is a philosopher" [283].

Predicates, on the other hand, are statements whose truth value depends on their arguments. For example, the predicate $\text{IsHuman}(x)$ might be true if x is a human and

false otherwise. By combining predicates with quantifiers, FOL can express intricate statements about the world that are beyond the expressiveness of propositional logic [141].

First-Order Logic is one of the most widely studied and used forms of logic in computer science, artificial intelligence, and formal methods [53]. Its expressiveness makes it suitable for a variety of tasks, from database querying and formal verification to natural language processing and knowledge representation [230, 343, 232, 344, 375, 132].

However, the increased expressiveness of FOL comes with computational challenges. Certain problems in FOL, such as determining the validity of a formula, are undecidable, and even those that are decidable can be computationally intensive [39, 31, 183]. Despite these challenges, FOL remains a cornerstone in the realm of formal logic and a crucial tool for researchers and practitioners aiming to model and reason about the world in a rigorous manner [22, 205, 131, 362].

2.2.3 Description Logic and OWL Profiles

Description Logic (DL) is a family of formal knowledge representation languages, tailored for expressing and reasoning about domain knowledge [26]. While grounded in the principles of First-Order Logic (FOL), DL is designed with specific constructs that ensure decidable reasoning over ontologies, making it particularly suitable for applications where tractability is of utmost importance [217, 274, 357].

Over the years, DL has seen various extensions, each enhancing its expressive power to cater to different modeling needs. For instance:

- $\mathcal{ALC}$: This is the foundational DL, encompassing basic constructs like conjunction, negation, and universal quantification [24].

- $\mathcal{SHOIN}$: Building upon $\mathcal{ALC}$, this extension introduces transitive roles, inverse roles, and number restrictions, forming the basis for OWL DL [143].

- $\mathcal{SHOIQ}$: A further enhancement of $\mathcal{SHOIN}$, $\mathcal{SHOIQ}$ incorporates qualified number restrictions [143].

The evolution of these DL extensions represents a careful balance between expressiveness and computational complexity. As we progress from basic variants like $\mathcal{ALC}$ to more expressive ones such as $\mathcal{SHOIQ}$, the reasoning tasks inherently become more resource-intensive, demanding more sophisticated algorithms and tools [25].

The Web Ontology Language (OWL) and its refined version, OWL2, are deeply rooted in DL, devised to encapsulate rich and intricate knowledge structures [196]. Both OWL variants offer distinct profiles, optimized for various reasoning tasks and computational trade-offs. For instance, OWL2 introduces profiles like OWL 2 EL, OWL 2 QL, and OWL 2 RL, each tailor-made for specific application scenarios and performance considerations [191].

Together, DL and its extensions, in conjunction with OWL, remain instrumental in modern knowledge representation, fostering robust and sophisticated modeling and reasoning mechanisms for diverse domains [59, 235, 224, 56].

2.2.4 Model Theoretic Semantics

Model theoretic semantics, a branch of mathematical logic, offers a framework for interpreting and understanding the meaning of logical formulas based on their models [177, 307]. It is a cornerstone in the field of formal semantics, underpinning the interpretation of languages, from natural languages to programming and logical languages.

In model theoretic semantics, the meaning of a sentence or formula is determined

by its truth-value in a model. A model, in this context, is a structured set that includes a domain of individuals and an interpretation function that assigns meanings to the symbols in the language. This approach allows for a rigorous and formal analysis of the semantics of logical statements, enabling the evaluation of their truth or falsity under various interpretations [15, 40].

Example: Consider the logical statement "All swans are white." In model theoretic semantics, this statement is evaluated in the context of a model that defines what 'swans' and 'white' refer to, and what it means for a swan to be white. Different models might lead to different evaluations of the statement's truth-value, reflecting the variability in interpretation inherent in language.

Model theoretic semantics is particularly relevant in the fields of computer science and artificial intelligence, especially in logical formula evaluation. It provides a systematic way to deal with the semantics of programming languages, enabling the analysis of program correctness, and the design of language features. In artificial intelligence, it aids in the interpretation of knowledge representations and the development of reasoning algorithms that operate based on the meanings assigned to symbols and constructs [312, 203, 162].

In conclusion, model theoretic semantics offers a powerful and versatile tool for understanding and interpreting logical formulas across various domains. Its application in the evaluation of logical formulas is of particular importance in this research, providing a robust framework for dealing with the complexities and nuances inherent in formal languages and systems [303, 193].

2.2.5 Herbrand Semantics

Herbrand Semantics, named after Jacques Herbrand, is a foundational concept in the realm of predicate logic and automated theorem proving. This semantics provides

a method to represent and reason about the models of predicate logic formulas without delving into the intricacies of the underlying domain or universe as is the case with tarksian semantics [289]. In essence, it reduces the complexity of reasoning in predicate logic by constraining the universe to a set of terms derived from the formula itself.

The central components of Herbrand Semantics are the Herbrand Universe and the Herbrand Base [163]:

- **Herbrand Universe (H_U)**: This refers to the set of all ground terms that can be formed using the function symbols and constants present in a given formula. If the formula has no constants, a special constant is introduced to ensure a non-empty universe.

- **Herbrand Base (H_B)**: Comprising the set of all possible ground atoms that can be constructed using the predicates and terms from the Herbrand Universe, the Herbrand Base represents the potential atomic sentences of the formula.

An essential property of Herbrand Semantics is that if a first-order formula has a model, then it also has a Herbrand model, which is a model whose domain is the Herbrand Universe of the formula [209, 247]. This property is pivotal for automated theorem proving, as it allows for the transformation of first-order logic reasoning tasks into propositional logic tasks, which are computationally more tractable.

In the broader context of formula evaluation and theorem proving, Herbrand Semantics plays a pivotal role. By providing a mechanism to ground complex logical formulas into a finite and more manageable set of terms and atoms, it facilitates more efficient reasoning and proof search, especially in automated theorem proving systems [111].

Herbrand Semantics, with its unique approach to modeling and reasoning, offers

valuable insights and tools for researchers and practitioners aiming to tackle intricate problems in formal logic, knowledge representation, and automated reasoning [314, 83].

2.2.6 Fuzzy Logic

Fuzzy logic, a form of many-valued logic, extends classical logic by introducing the concept of partial truth — truth values between "completely true" and "completely false" [382]. It was developed by Lotfi Zadeh in the 1960s as a way to model the uncertainty and vagueness inherent in human reasoning. Fuzzy logic provides a mathematical framework for dealing with imprecise information, making it particularly useful in fields where binary distinctions (true/false, yes/no) are not significant.

At its core, fuzzy logic is characterized by its use of fuzzy sets, which are classes with a continuum of grades of membership. This contrasts with classical set theory, where an element either belongs to a set or does not. In fuzzy set theory, membership is expressed with a degree ranging from 0 to 1, allowing for a more nuanced representation of concepts [186].

Example: Consider the concept of "tallness." In classical logic, a person might be classified as either "tall" or "not tall." Fuzzy logic, however, allows for degrees of tallness. A person might have a membership value of 0.8 in the set of "tall people," reflecting their relative height in a more flexible manner.

The methods of fuzzy logic involve the manipulation of these fuzzy sets and the application of fuzzy operators (like AND, OR, NOT) that handle partial truth values. Fuzzy logic systems often employ a set of rules (fuzzy if-then rules) to describe how to process inputs and compute outputs, making them highly adaptable for various applications [241, 88, 288].

In practical applications, fuzzy logic has been widely adopted in control systems,

decision-making, pattern recognition, and artificial intelligence. For instance, it is used in household appliances like washing machines and air conditioners to provide more efficient and human-like control. In artificial intelligence, fuzzy logic enhances the ability to make decisions based on vague or incomplete information, mimicking human reasoning more closely [276, 329, 358]

In the context of logical formula evaluation, fuzzy logic offers a valuable approach for handling uncertainty and ambiguity. It allows for the evaluation of formulas under conditions where information is incomplete or imprecise, providing a more realistic and flexible framework compared to traditional binary logic systems. This flexibility is particularly useful in areas such as knowledge representation and natural language processing, where the binary true/false paradigm is often too restrictive to capture the subtleties of human language and thought [65, 81].

Overall, fuzzy logic provides an essential toolset for dealing with complexity and uncertainty in various domains. Its ability to model degrees of truth and handle imprecise information makes it a powerful approach in both theoretical and practical applications, offering significant advantages in fields ranging from engineering to artificial intelligence [134].

2.2.7 Logical Formulas and Axioms

In the vast domain of formal logic and knowledge representation, logical formulas serve as the fundamental constructs that allow for the expression, modeling, and reasoning of propositions, facts, and relationships [339, 340]. These formulas, represented in symbolic form, embody the core semantics and syntax of a logical system, providing a formal mechanism to capture and deduce knowledge [252].

A logical formula typically consists of [158, 322]:

- **Atoms**: The basic units that represent propositional variables or predicate

symbols applied to terms.

- **Connectives**: Symbols such as $\wedge$ (AND), $\vee$ (OR), and $\neg$ (NOT) that allow for the composition of complex formulas from simpler ones.

- **Quantifiers**: Symbols like $\forall$ (for all) and $\exists$ (there exists) which enable the expression of general statements about a domain.

Building upon logical formulas, axioms hold a special place in formal systems. Axioms are foundational formulas that are accepted as true without requiring proof within the system [293]. They set the groundwork, serving as starting points upon which other truths (theorems) can be derived using rules of inference. In essence, axioms are the bedrock principles or assumptions of a logical system or theory [44].

Transitioning from traditional logical systems to the realm of ontology languages, OWL (Web Ontology Language) introduces its own set of axioms tailored for knowledge representation and reasoning in the Semantic Web [94, 389]. These OWL axioms define classes, properties, and their interrelationships within a domain.

Focusing on OWL Class axioms, they serve as the building blocks of ontological modeling [254]. An OWL Class axiom typically comprises:

- **Class Expressions**: Represent sets of individuals (entities) sharing common characteristics. They can be atomic or complex, involving constructs like union, intersection, and complement.

- **Property Expressions**: Describe the relationships between individuals or between individuals and data values. These can be object properties, data properties, or annotation properties.

- **Constraints**: Define conditions or restrictions on class or property expressions, such as cardinality constraints, disjointness, and equivalence.

The expressiveness of OWL, especially when modeling Class axioms, allows for the capture of intricate domain knowledge, ranging from simple categorizations to complex hierarchical relationships and constraints [110].

In the context of automated reasoning and formula evaluation, understanding both traditional logical formulas and specialized constructs like OWL Class axioms is indispensable. As this thesis delves into the realm of evaluating such formulas, especially axioms, using machine learning techniques, a solid grasp of their intricacies, semantics, and significance is crucial for appreciating the challenges and contributions of the subsequent chapters.

2.3 Formula Generation Methods

In the domain of knowledge representation and reasoning, the task of generating logical formulas plays a pivotal role in defining, constraining, and extending knowledge bases and ontologies [184, 393]. As structured knowledge repositories grow in size and complexity, the manual crafting of logical formulas becomes a daunting, if not impossible, endeavor. This underscores the need for automated methods to generate these formulas efficiently and accurately.

Automated formula generation not only accelerates the development of knowledge bases but also aids in tasks like hypothesis testing, knowledge discovery, and ontology enrichment [346, 246, 341]. Various methods, ranging from deterministic generative techniques to probabilistic and evolutionary algorithms, have been proposed to tackle this challenge [173, 285, 114, 244]. This section delves into the prominent methods used for formula generation, highlighting their underlying principles, advantages, and limitations. Figure 2.2 shows a simple overview of formula generation methods.

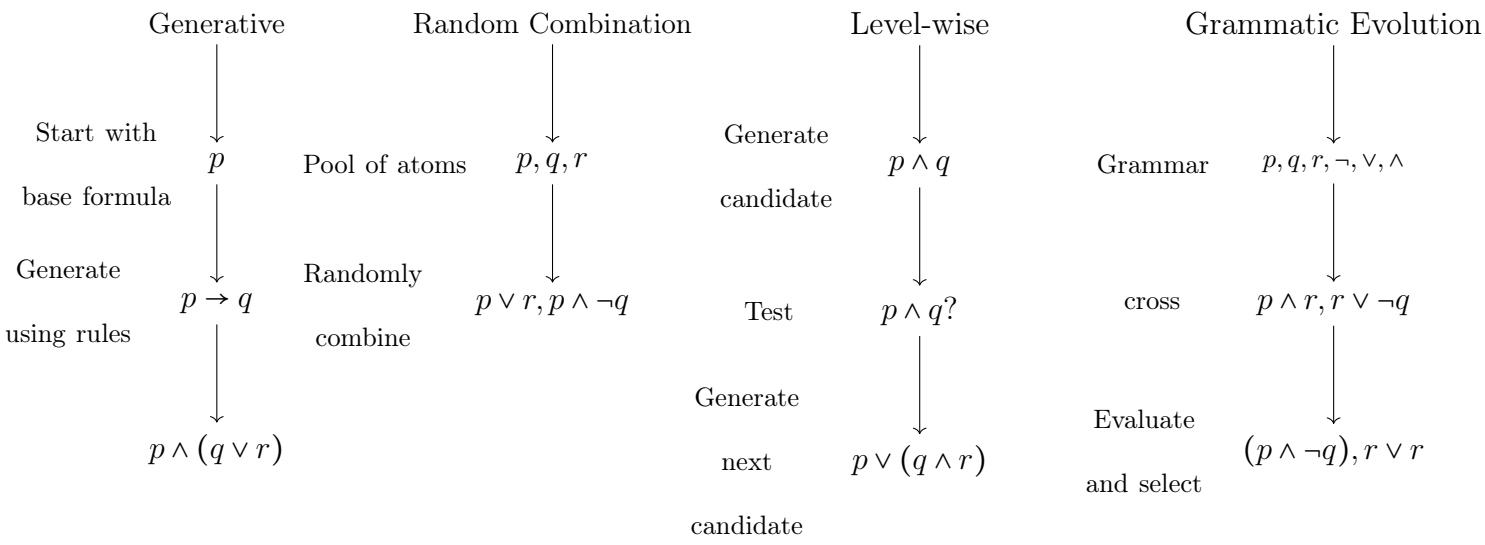


Figure 2.2: Overview of formula generation methods.

2.3.1 Generative Methods

Generative methods serve as foundational techniques in the automated generation of logical formulas. At their core, these methods rely on predefined rules, patterns, or templates to produce formulas based on the existing knowledge within a domain [194, 256]. By exploiting the inherent structure and semantics of the knowledge base or ontology, generative methods can create formulas that are both syntactically correct and semantically meaningful [256, 5].

The primary advantage of generative methods is their deterministic nature, ensuring consistency and repeatability in the generated formulas [182]. However, their deterministic approach can also be a limitation when exploring novel or unexpected relationships within the knowledge domain. This is because they strictly adhere to the predefined patterns and may not capture emerging patterns or anomalies [208, 335].

In practice, generative methods are often combined with other techniques, such as probabilistic or evolutionary algorithms, to balance their deterministic nature with the flexibility of more adaptive approaches [103].

2.3.2 Random Combination

Random combination methods introduce an element of stochasticity in formula generation. Instead of strictly adhering to predefined patterns, these methods randomly combine elements from a set of logical symbols, entities, and predicates to form potential formulas [185, 301]. The aim is to explore a vast space of possible formulas that might not be immediately evident through deterministic methods.

While this approach offers a higher degree of exploratory freedom, it also poses challenges. The sheer number of possible combinations can lead to a plethora of formulas, many of which might be nonsensical or irrelevant to the domain [70, 301]. To counteract this, constraints or heuristics are often employed to guide the random generation process, ensuring that the produced formulas adhere to certain desired properties or characteristics [185].

Random combination methods are particularly useful in scenarios where the knowledge domain is not well-defined, or when there is a need to discover unexpected relationships and rules. However, due to their stochastic nature, the results might vary across different runs, necessitating further validation and refinement of the generated formulas.

2.3.3 Evolutionary Methods

Evolutionary methods adopt principles from natural evolution, such as mutation, crossover, and selection, to optimize and generate logical formulas [306]. Contrary to systematic Level-wise Generate and Test approaches, Evolutionary methods are heuristic and stochastic. They make use of the notions of mutation and crossover to manipulate the complexity of formulas. The generated formulas undergo testing at each level, ensuring that only relevant and valid formulas progress to the next

level, thereby enabling a guided yet probabilistic exploration of the formula space [342, 176, 64].

Grammatic Evolution (GE) stands out as a particularly intriguing form of genetic programming. In GE, formulas are represented as linear genomes, which are then mapped to production rules dictated by a specific grammar. This ensures that the generated formulas maintain syntactic correctness and adhere to the structure of the target logic system. One implementation of GE tailored for generating OWL axioms is detailed in [268]. Here, OWL axioms are evolved by employing a two-stage process: an initial generation phase followed by an evolutionary phase. Genomes undergo mutation, crossover, and selection operations, driven by a fitness function that evaluates the quality and relevance of the axioms.

The fitness scoring process, which evaluates the generated formulas, plays a pivotal role in guiding the evolutionary search towards more meaningful and accurate constructs. Efficient and accurate formula evaluation can significantly expedite the formula generation process. This is especially crucial in domains with vast search spaces, where the computational cost of evaluating a large number of candidate solutions can be prohibitive [146, 210, 374, 390].

Both Level-wise and GE approaches offer unique strengths. While Level-wise provides a structured progression, GE's flexible exploration of the solution space, coupled with the significance of the fitness function, makes it particularly suited for intricate domains [212, 268, 64]. The need for effective formula evaluation techniques, as highlighted in GE's fitness scoring process, underscores the importance of the upcoming sections, which delve deeper into formula evaluation methods.

2.4 Formula Evaluation Techniques

To streamline and enhance decision-making processes in various scientific and engineering domains, we require the automated generation of logical formulas. Automatically generating logical formulas allows for dynamic systems adaptation, automated reasoning, and supports artificial intelligence systems in performing complex tasks [291, 62]. Additionally, it addresses the scalability issues associated with manually crafting these formulas, making it possible to handle large datasets and complex decision-making scenarios efficiently. Achieving this makes computational models that can autonomously derive meaningful insights from vast amounts of data possible.

The automated generation of logical formulas, whether through evolutionary methods or other means, necessitates a robust evaluation mechanism to gauge their validity, relevance, and quality. Evaluating the quality of a formula is a multifaceted task, encompassing various metrics and methods. While the syntactic correctness of a formula can be easily verified, its semantic relevance, applicability, and utility within a specific domain or context are more challenging to ascertain. This section delves into various techniques employed to evaluate logical formulas, highlighting their strengths, limitations, and applicability. At the end of the section, we provide a summarized comparison of the different evaluation methods in Table 2.2.

2.4.1 Manual Expert Evaluation

Manual expert evaluation holds a pivotal position in the assessment of logical formulas. Trusted for its accuracy, this method relies on domain experts to gauge the quality, relevance, and validity of a formula within a specific context [261]. Experts bring domain-specific knowledge, intuition, and experience that can significantly enhance the evaluation process.

The process typically involves presenting the generated formulas to experts who then assess them based on predefined criteria. These criteria may encompass the formula's syntactic correctness, semantic relevance, consistency with existing knowledge, and potential utility in applications [295]. Moreover, experts can identify nuances and subtleties in the formulas that might be overlooked by automated evaluation methods.

While manual expert evaluation offers unparalleled accuracy, it has its challenges. The process can be time-consuming, especially when evaluating a vast number of formulas. Subjectivity is another concern, as different experts might have varying opinions on the same formula. Additionally, the limited availability of domain experts can pose scalability challenges for extensive evaluations [229].

Nevertheless, manual expert evaluation serves as a valuable benchmark. It is often used alongside automated methods to validate their results and provide insights into areas of improvement [261, 295, 229]. In this thesis, manual expert evaluation serves as the golden standard and ground truth against which methods are benchmarked. This ensures the reliability of our developed techniques and provides a comprehensive assessment framework. By comparing the outcomes of automated evaluations with expert judgments, the thesis identifies discrepancies and refines the computational models accordingly. This method enhances the validity of the research findings and provides a solid foundation for future explorations.

2.4.2 Data-Driven Approaches

In the realm of logical formula evaluation, data-driven approaches primarily encompass methods rooted in statistics, probability, and measures of uncertainty or possibility. These methods leverage empirical evidence, often in the form of instance data, to make informed judgments about the validity or acceptability of a given formula.

Statistical Methods: Statistical techniques utilize data to infer the likelihood of a formula being true. By analyzing the distribution and frequency of certain patterns or occurrences in the data, these methods provide a quantitative measure of a formula's validity [260].

Probabilistic Methods: Probabilistic approaches go a step further, for example in association rule mining probabilities are assigned to the truth values of formulas. These probabilities are derived from the observed data, and they offer a measure of the formula's certainty or confidence. Such methods are particularly useful when dealing with incomplete or noisy data, where absolute determinations are challenging [1, 2, 124, 52].

Beyond confidence-based evaluations, probabilistic methods in logical formula evaluation also include Bayesian inference and Markov models, which provide a framework for incorporating uncertainty in predictive modeling. Bayesian methods, for instance, update the probability estimate for a formula as more evidence becomes available, allowing for dynamic adjustments based on new data [231].

Markov models, on the other hand, assess the likelihood of sequential or temporal transitions between different logical states, making them ideal for analyzing time-series data or scenarios where the system's past state influences its future state. These models can capture complex dependencies and sequences in data, offering a nuanced view of the formula's validity over time [58, 180].

Together, these probabilistic techniques enrich the toolkit for evaluating logical formulas by providing robust mechanisms to handle uncertainty, learn from data, and adapt to new information, thereby enhancing the predictive capabilities and applicability of automated logical reasoning. These approaches are particularly powerful in domains where data continuously evolve, such as in bioinformatics and financial

forecasting.

Possibilistic Methods: Unlike probabilistic methods that deal with likelihood, possibilistic approaches focus on degrees of possibility and necessity [383, 105]. One key aspect of possibilistic methods is their ability to handle uncertainty in a different way compared to probabilistic approaches [270]. They are particularly relevant when the available data is imprecise or vague. While not as widely used as probabilistic techniques, possibilistic methods provide a valuable alternative perspective in situations where the traditional probabilistic models may not be as effective or applicable [270, 10, 63].

2.4.3 Probability-Based vs Possibility-Based Evaluation

The evaluation of logical formulas, especially in uncertain and complex domains, often involves managing uncertainty. Two primary approaches to deal with this uncertainty are probability-based and possibility-based methods.

Probability-Based Evaluation

Probability theory is a well-established mathematical framework for representing and reasoning with uncertain information. In the context of formula evaluation:

- Probabilistic methods assign a probability value to the truth of a formula based on prior information, evidence, or observed patterns [140].
- These methods often rely on Bayesian inference, likelihood ratios, and other probabilistic techniques to update beliefs about the truth of a formula when new evidence is available [308].
- The major advantage of probabilistic methods is their rigorous mathematical

foundation, which allows for precise computations and inferences. However, they require a well-defined probability distribution, which might not always be available or easy to determine [155, 270, 348].

Possibility-Based Evaluation

Unlike probability theory, which quantifies uncertainty by assigning likelihoods to events, possibility theory provides a framework for dealing with different forms of uncertainty based on the concepts of possibility and necessity. Also, contrary to probability theory, which quantifies the likelihood of events within a framework assuming the total probability of all possible outcomes is one, possibility theory focuses on the plausibility of events given available information. This distinction makes it suitable for the evaluation of logical formulas, where the truth of a statement may not always be determined with probabilistic certainty due to incomplete or imprecise information [105].

Mathematical Formulation Possibility and necessity are dual measures defined on a universe of discourse X. For any event $A \subseteq X$, the possibility Π and necessity N measures are defined as follows:

- **Possibility Measure:** The possibility measure is defined by:

$$\Pi(A) = \sup_{x \in A} \pi(x) \tag{2.1}$$

  where $\pi : X \to [0, 1]$ is a possibility distribution function. $\Pi(A)$ quantifies the extent to which the data available does not contradict the occurrence of A. It ranges from 0 (completely impossible, given the data) to 1 (totally possible or certain given the data).

- **Necessity Measure:** The necessity measure is given by:

$$N(A) = 1 - \Pi(\overline{A}) \tag{2.2}$$

where $\overline{A}$ is the complement of A. $N(A)$ quantifies the extent to which the data available confirms the occurrence of A. It ranges from 0 (no confirmation) to 1 (fully confirmed).

Interpretation and Application In the context of evaluating logical formulas, possibility measures can be particularly useful in situations where data about the phenomena being modeled is incomplete or imprecise. For instance, in knowledge representation systems where some relationships may not be fully understood or are subject to change, possibility theory allows for a flexible and robust evaluation mechanism [349, 348].

Example: Consider a scenario in a semantic Web application where we are uncertain if a particular resource should be classified under a specific category due to vague definitions. Possibility theory can help evaluate the plausibility of different classifications without requiring precise probability estimates for each classification.

Both probability-based and possibility-based methods are tailored to specific types of uncertainty and data conditions. Choosing between them depends on the nature of the data, the type of uncertainty involved, and the requirements of the task at hand. For instance, possibility-based evaluation is particularly advantageous in situations requiring a flexible approach to imprecise information, whereas probability-based methods might be preferred where events and their likelihoods are well-defined and quantifiable.

2.4.4 Tableaux Methods

Tableaux methods are a significant and influential approach in automated reasoning, especially in the field of logic and formula evaluation. Originating from proof theory, these methods have been adapted and applied to a variety of logical systems, including propositional, first-order, and description logics [28].

Fundamentals of Tableaux Methods: At its core, a tableau method is a form of decision procedure used for determining the satisfiability of logical formulas. It involves the construction of a tree-like structure called a tableau, where each branch represents a possible world or interpretation under which the formula might be true. The process systematically breaks down complex formulas into simpler components, making it easier to assess their logical validity [95, 125].

Application in Automated Reasoning: In automated reasoning, tableaux methods are particularly valued for their intuitive approach to formula evaluation. They are often used in theorem proving and model checking, where they help in verifying the correctness of logical statements or programs. Their step-by-step decomposition makes them a practical tool for understanding the logical structure of complex formulas [62, 305].

Advantages and Limitations: One of the key advantages of tableaux methods is their ability to provide counter-examples in cases where a formula is not satisfiable. This feature is particularly useful for debugging logical errors in formal specifications [90]. However, these methods can be computationally intensive, especially for formulas with a high degree of complexity or those in expressive logical systems. This limitation can impact their scalability and efficiency in large-scale applications [20].

Comparison with Other Techniques: When compared to other formula evaluation techniques, tableaux methods stand out for their rigorous and detailed analysis

of logical constructs [96]. Unlike some probabilistic or data-driven approaches, they do not rely on statistical models or external data. Instead, they provide a purely logical evaluation based on the intrinsic properties of the formulas themselves. However, this strength also contributes to their computational intensity, contrasting with more scalable methods [28, 96].

2.4.5 Logical Reasoners

Logical reasoners are software tools that assist in processing and inferencing tasks over logical representations such as ontologies. These tools play a crucial role in the evaluation, validation, and discovery of logical formulas, especially in the context of knowledge representation systems like OWL ontologies [27, 139].

Reasoners are capable of tackling a range of tasks, following are some of them:

- **Consistency Checking:** One of the primary functions of a logical reasoner is to check the consistency of a set of axioms or an ontology. An ontology is considered consistent if there are no contradictions within its axioms. Consistency checking ensures the reliability and soundness of the represented knowledge [328, 381].

- **Inference:** Logical reasoners can derive new knowledge by inferring implicit relationships or facts from the given explicit knowledge. This is especially valuable in knowledge bases where direct representation of every fact or relation is infeasible [68, 192].

- **Classification:** Reasoners can classify or categorize entities based on their properties and relationships. In the context of ontologies, this involves organizing classes in a hierarchy based on their defined and inferred subclass relationships [178, 324, 216, 373].

- **Realization:** Logical reasoners can determine the most specific classes that an individual belongs to, given its properties and the ontology's axioms [279, 363, 333].

Optimization: Over the years, optimizing the performance of logical reasoners has been a significant research area. Given the complexity of some reasoning tasks, especially with large and intricate ontologies, efficient algorithms and heuristics are crucial [373, 171, 59, 14].

Popular logical reasoners include tools like HermiT, Pellet, and FaCT++. These tools are widely used in the Semantic Web community and have been integral in various applications ranging from ontology development to data integration [138, 334, 355].

The choice of a particular reasoner often depends on the specific requirements of the task, the complexity of the ontology, and the desired reasoning capabilities. While logical reasoners provide robust and reliable evaluation mechanisms, they are computationally intensive, especially for large-scale or complex knowledge bases. This underlines the need for alternative or complementary evaluation methods, especially those that can leverage machine learning techniques for scalability and efficiency [333]. A comparison of different formula evaluation approaches is shown in Table 2.2.

Approach	**Strengths**	**Weaknesses**
Manual Expert Evaluation	• High accuracy and precision • Incorporates domain expertise	• Time-consuming • Not scalable • Subjective
Statistical Methods	• Scalable • Empirical • Can be automated	• Requires large datasets • Might lack depth in understanding
Probabilistic Methods	• Scalable • Empirical • Can handle uncertainty	• Requires probabilistic models • Can be computationally intensive
Possibilitic Methods	• Empirical • Can handle uncertainty and vagueness	• Requires specific models (e.g., fuzzy logic) • Can be complex
Logical Reasoners	• Reliable and robust • Comprehensive • Widely accepted in the community	• Computationally intensive • Can be slow for large/complex ontologies

Table 2.2: Comparison of Different Formula Evaluation Approaches.

2.5 General Machine Learning Techniques

In this section, we delve into the realm of supervised machine learning, exploring its various methods and how they contribute to the field of logical formula evaluation. Machine learning, a pivotal aspect of modern computational research, offers a diverse toolkit for understanding and processing complex data structures [188, 219].

We begin by discussing the foundational elements of machine learning: regression and classification. These form the bedrock of many machine learning approaches, providing a framework for making sense of data and drawing predictions [189, 190].

Our journey through machine learning methods then leads us to explore algorithms like K-Nearest Neighbors (KNN), Random Forests, and Support Vector Machines. Each of these techniques brings unique strengths and applications, suitable for different types of data and specific challenges [220, 148, 187, 223].

Equally important in machine learning is the process of feature ranking and selection, a crucial step in fine-tuning the model's focus on the most informative aspects of the data. This practice enhances model efficiency and accuracy [197, 201].

Further, we touch on the concept of active learning. This strategy is particularly valuable in scenarios where data is limited or labeling is expensive, as it allows models to actively query for new data, enhancing their learning efficiency [318].

By exploring these general machine learning techniques, we establish a foundation for understanding their application and effectiveness in the evaluation of logical formulas, a key focus of this thesis.

2.5.1 Regression and Classification

Regression and classification, as depicted in Figure 2.3, form the bedrock of supervised learning techniques in machine learning, each addressing different types of

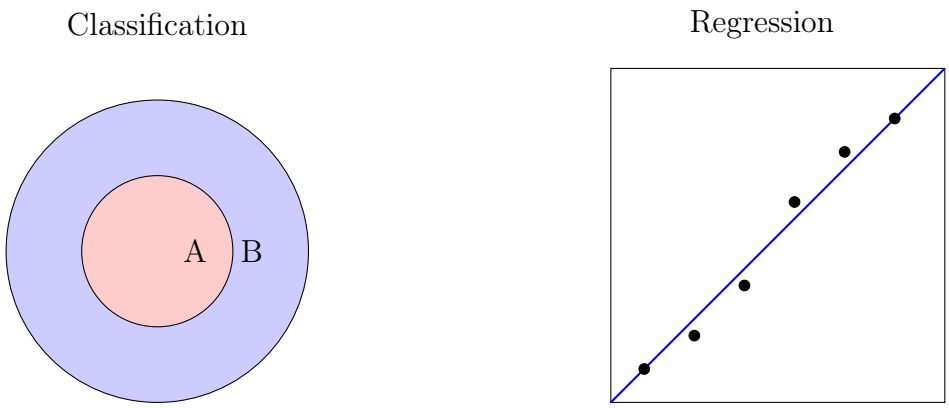


Figure 2.3: Regression and Classification.

prediction problems [189, 190]. Regression is used for predicting a continuous outcome, such as estimating the numerical score of a logical formula based on certain features. It models the relationship between a dependent variable and one or more independent variables, using techniques like linear regression or more complex methods like decision trees and neural networks [151, 218, 149].

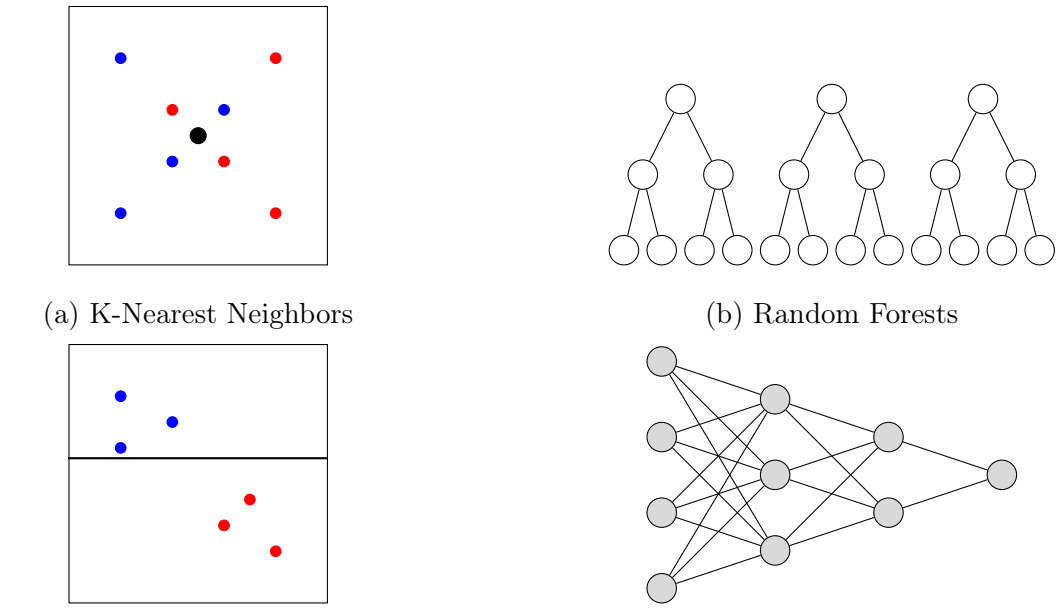
Classification, in contrast, is about assigning labels to instances. For example, in the context of logical formula evaluation, classification models could categorize formulas as valid or invalid, or even classify them into different types based on their structure or semantic properties. Common classification algorithms include logistic regression, support vector machines, decision trees, and neural networks [222, 223, 218, 149].

Both regression and classification have found wide applications in various fields, including finance, healthcare, and social sciences. In the realm of logical formula evaluation, these techniques can be particularly powerful. They can help automate the process of assessing the validity, relevance, or quality of logical formulas, thereby aiding in tasks like theorem proving, knowledge base completion, and semantic reasoning [360, 11, 237].

While regression and classification offer powerful tools for prediction, they are not without challenges. Selecting the right model, tuning hyperparameters, and avoiding overfitting are crucial steps in ensuring the effectiveness of these techniques. Moreover, the quality and quantity of the training data significantly influence the performance of these models, necessitating careful data preparation and preprocessing [150, 198].

2.5.2 Machine Learning Algorithms

Machine learning algorithms have revolutionized the way we approach complex computational tasks. In the context of evaluating logical formulas, these algorithms can be leveraged to gain insights and make predictions that would otherwise be impractical or impossible with traditional methods. Here, we delve into some of the widely used machine learning algorithms, such as K-Nearest Neighbors (KNN), Random Forests, Support Vector Machines (SVM), and Neural Networks, and explore their applications and nuances in the domain of logical formula evaluation. Figure 2.4 provides a simple visualization of these methods. While Table 2.3 at the end of this subsection provides a comparison of these methods.



(a) K-Nearest Neighbors

(b) Random Forests

(c) Support Vector Machines

(d) Neural Network

Figure 2.4: Overview of Machine Learning Algorithms

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN), shown in Figure 2.4a, is a simple yet effective algorithm widely used for classification and regression tasks [220]. It operates on the principle of feature similarity, where the prediction for a new instance is made based on the majority vote or average of the 'K' closest training examples in the feature space. KNN is particularly useful in scenarios where the decision boundary is irregular, as it makes no assumptions about the underlying data distribution [356, 359].

In evaluating logical formulas, KNN can be employed to categorize formulas based on their similarity to known valid or invalid formulas. This approach could be especially useful in systems where new samples are continually introduced, and a rapid, approximate assessment is required [359, 277].

Random Forests

Random Forests are an ensemble learning method, primarily used for classification and regression [148]. They operate by constructing multiple decision trees, as shown in Figure 2.4b, during training and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forests are known for their robustness against overfitting, their ability to handle large datasets with numerous input variables, as well as being potentially explainable [7].

In the realm of logical formula evaluation, Random Forests can be particularly effective in handling complex and high-dimensional data, providing a more nuanced understanding of the relationships between different formula components [330, 371, 316].

Support Vector Machines (SVM)

Support Vector Machines (SVM) are powerful supervised learning models used for classification and regression tasks [223]. They are particularly known for their ability to create optimal hyperplanes in a multidimensional space, which act as a decision boundary between different classes. A simple demonstration of this is provided in Figure 2.4c. SVMs are effective in high-dimensional spaces and are versatile in the sense that they can be customized with different kernel functions to suit various types of data [202].

In the evaluation of logical formulas, SVMs are a viable option due to their exceptional handling of high-dimensional spaces and strong generalization capabilities, which are essential for complex logical reasoning. Their kernel functions enable effective management of non-linear relationships, making them highly efficient and accurate in processing intricate logical structures, even with limited training

data [377, 376].

Neural Networks

Neural Networks, inspired by the structure and function of the human brain, are a set of algorithms designed to recognize patterns and interpret sensory data through machine perception, labeling, and clustering [149]. They are particularly adept at handling non-linear and complex relationships between inputs and outputs. Figure 2.4d shows a typical structure of a fully connected neural network.

In the context of logical formula evaluation, Neural Networks can be trained to recognize patterns and structures in formulas, potentially offering insights that go beyond traditional evaluation methods. Their ability to adapt and learn from new data makes them particularly suited for dynamic environments where formulas and their interpretations may evolve over time [275, 287, 299].

Algorithm	**Strengths**	**Weaknesses**	**Applications**
K-Nearest Neighbors (KNN)	• Easy to implement • Interpretable • Non-parametric	• Slow on large datasets • Sensitive to feature scaling • Saves the entire dataset	• Classification • Regression
Random Forests	• Robust to overfitting • Handles high dimensionality • Good performance on many problems • Might be interpreted	• Complex model • Can be slow to train	• Classification • Regression
Support Vector Machines (SVM)	• Effective in high-dimensional spaces • Versatile	• Requires careful tuning • Not suitable for large datasets	• Classification • Regression
Neural Networks	• Models nonlinear and complex relationships • Good for extremely large datasets • Adaptable	• Requires a lot of data • Can be opaque (black box)	• Classification • Regression • Pattern Recognition

Table 2.3: Comparison of Machine Learning Algorithms

2.5.3 *Feature Ranking and Selection*

Feature ranking and selection play a crucial role in machine learning, especially when dealing with high-dimensional data. These processes are distinct from, yet sometimes intersect with, dimensionality reduction techniques. Dimensionality reduction typically involves transforming or combining features to reduce the total number of variables. In contrast, feature ranking and selection aim to identify and preserve the most informative features, without altering the original features themselves [201, 152, 200].

The primary goal of feature ranking and selection is to enhance a model's performance by eliminating irrelevant or redundant features, thereby improving accuracy, reducing overfitting, and decreasing computational costs. These processes can also enhance model interpretability by simplifying the model structure [197].

Feature selection methods are broadly classified into three categories: filter, wrapper, and embedded methods. Each category employs a different strategy for feature evaluation and selection [197]. Figure 2.5 provides an overview of feature ranking and selection methods.

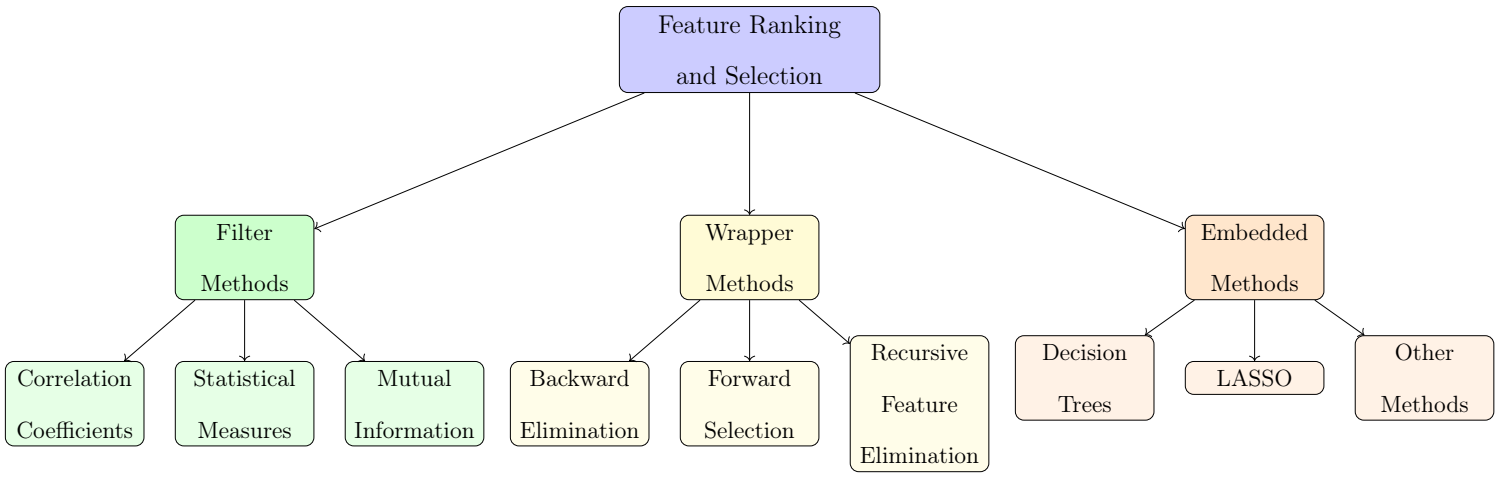


Figure 2.5: Overview of Feature Ranking and Selection Methods in Machine Learning.

Filter Methods use statistical measures to assess the importance of features and are generally independent of any machine learning algorithm [66]. Common techniques under filter methods include:

- *Correlation Coefficients*: Measure the linear relationship between two variables. Features with high correlation to the target variable but low inter-correlation are preferred [41].
- *Mutual Information Gain*: Evaluates the reduction in uncertainty for one variable given a known value of another variable. It is particularly useful in capturing nonlinear relationships [41].
- *ANOVA (Analysis of Variance)*: Assesses the difference in means among groups, helping in identifying features that discriminate effectively between different classes [273].
- *Chi-Square Test*: Used for categorical features, this test assesses the independence of two distributions, helping to identify features that have a strong association with the target variable [245].

Wrapper Methods involve using a specific machine learning model to assess the effectiveness of subsets of features. These methods search through the feature space to find the subset that yields the best model performance, often utilizing techniques like forward selection, backward elimination, and recursive feature elimination [66].

Embedded Methods integrate feature selection as a part of the model training process. Methods like LASSO and decision tree algorithms inherently perform feature selection during model fitting, identifying relevant features based on criteria specific to each algorithm [66].

Careful application of feature ranking and selection methods not only improves model accuracy but also contributes to model interpretability and reduces computational costs. By eliminating redundant and irrelevant features, they make models more efficient and scalable as well as easier to understand [201].

2.5.4 Active Learning

Active Learning is a specialized technique in machine learning that strategically selects the most informative data points for training. It is particularly beneficial in scenarios where labeled data is scarce or expensive to obtain. By focusing on the most informative samples, Active Learning aims to maximize model performance with minimal training data [318]. Figure 2.6 gives an overview of of the Active Learning process showing its flow and components.

The process of Active Learning involves iteratively selecting samples from an unlabeled dataset, querying an oracle (typically a human expert) for labels, and subse-

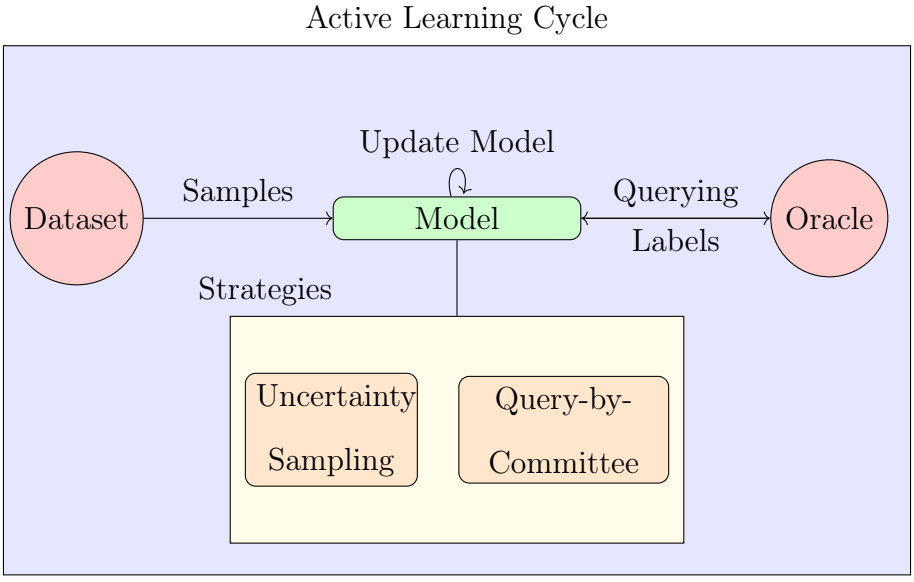


Figure 2.6: Overview of the Active Learning Process

quently updating the model with this new information. This iterative cycle is repeated until a desired level of performance is achieved or resources are exhausted [318].

There are several strategies for selecting the most informative samples in Active Learning, including:

- *Uncertainty Sampling*: This strategy selects samples for which the model has the lowest confidence in its predictions. The intuition is that learning from uncertain examples will provide the most significant knowledge gain for the model [258].

- *Query-by-Committee*: In this approach, multiple models (the committee) are trained on the same data, and samples are selected based on the disagreement among the committee members. The rationale is that disagreement indicates uncertainty or a lack of consensus, thus highlighting informative samples [320].

- *Expected Model Change*: This strategy chooses samples that, when labeled, are expected to bring the most significant change to the current model, indicating a high potential for learning [55].

- *Expected Error Reduction*: Here, the focus is on selecting samples that are expected to most reduce the overall error of the model on the unlabeled data [302, 259].

- *Density-Weighted Methods*: These methods consider not only the model's uncertainty but also the density of the samples in the feature space, aiming to select representative samples from dense regions [345, 392, 391].

Active Learning has been successfully applied in various domains such as text classification, image recognition, and medical diagnosis, where labeled data can be

particularly costly to obtain. To fully harness the potential of Active Learning, several key aspects need to be carefully managed. Firstly, it necessitates access to a reliable oracle for accurate labeling. Secondly, given its iterative process, it can be computationally demanding, requiring efficient resource management. Lastly, to avoid bias and ensure the robustness of the learning model, the initial training set must be thoughtfully curated to represent the overall data distribution comprehensively [233, 262, 352].

In conclusion, Active Learning stands as a powerful tool in the machine learning arsenal, capable of effectively addressing challenges associated with limited labeled data. Its strategic data selection process can lead to significant improvements in model performance, making it a valuable technique for data-efficient learning.

2.6 Vector Space Approaches

Vector space approaches represent a fundamental technique in the field of machine learning and data analysis, particularly when handling high-dimensional data. These approaches, encompassing both embeddings and a variety of similarity measures, provide a means to convert complex data into a format that is more accessible and interpretable for various algorithms. In the context of logical formula evaluation, these vector space methods are instrumental in presenting and comparing the attributes of formulas within a multidimensional framework.

Embeddings and similarity measures, the two main pillars of vector space approaches, play distinct yet complementary roles. Embeddings translate discrete entities, such as words or logical formulas, into continuous vector spaces, maintaining their semantic relationships and contextual nuances. Similarity measures, divided into semantic and non-semantic categories, offer a way to gauge the closeness or distance between data points in these vector spaces. Semantic measures focus on the

meaning and context, crucial for understanding the conceptual similarities between logical formulas. Non-semantic measures, conversely, emphasize the structural or syntactical aspects of the data.

Similarity matrices represent vector spaces where each matrix element characterizes the degree of similarity between pairs of vectors. Each vector corresponds to a data point, and the value of the similarity measure (like cosine similarity or Euclidean distance) between any two data points is represented by the inner product of their corresponding vectors. In this space, the geometric relationships between vectors can be explored to understand the similarities and differences among the data points, making it a powerful tool for clustering, nearest neighbor search, and other machine learning applications.

This section will explore the vital role of vector space approaches in comprehending, processing, and utilizing complex data and structures across various domains. Their application is crucial across domains where nuanced comprehension and advanced processing are key, enhancing our understanding beyond surface-level analysis.

2.6.1 Embeddings

Embeddings are a powerful tool in machine learning, providing a way to represent high-dimensional data, such as logical formulas, in a lower-dimensional vector space [286]. This representation facilitates capturing the underlying semantic and syntactic relationships in a more manageable and computationally efficient format. In the context of logical formula evaluation, embeddings play a crucial role in transforming complex logical structures into vectors that retain their intrinsic characteristics [164, 199, 248].

One common application of embeddings is seen in natural language processing

(NLP), where words or phrases are converted into vectors. This concept can be extended to logical formulas, where each formula is represented as a point in a multi-dimensional space. The proximity of these points reflects the degree of similarity or dissimilarity in their semantic content or structural form. Techniques such as Word2Vec, GloVe, and BERT have demonstrated the effectiveness of embeddings in NLP, and similar methods can be adapted for logical formulas [249, 282, 97].

The process of generating embeddings for logical formulas involves analyzing the structure and content of the formulas, often using neural networks or other advanced algorithms. These methods learn to capture the essential aspects of the formulas, encoding them into a vector space where machine learning models can easily process them. The choice of the embedding technique and the dimensions of the vector space are critical decisions that can significantly impact the performance of subsequent tasks, such as formula classification or similarity assessment [286].

Furthermore, embeddings can be enriched by incorporating domain knowledge or contextual information, enhancing their ability to represent complex logical relationships. This enriched representation is particularly beneficial in tasks requiring a deep understanding of the logical formulas, such as automated reasoning or knowledge extraction [86, 199, 248].

Overall, embeddings offer a transformative approach for handling logical formulas, bridging the gap between the abstract nature of logic and the practical requirements of computational models. They enable a more intuitive and efficient way to manipulate and analyze logical structures, making them an indispensable tool in the field of formula evaluation.

2.6.2 Similarity Measures

In the domain of vector space approaches, similarity measures assume an important role in quantifying the degree of resemblance between entities represented as vectors. These measures are fundamental in tasks involving comparison, clustering, and classification, especially when dealing with complex data like logical formulas. In the context of logical formula evaluation, similarity measures can be utilized to compare the semantic and structural aspects of formulas, aiding in tasks such as formula clustering, retrieval, and even in the evaluation process itself [38, 156, 199].

The essence of a similarity measure lies in its ability to encapsulate the likeness between two data points in a meaningful and quantifiable manner. This becomes particularly challenging with logical formulas, as their comparison involves not only the syntactic alignment but also the semantic congruence. As a result, the choice of similarity measure is crucial and depends on the specific requirements of the task at hand [12]. There are broadly two categories of similarity measures applicable in this context:

1. **Non-Semantic Measures**: These measures quantify the likeness of entities based on their structural or statistical properties without considering their meaning [38].

2. **Semantic Measures**: These measures evaluate the relatedness of concepts by understanding and interpreting their underlying meanings and contexts [156].

The following subsubsections will explore these categories in more detail, highlighting their applications, strengths, and limitations in the context of logical formula evaluation and other related tasks.

Non-Semantic Measures

Non-semantic similarity measures quantify the likeness between entities based on structural, statistical, or geometric properties without considering the meaning or context of the data. They are essential for tasks that require objective comparisons, such as error detection, information retrieval, and data clustering, leveraging mathematical computations like distances or angles [38].

Some commonly used non-semantic similarity measures include:

- **Hamming Distance**: Quantitative, comparing two strings of equal length by counting the number of positions where the corresponding symbols differ.
- **Levenshtein Distance**: Calculates the minimum number of single-character edits needed to change one string into another.
- **String Matching Algorithm**: Compares two strings to assess their similarity or dissimilarity based on character-by-character comparison.
- **Cosine Similarity**: Measures the cosine of the angle between two vectors in a vector space, applicable to any vectors, including those representing text.
- **Jaccard Index**: Compares the similarity and diversity of sample sets, calculating the ratio of the intersection to the union of the sets.
- **Euclidean Distance**: Calculates the straight-line distance between two points (or vectors) in a multi-dimensional space.

While non-semantic measures can be highly efficient and straightforward in their application, they often overlook the deeper, contextual meanings that might be inherent in logical formulas, which can be critical in certain domains or applications.

Interestingly, non-semantic measures like Cosine or Jaccard similarity are often incorporated in the creation of semantic similarity measures when applied to vectors. This demonstrates the utility of non-semantic measures in enriching the process of evaluating semantic similarity by providing a quantitative basis for the comparison, which is then contextualized within the semantic framework. Embeddings, for example, can be used as a tool to create such vectors for pairwise comparison between logical formulas. However, this embedding process is distinct from using the embedded vectors to directly create the vector space. Instead, it serves as a step towards creating a similarity matrix (vector space) between logical formulas, where the similarity measures can be applied to assess semantic relatedness. Figure 2.7 gives an example of these two processes and matrices.

Semantic Measures

Semantic measures in the context of logical formulas focus on assessing the meaning and interpretative aspects of these formulas. Unlike approaches that only consider structural or syntactic characteristics, semantic measures aim to understand the inherent relationships and logical implications within formulas. This understanding is crucial when evaluating the similarity or relevance of logical formulas, especially in domains where the semantic content holds more significance than the syntactic form [156].

One of the primary challenges in implementing semantic measures is capturing the essence of logical constructs in a way that accurately reflects their intended meaning. This often involves sophisticated techniques that go beyond surface-level analysis, requiring a deeper understanding of the domain and the logical constructs involved [199].

Applications of semantic measures are diverse and include tasks such as:

- Semantic clustering of formulas based on their underlying meaning.
- Retrieval of logically similar formulas from a large dataset.
- Semantic evaluation and ranking of formulas in tasks such as formula generation or axiom discovery.

Examples of semantic similarity measures include:

- **WordNet-based Similarity Measures [37]:**
  - **Path Similarity** – Measures the distance between two synsets (a collection of synonyms) in the hierarchy.
  - **Wu and Palmer Similarity (WUP)** – Based on the depths of two synsets and their least common subsumer (LCS).
  - **Leacock-Chodorow Similarity** – Uses the shortest path between two synsets, normalized by taxonomy depth.
- **Latent Semantic Analysis (LSA) [221]:** Analyzes relationships between documents and terms to produce concepts related to them.
- **Ontology-based Similarity Measures [211, 310]:**
  - **Semantic relatedness using information content (IC):** – Based on the information content of the LCS.
  - **Jiang-Conrath Similarity (JC):** – Considers both the LCS's specificity and the individual concepts' information content.

In summary, semantic measures provide a nuanced and meaningful way to evaluate logical formulas, making them indispensable in applications where understanding of semantic content is paramount.

Formula	A	B	C
$A \wedge B$	1	1	0
$\neg A \wedge C$	1	0	1
$B \vee C$	0	1	1

(a) Formula Embedding Vector Space

	$A \wedge B$	$\neg A \wedge C$	$B \vee C$
$A \wedge B$	1	Cos(1,2)	Cos(1,3)
$\neg A \wedge C$	Cos(1,2)	1	Cos(2,3)
$B \vee C$	Cos(1,3)	Cos(2,3)	1

(b) Formula Based Vector Space Using Cosine Similarity of Vectors From 2.7a

Figure 2.7: Representation of Propositional Logic Formulas and Similarity Matrix

2.6.3 Similarity Matrices

The use of similarity matrices in machine learning offers a nuanced approach to understanding and modeling complex relationships between data points. In the context of logical formula evaluation, they play a pivotal role, especially within vector space-based methods [142, 159, 239].

Definition and General Application:

- *Defining Similarity Matrices*: Similarity matrices, also known as proximity matrices, are structured representations where each element quantifies the similarity or distance between pairs of data points (e.g., logical formulas). These matrices are essential in algorithms where the notion of "distance" or "similarity" between samples is fundamental to the learning process [142].

- *Machine Learning Applications*: In broader machine learning applications, similarity matrices facilitate tasks like clustering, nearest neighbor searches, anomaly detection, complex network analyses, and dimensionality reduction techniques like t-SNE and PCA. They enable the models to capture patterns and relationships not immediately apparent in raw data [109, 99, 159, 278].

Structure and Measures:

- *Symmetry in Training*: For training purposes, similarity matrices exhibit symmetry with rows and columns representing the same formulas. This symmetry allows algorithms to understand intra-sample relationships, crucial for tasks such as clustering or similarity-based classification [239]. The shape of this matrix is a square as shown in Figure 2.8a, where each element M_{ij} represents a measure of the similarity or distance between the i^{th} and j^{th} data points. Depending on the context, this measure can represent either a distance (with smaller values indicating closer or more similar items) or a similarity score (where larger values indicate greater similarity).

- *Asymmetry in Testing and Prediction*: During the testing and prediction phases, the matrices are asymmetric as shown in Figure 2.8b. In these phases, these matrices assist in comparing new formulas against a repository of evaluated formulas, providing insights into their novelty, similarity, or potential score.

- *Similarity Measures*: The choice of similarity or distance measure (e.g., cosine similarity, Jaccard index, Euclidean distance) significantly impacts predictions. This choice depends on the nature of formulas, the desired model performance, and the specific objectives of the evaluation task [38].

Challenges and Considerations:

- *Computational Complexity*: The size of the matrix grows with the number of data points (formulas), posing challenges in terms of computational and storage requirements, especially in large-scale evaluations or instances where computing the similarity is computationally intensive [380, 99, 239].

- *Matrix Density and Sparsity*: The decision between using sparse or dense matrices affects both computational efficiency and the accuracy of outcomes, particularly when dealing with datasets where most data points are dissimilar [107, 379].

- *Scalability and Efficiency*: Efficient handling of similarity matrices, especially in the context of large datasets or complex formulas, remains a significant challenge, requiring sophisticated algorithms and often high-performance computing resources [380, 99].

Conclusion: In conclusion, similarity matrices are a fundamental component in the toolbox of machine learning. Their ability to encapsulate complex relationships and facilitate various learning tasks makes them invaluable, though not without challenges related to scalability, computational demand, and the appropriateness of the similarity measures used.

	Train 1	Train 2	Train 3
Train 1	1	$M_{1,2}$	$M_{1,3}$
Train 2	$M_{1,2}$	1	$M_{2,3}$
Train 3	$M_{1,3}$	$M_{2,3}$	1

(a) Symmetric Matrix in Training

	Train 1	Train 2	Train 3
Test 1	$M_{Test_1,Train_1}$	$M_{Test_1,Train_2}$	$M_{Test_1,Train_3}$
Test 2	$M_{Test_2,Train_1}$	$M_{Test_2,Train_2}$	$M_{Test_2,Train_3}$

(b) Asymmetric Matrix in Testing

Figure 2.8: Similarity Matrices in Different Phases

In the next chapter, we discuss work related to this thesis as well as state of the art approaches in evaluating OWL class axioms.

Chapter 3

RELATED WORK

In recent years, significant progress has been made in the field of OWL class axiom evaluation. This chapter provides a comprehensive overview of these advancements, beginning with a discussion on the performance measures and ontologies utilized in evaluating and benchmarking innovative methods. We then showcase a variety of approaches that have emerged at the forefront of this research area, including the development and refinement of a possibilistic heuristic, the application of grammatical evolution in axiom evaluation, and the examination of sophisticated machine learning models. By presenting these developments and the tools used for their evaluation, this chapter aims to offer a thorough understanding of the current state-of-the-art in OWL class axiom evaluation, setting a foundation for future explorations and enhancements in the field.

3.1 Ontologies and Evaluation Metrics

This section outlines two elements utilized in our research: the ontologies and evaluation metrics. The ontologies discussed herein provide the datasets upon which our evaluation and analysis is built. They range from general knowledge bases to domain-specific structures. Additionally, we detail the metrics employed to assess the performance of the proposed models. These components are needed to understand the effectiveness of our approaches in enhancing semantic web technologies and ontology-based applications.

3.1.1 Ontologies Utilized in This Thesis

In this thesis, a variety of ontologies have been utilized to underpin the research and experiments conducted. These ontologies, selected for their relevance to the specific domains and scenarios under investigation, serve as a fundamental component of our framework. They range from comprehensive, widely-recognized knowledge bases to more specialized, domain-specific ontologies. Each ontology has been chosen based on its way of contruction (manual or automatic), size in terms of classes, and number of axioms which are all pertinent to the research questions posed, demonstrating the ability of our proposed models when handling different scenarios.

The NTNames Ontology

The NTNames (New Testament Names) Ontology represents a significant endeavor in the domain of biblical studies, specifically focusing on the semantic knowledge base of named entities within the New Testament. Comprising approximately 600 names, it categorizes entities such as individuals (men, women), groups of people, locations, and divine figures into a structured, interconnected web of information. This ontology is particularly notable for its educational and research utility in theological studies, offering a detailed and standardized representation of the complex relationships and attributes found within New Testament narratives [48].

Ontology Statistics Table 3.1 below provides a quantitative overview of the NTNames Ontology, reflecting its scope and depth in representing New Testament entities.

Metric	Value
Number of Classes	47
Number of Triples	518
Number of Properties	38
Number of subClassOf axioms	278
Number of disjointWith axioms	10
Number of equivalentClass axioms	50

Table 3.1: NTNames Ontology Statistics.

This ontology delineates a structured approach to understanding the relationships and attributes of individuals and locations mentioned in the New Testament, such as familial ties, residence or origin, and religious beliefs. The ontology's detailed classification and its support for re-use in other contexts mark it as a pivotal resource for both theological education and semantic web research within religious studies.

The Pizza Ontology

The Pizza Ontology, a hallmark in the domain of semantic web education, provides a structured framework for representing pizza-related concepts, toppings, and bases within an ontology. It is primarily used as an educational tool to demonstrate the capabilities of the Protégé ontology editor and OWL (Web Ontology Language), illustrating the creation of class hierarchies, properties, and restrictions. This ontology is pivotal for understanding the intricacies of ontology construction and reasoning [76].

Ontology Statistics Table 3.2 provides an overview of the Pizza Ontology's scale and scope, emphasizing its educational use and the number of classes it covers.

Metric	Value
Number of Classes	101
Number of Individuals	5
Number of Properties	8
Maximum depth	6
Number of subClassOf axioms	651
Number of disjointWith axioms	5
Number of equivalentClass axioms	101

Table 3.2: Pizza Ontology Statistics.

This table reflects the educational nature and design of the Pizza Ontology, highlighting its foundational role in teaching and understanding ontology concepts within the Semantic Web field.

MatOnto

MatOnto is designed to serve the materials science domain, offering a structured vocabulary for representing and linking data across this field. Its development signifies a step towards standardizing materials science data representation, facilitating data sharing, and enabling interoperability between systems. With its comprehensive coverage, MatOnto plays a crucial role in advancing materials science research and development [74].

Ontology Statistics Table 3.3 provides a summary of MatOnto's structure and coverage.

Metric	Value
Number of Classes	848
Number of Individuals	131
Number of Properties	96
Maximum depth	10
Number of subClassOf axioms	853
Number of disjointWith axioms	158
Number of equivalentClass axioms	9

Table 3.3: MatOnto Ontology Statistics.

The diverse array of classes and intricate relationships outlined in the MatOnto statistics reveal its sophisticated structure, designed to meticulously represent the materials science domain. Its detailed classification enable precise data interoperability within this specialized field.

The DBpedia Ontology

The DBpedia ontology, at the core of the DBpedia knowledge base, crafted meticulously to structure the wealth of information available on Wikipedia into a well-organized, semantic web-compatible format. It represents a collaborative effort to convert Wikipedia's infobox data into a structured, universally accessible ontology, encompassing a wide range of domains. This ontology is not only a product of automated extraction processes but also benefits from ongoing community contributions, ensuring its continuous expansion and update to reflect the latest knowledge captured in Wikipedia. It serves as a cross-domain semantic framework that significantly contributes to the field of knowledge representation [92].

Ontology Statistics Table 3.4 provides an overview of the DBpedia ontology's scale and scope, highlighting its extensive coverage and the breadth of concepts it encompasses. The DBpedia version we used is **2015-04**, we do so to maintain consistency in our comparison with other work. The statistics provided include disjoint axioms we added from the results of Nguyen's work [263].

Metric	**Value**
Number of Classes	768
Number of Facts (Triples)	> 850 million
Number of Properties	2,861
Number of subClassOf axioms	4,300
Number of disjointWith axioms	581
Number of equivalentClass axioms	834

Table 3.4: DBpedia Ontology Statistics.

These statistics underscore the comprehensive nature of the DBpedia ontology, demonstrating its wide applicability in semantic web and linked data projects. The ongoing contributions from the community ensure its continuous growth and relevance across various research and development endeavors.

The Gene Ontology

The Gene Ontology (GO) is an extensive framework designed to represent knowledge across the biological domain comprehensively. It is structured around three core aspects: Molecular Function, Cellular Component, and Biological Process. These aspects capture the essence of gene products' activities, the locations within the cell where these activities occur, and the broader processes or programs accomplished by

multiple molecular activities, respectively [21].

The GO is dynamic, aimed at representing the current state of biological understanding, and thus is regularly updated to reflect new scientific discoveries. These updates are collaboratively managed by a team of ontology editors and the broader scientific community, ensuring that GO remains a critical and up-to-date resource for computational analysis in molecular biology and genetics research [6].

Ontology Statistics For a better understanding of the Gene Ontology, Table 3.5 offers a structured overview of its components. While specific numeric details are dynamically updated and accessible through the GO's official platform [1] , this table underscores the ontology's breadth in encapsulating gene functions across diverse biological aspects. The statistics presented are unique for the version used in our work which is **2022-10**.

Metric	**Value**
Number of GO Terms (Classes)	43,329
Number of Annotations (Triples)	7,694,564
Number of Gene Products	1,503,740
Number of Covered Species	5,257

Table 3.5: Gene Ontology Statistics.

GO stands as a cornerstone in bio-informatics, offering expansive insights into gene functions and biological processes. The presented statistics highlight its breadth, with over 42,000 terms, 7.6 million annotations, and data on more than 1.5 million gene products across 5,387 species. This vast dataset underpins the GO's utility in

[1]https://www.geneontology.org/stats.html

facilitating comprehensive cross-species genetic and functional analyses, essential for advancing our understanding of biological systems and disease mechanisms.

The Cell Ontology

The Cell Ontology (CL) is a prominent ontology within the OBO Foundry, dedicated to the classification and description of biological cell types. Focused primarily on animal cell types, CL plays a crucial role in enhancing interoperability among specialized ontologies, facilitating a comprehensive understanding of cell functions and their locations. Its integration with other ontologies, such as the Uberon multi-species anatomy ontology and the Gene Ontology, allows for detailed recording of cell location and function, adhering to FAIR principles for scientific data management [98].

CL's development is community-driven, with active engagement from editors and contributors across multiple projects. This collaborative approach ensures that CL remains relevant and up-to-date, supporting a wide array of applications. Notable projects leveraging CL include the HuBMAP, Human Cell Atlas, Single Cell Expression Atlas, and ENCODE, among others. These applications highlight CL's role in annotating cell types and facilitating cellular reference mapping across diverse research initiatives [129].

Ontology Statistics Table 3.6 offers a glimpse into Cell Ontology's scale, reflecting its utility in addressing the complex requirements of modern biological research and bio-informatics applications.

Metric	Value
Number of Classes	16,163
Number of Individuals	18
Number of Properties	529

Table 3.6: Cell Ontology Statistics.

The Cell Ontology showcases a robust framework with over 16,000 classes, underscoring its depth in categorizing cell types. With a modest count of individuals and a significant number of properties, it reflects a detailed, property-rich structure for describing cell characteristics and relationships. This extensive compilation facilitates comprehensive bio-informatics analyses, supporting diverse research applications from basic science to clinical studies.

The Food Ontology

The Food Ontology (Foodon) is a comprehensive semantic framework designed to represent and integrate knowledge about food across various dimensions, from production to consumption. It leverages the foundational structures of the LanguaL system, a food classification scheme developed by the FDA, enriched with ontology technology to ensure global interoperability and data integration. The ontology encapsulates a wide range of food-related concepts, including raw ingredients, processing methods, and product types, facilitating detailed and harmonized descriptions of food items. It supports a broad array of applications, from enhancing food traceability and quality control to promoting data sharing across international borders [102].

Ontology Statistics Table 3.7 below outlines the structural metrics of the Food Ontology, demonstrating its extensive coverage of the food domain.

Metric	Value
Number of Classes	35,304
Number of Individuals	129
Number of Properties	435

Table 3.7: Food Ontology Statistics.

This ontology offers a rich and dynamic vocabulary that is crucial for addressing the complexities of the global food system. By providing a detailed classification of food products along with their properties and relationships, the Food Ontology serves as a vital resource for researchers, industry stakeholders, and policymakers aiming to advance food science, safety, and nutrition.

Summary

The ontologies discussed in this section, ranging from toy ontologies like the Pizza Ontology to comprehensive ones like the Gene Ontology (GO), are integral for this work. They provide a practical foundation for applying the axiom evaluation methods developed herein. Toy ontologies are utilized for preliminary testing and demonstrations due to their simplicity and manageability, which are ideal for initial model tuning. In contrast, complex ontologies like GO provide a realistic and challenging dataset for evaluating the scalability and robustness of the active learning models developed, which are essential for real-world applications of axiom discovery and validation in semantic web environments.

3.1.2 Evaluation Metrics

In this subsection, we will explore the evaluation metrics used for assessing the performance and effectiveness of the methods developed throughout this PhD thesis. Evaluation metrics are indispensable tools that provide quantitative measures to compare different models, algorithms, or techniques on a common ground [387]. By carefully selecting and applying appropriate metrics, we can objectively determine the strengths and weaknesses of our approaches. Some metrics are used in evaluating different tasks, we give an example of how they are used in each.

Regression Metrics

Regression metrics are crucial for evaluating the performance of regression models, which predict a continuous outcome. These metrics help in understanding how well a model's predictions match the actual data, highlighting the model's accuracy and efficiency [388].

Mean Squared Error (MSE) MSE assesses the average squared difference between the estimated values and the actual value [101].

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) RMSE is the square root of the mean squared error, offering a measure of the average error magnitude [388].

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

R-squared (R^2) R^2, also known as the coefficient of determination, indicates the proportion of the variance in the dependent variable that is predictable from the

independent variable(s) [100].

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

These metrics collectively provide a comprehensive assessment of a regression model's predictive performance, enabling researchers to gauge the model's effectiveness in capturing and explaining the variability in the data.

Binary Classification Metrics

Binary classification metrics evaluate the performance of models that categorize instances into one of two groups. These metrics are essential for assessing the effectiveness and reliability of classification algorithms in distinguishing between the two possible outcomes [388].

Accuracy Accuracy measures the proportion of true results (both true positives and true negatives) among the total number of cases examined [388]. For instance, if a model correctly identifies 90 out of 100 instances, its accuracy is 0.9 or 90%.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP, TN, FP, and FN represent the numbers of true positives, true negatives, false positives, and false negatives, respectively.

Precision Precision is the ratio of correctly predicted positive observations to the total predicted positive observations [388]. If a model identifies 80 positives out of 100 instances, and 60 of these are correct, precision is 0.75.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall Recall (Sensitivity) is the ratio of correctly predicted positive observations to all observations in the actual class [388]. If a model identifies 80 positives out of 100 instances, and 60 of these are correct, if there were 80 actual positives, recall is 0.75.

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1 Score The F1 score is the harmonic mean of precision and recall, providing a balance between the two. It is particularly useful when the cost of false positives and false negatives is high [388]. For precision and recall both equal to 0.75, the F1 score is also 0.75.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Area Under the ROC Curve (AUC) The Area Under the Receiver Operating Characteristic (ROC) Curve, or AUC, measures the ability of a binary classification model to discriminate between positive and negative classes. An AUC of 1 indicates perfect prediction, while an AUC of 0.5 suggests no discriminative power [388].

Confusion Matrix A confusion matrix is a table used to describe the performance of a classification model on a set of test data for which the true values are known [388]. It allows the visualization of the model's predictions, including true positives, true negatives, false positives, and false negatives. Table 3.8 depicts the composition of a simple confusion matrix.

Matthews Correlation Coefficient (MCC) The Matthews correlation coefficient (MCC) is a measure of the quality of binary classifications [75]. It takes into account true and false positives and negatives and is regarded as a balanced measure

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Table 3.8: Sample Confusion Matrix for Binary Classification.

which can be used even if the classes are of very different sizes. The MCC is calculated as:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

These metrics offer a comprehensive view of a binary classification model's performance, highlighting its strengths and areas for improvement.

Hits@K Metric Used for Ontology Completion Tasks

Hits@K assesses whether the relevant document appears in the top K search results. It is a way to evaluate the model's ability to rank relevant documents higher than others [71].

For instance, if the relevant document appears within the top 5 results of a query, Hits@5 for that query is 1 (success); otherwise, it is 0 (failure).

In ontology completion tasks, the Hits@K metric serves as a crucial measure for assessing model performance in predicting the correct class of a head entity from a set of candidates, against the tail entity's class, considered as the ground truth. The setup involves ranking each axiom in the testing set based on the model's predicted score, which reflects the likelihood of a candidate class being the correct class for the head entity [70]. The Hits@K equation, tailored for this context, is represented as:

$$\text{Hits@K} = \frac{1}{|A|} \sum_{a \in A} \mathbb{1}(\text{rank}_a \leq K)$$

where:

- $|A|$ denotes the total number of axioms in the testing set,
- a represents an individual axiom under consideration,
- rank_a is the ranking of the ground truth class (tail entity) among the ranked list of candidate classes for the head entity, based on the model's predicted score,
- $\mathbb{1}$ is the indicator function that returns 1 if the ground truth class is within the top K ranked classes, and 0 otherwise.

This equation encapsulates the model's efficiency in accurately predicting the class of head entities, thereby directly influencing ontology completion's effectiveness by ensuring only the most relevant and likely class predictions are considered in the top K ranks.

Mean Reciprocal Rank (MRR) Mean Reciprocal Rank is a metric used to evaluate the performance of a ranking algorithm. MRR calculates the average of the reciprocal ranks of results for a sample of queries, with the rank being the position of the first relevant entity. The higher the MRR, the better the model's performance [85].

Summary

The evaluation metrics discussed in this section are essential for validating the effectiveness and efficiency of the proposed axiom evaluation methods. For instance,

metrics like F1-score are used for assessing the balance between precision and recall, particularly useful in optimizing the performance of classification models on complex datasets. Meanwhile, computational time and memory usage metrics are indispensable for evaluating the practicality of the machine learning models in handling large-scale ontological data, ensuring that the developed solutions are not only accurate but also scalable and efficient in resource-constrained environments.

3.2 Evolution of Possibilistic Heuristic in Ontology Enrichment

3.2.1 Initial Proposition: A Possibilistic Approach

The pioneering work in 2014 by Tettamanzi et al. introduced a groundbreaking scoring heuristic for evaluating candidate axioms, marking a shift from the conventional statistical inference methods to a framework based on possibility theory [349]. This heuristic was characterized by its innovative approach to assigning bipolar scores to candidate axioms, encapsulating both a degree of possibility (Π) and a degree of necessity (N).

Heuristic Foundations The heuristic's foundation lays in treating each formula that logically follows from an axiom as both a potential falsifier and a confirmation, depending on the RDF data. The absence of counterexamples to a hypothesis in the RDF repository implies that the hypothesis is completely possible ($\Pi(\phi) = 1$), whereas the presence of confirmations and absence of counterexamples increase the necessity of the hypothesis, approaching a score of 1 [349]. The heuristic uniquely interpreted a confirmation as a fact that not only satisfies an axiom but also favors it over its contrary, aligning with Scheffler and Goodman's selective confirmation theory [313].

The Approach Method wise, the paper leveraged the model-theoretic semantics of OWL 2, defining an interpretation domain (Δ_I) as the set of all resources occurring in a given RDF store. This approach facilitated the axiom evaluation by checking if the interpretation is a model of the axiom under consideration. This was particularly relevant in the context of RDF stores, which are often incomplete and noisy, necessitating the adoption of the open-world hypothesis. Thus, the heuristic accounted for the possibility that an axiom might hold true even in the presence of a few counterexamples [349].

Experimental Evaluation The experimental evaluation was conducted using the DBpedia 3.9 dataset as the RDF fact repository. This involved downloading the DBpedia dumps and performing tests of subsumption axioms. The evaluation on a substantial dataset like DBpedia, consisting of over 800 million RDF triples, demonstrated the heuristic's applicability to ontology learning and knowledge-base validation. The heuristic was applied to subClassOf axioms within the DBpedia RDF dataset. The experiment was significant not only for validating the heuristic's effectiveness but also for highlighting its suitability for large-scale ontology evaluation tasks [349].

Future Directions Despite its innovative approach, the heuristic faced critiques about its subjective nature, given its foundation in possibility theory. Nevertheless, the authors recognized this and suggested future work to extend the heuristic to a broader set of axioms and include additional RDF datasets from the Linked Open Data cloud, along with improvements in computational efficiency, particularly by implementing a time-out on query evaluation to reduce the overhead of axiom testing [349].

3.2.2 Intermediate Developments: Dynamically Time-Capped Testing

The 2015 study by Tettamanzi et al. represented an important intermediate step in the evolution of the possibilistic heuristic, introducing dynamically time-capped testing of subClassOf axioms against RDF data to enrich schemas [350]. This development was an important step in addressing the computational challenges associated with the heuristic's earlier iterations.

Axiom Scoring and Possibility Theory Building on the foundation laid in 2014, the 2015 paper further refined the axiom scoring heuristic based on possibility theory. The method centered on assigning a degree of possibility and necessity to a candidate axiom, with a focus on the open-world assumption prevalent in linked data on the Web. The heuristic was designed to overcome some limitations of scoring heuristics based on statistical inference, particularly in handling the open-world assumption [350].

Time-Capped Testing A significant contribution of this paper was the introduction of a method based on time capping to alleviate the computational burden of the heuristic without sacrificing the precision of the scores. This approach was particularly effective in testing subClassOf axioms against large RDF datasets like DBpedia, where computing the possibilistic score could be computationally intensive [350].

Innovations The paper presented several innovations, including a detailed computational framework for axiom scoring and scalable axiom scoring based on time prediction. These innovations were instrumental in making the heuristic more practical for large-scale ontology learning and validation tasks. The approach involved translating OWL 2 axioms into SPARQL queries, enabling the efficient testing of

axioms against RDF data [350].

Experimental Evaluation The experimental evaluation conducted using the DBpedia dataset demonstrated the heuristic's applicability and effectiveness. The dynamically time-capped approach led to a significant reduction in computation time, showcasing the method's practicality in handling large RDF datasets. This evaluation was crucial in validating the heuristic's improvements and setting the stage for further refinements [350].

Future Directions The 2015 paper laid the groundwork for future advancements in the possibilistic heuristic. It highlighted the need for further optimization to handle the growing scale of RDF datasets and the complexity of OWL axioms. The study pointed towards a more scalable and efficient approach to ontology learning and validation, paving the way for the subsequent developments that were to come in the 2017 iteration of the heuristic [350].

3.2.3 Refinement and Extension

In their 2017 paper, Tettamanzi et al. advanced the theory of the possibilistic framework for OWL 2 axiom testing, refining the initial heuristic and extending its application scope [348]. This version emphasized a more systematic and rigorous approach to evaluating the credibility of OWL 2 axioms based on available evidence in RDF datasets.

Conceptual Framework The core of the refinement was the introduction of the notions of development, content, support, confirmation, and counterexample of an axiom. These concepts were instrumental in defining the possibility and necessity of an axiom, as well as its acceptance/rejection index, a combination of the two. This

framework was then applied to test subClassOf axioms against the DBpedia RDF dataset, showcasing the practical application of the refined heuristic [348].

Advancements A key advancement was the operationalization of the model-theoretic semantics of OWL 2 axioms into corresponding first-order logic formulas. This transformation was crucial for querying RDF datasets to test OWL 2 candidate axioms. The process involved translating OWL 2 axioms into first-order logic based on the set-theoretic formulas of the OWL direct semantics, thus providing a robust foundation for the possibilistic framework [348].

Technical Implementation The transformation process involved a recursive definition of a translation function, $t(\cdot; x, y)$, that took OWL 2 entity expressions or axioms as arguments and translated them into first-order logic expressions. This translation was aligned with the direct model-theoretic semantics of OWL 2 and allowed for the systematic development of axioms with respect to an RDF dataset [348]. Examples included translating atomic concepts, relations, and complex expressions like $\exists R.C$ and $\forall R.C$ into first-order logic.

Computational Efficiency Addressing computational efficiency, the paper discussed the implementation of time capping as a means to alleviate the computation load of the possibilistic axiom scoring heuristic. This approach helped maintain the precision of the scores while managing the computational resources more effectively, especially in large-scale applications [348].

Future Directions The paper concluded with insights into future work, emphasizing the need for further refinement of the heuristic, exploring its applicability to a wider range of OWL 2 axioms, and enhancing its computational efficiency. This

direction was critical for making the heuristic more practical for large-scale ontology learning and validation tasks.

3.2.4 Further Advancements: Enhanced Computational Efficiency

The 2022 paper marked a significant step forward in the evolution of the possibilistic heuristic for ontology enrichment. In it, Felin et al. addressed the critical challenge of computational efficiency in assessing OWL subClassOf axioms against RDF data [123].

Optimization of Heuristic Assessment This paper introduced three major contributions to enhance the efficiency of the possibilistic heuristic. Firstly, a multithreading system was implemented to parallelize the evaluation of axioms, significantly reducing the overall computational time. Secondly, an extension of the original heuristic was proposed to avoid redundant computation, addressing the computational bottleneck identified in previous iterations. Thirdly, the optimization of SPARQL query chunking was achieved by leveraging an extension of the SPARQL 1.1 Federated Query standard [123].

Impact on Computational Performance The optimizations brought forth in this paper had a profound impact on computational performance. Experiments conducted on the DBpedia 3.9 dataset, comprising over 463 million triples and 532 OWL classes, demonstrated the efficacy of these improvements. The maximum computation time for axiom assessment was significantly reduced from approximately 71,699 minutes to just 489 minutes. Importantly, these optimizations did not compromise the accuracy of the heuristic, as evidenced by the unchanged Acceptance/Rejection Index (ARI) values for each axiom [123].

Computational Efficiency Analysis A comparative analysis of the computation times using the original heuristic versus the optimized approach showed that the average CPU time for evaluating an axiom was reduced from 578 minutes to 30 minutes. This optimization was effective for approximately 82% of the tested candidate axioms, significantly reducing the computational burden and making the heuristic more practical for large-scale applications [123].

Future Directions Looking forward, the authors proposed to extend these optimizations to other types of OWL axioms and to generalize the optimization to computational problems with SPARQL queries of a similar nature. This direction aims to broaden the applicability of the heuristic and contribute further to the semantic Web community [123].

3.2.5 Strengths and Weaknesses of the Possibilistic Heuristic

The possibilistic heuristic developed for evaluating OWL axioms against RDF data exhibits several notable strengths and weaknesses, as revealed through its evolution from 2014 to 2022.

Strengths

- **Adaptability to the Open-World Assumption:** The heuristic effectively handles the open-world assumption inherent in RDF datasets, a crucial aspect for semantic web applications.
- **Bipolar Scoring System:** The introduction of bipolar scores (possibility and necessity) provides a nuanced way to evaluate axioms, moving beyond simplistic binary evaluations.

- **Computational Efficiency:** Subsequent improvements, particularly in 2022, significantly enhanced the computational efficiency, making the heuristic more viable for large-scale datasets.
- **Scalability:** The heuristic's ability to work with extensive RDF datasets like DBpedia demonstrates its scalability, an essential factor for practical semantic web applications.

Weaknesses

- **Subjectivity in Scoring:** Relying on possibility theory introduces a level of subjectivity, as the scoring is influenced by the presence or absence of evidence rather than statistical probability.
- **Initial Computational Demand:** Earlier versions of the heuristic, particularly the 2014 and 2015 iterations, faced challenges in computational demand, making them less efficient for large datasets.
- **Limited Axiom Scope:** The primary focus on subClassOf axioms means the heuristic might not be directly applicable to other types of axioms, like DisjointWith, without further adaptation.
- **Potential for False Negatives:** The conservative nature of the heuristic, especially in its earlier versions, could lead to false negatives, where valid axioms might be rejected under certain conditions.

These strengths and weaknesses highlight the heuristic's potential and the areas where further research and development could be beneficial. The evolution of the heuristic over the years has addressed several of its initial shortcomings, particu-

larly in terms of computational efficiency, making it a more robust tool for ontology enrichment and semantic web applications.

3.2.6 Conclusion

The progression of the possibilistic heuristic, as described in this section, forms a solid foundation for this thesis by offering a sophisticated framework for understanding and enhancing ontology enrichment. This heuristic serves dual purposes in this research: firstly, as a benchmark against which the efficiency of our proposed methods is measured, and secondly, as the base scorer whose outputs are treated as the ground truth during evaluations. A principal aim of our research is to develop a model that replicates the heuristic's scoring function with greater computational efficiency. Achieving this would allow for the integration of our evaluation model into larger ontology enrichment systems, thereby enhancing their performance and scalability.

3.3 RDFMiner: An Evolutionary Approach for Axiom Mining in the Semantic Web

3.3.1 Initiating RDFMiner: Grammatical Evolution for Class Disjointness Axioms

The pioneering work by Nguyen and Tettamanzi present RDFMiner [264, 265], a tool developed to enrich ontologies by automatically discovering OWL class disjointness axioms from RDF data. This work represents a significant advancement in ontology learning within the Semantic Web.

Innovative Method RDFMiner employs Grammatical Evolution (GE), an evolutionary algorithm, to evolve complex class disjointness axioms. This approach allows for iterative refinement of axioms, enhancing their quality and relevance.

Theoretical Foundations The papers underscore the necessity for automatic axiom generation in the Semantic Web, emphasizing the role of RDF and OWL in ontology construction and the importance of axioms in ensuring data consistency and enabling effective data reasoning.

Implementation and Evaluation RDFMiner's implementation includes a Backus-Naur Form (BNF) grammar for axiom generation and a genotype-to-phenotype mapping process. The paper demonstrates RDFMiner's precision and coverage in comparison to a Gold Standard, validating its efficacy.

Possibilistic Axiom Scoring In RDFMiner, the scoring of axioms utilizes the possibilistic heuristic detailed in Sec3.2, and expands upon the heuristic to deal with disjointness axioms. The principle is that an axiom ϕ is considered completely possible (i.e., its possibility measure $\Pi(\phi)$ equals 1) if there are no counterexamples to it in the RDF repository. The content of an axiom ϕ, defined as a finite set of logical consequences, is given by:

$$\text{content}(\phi) = \{\psi : \phi \vDash \psi\} \tag{3.1}$$

The support of ϕ, denoted as $\text{support}(\phi)$, is the cardinality of $\text{content}(\phi)$. The number of confirmations (statements satisfied by the RDF repository) and counterexamples (statements falsified by the RDF repository) are also determined. The possibilistic measures, possibility $\Pi(\phi)$ and necessity $N(\phi)$, of an axiom are defined as:

$$\Pi(\phi) = 1 - \sqrt{1 - \left(\frac{\text{counterexamples}(\phi)}{\text{support}(\phi)}\right)^2} \tag{3.2}$$

$$N(\phi) = \begin{cases} \sqrt{1 - \left(\frac{\text{confirmations}(\phi)}{\text{support}(\phi)}\right)^2} & \text{if } \Pi(\phi) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

The fitness of an axiom, which is a measure of its quality, is directly proportional to its necessity and possibility. However, the fitness function used in RDFMiner focuses on counterexamples only, as RDF datasets naturally provide counterexamples for disjointness axioms. The generality of an axiom, a measure of how broadly applicable it is, is defined as the cardinality of its support. The refined definition of the fitness function is then given by:

$$f(\phi) = \text{support}(\phi) \cdot \Pi(\phi) \tag{3.4}$$

This fitness function, combining the generality of an axiom with its possibility measure, is central to the evaluation process in RDFMiner, ensuring that the axioms generated are both credible and broadly applicable.

BNF Grammar for Axiom Generation The papers outline the development of a Backus-Naur Form (BNF) grammar crucial for RDFMiner's axiom generation process. This BNF grammar is divided into two main parts: static and dynamic.

Static Part The static part of the BNF grammar consists of production rules defining the structure of axioms. These rules are loaded from a hand-crafted text file and are fundamental in determining the kinds of axioms generated. The static grammar is responsible for generating the syntactic structure of class disjointness axioms.

Dynamic Part The dynamic part involves production rules for the primitives, constructed automatically at runtime by querying the SPARQL endpoint of the RDF

repository. This dynamic nature ensures that the grammar adapts to changes in the RDF dataset without requiring manual updates.

An example of the grammar structure for generating class disjointness axioms is as follows:

(r1) Axiom := ClassAxiom

(r2) ClassAxiom := DisjointClasses

(r3) DisjointClasses := 'DisjointClasses' '(' ClassExpression ' ' ClassExpression ')'

(r4) ClassExpression := Class | ObjectUnionOf | ObjectIntersectionOf

(r5) ObjectUnionOf := 'ObjectUnionOf' '(' ClassExpression ' ' ClassExpression ')'

(r6) ObjectIntersectionOf := 'ObjectIntersectionOf' '(' ClassExpression ' ' ClassExpression ')'

(r7) Class := [SPARQL query generated classes]

This BNF grammar facilitates the transformation of genotypes (integer strings) into phenotypes (class disjointness axioms) through a mapping process. The production rule for the primitive class is dynamically filled using SPARQL queries to extract classes from the RDF dataset, ensuring that the grammar reflects the current dataset's contents.

Conclusion RDFMiner, with its GE-based approach, marks a significant milestone in automating ontology enrichment, setting the stage for further advancements in the automated generation of complex axioms in the Semantic Web.

3.3.2 Multi-Objective Evolutionary Approach in RDFMiner

In their work "A Multi-Objective Evolutionary Approach to Class Disjointness Axiom Discovery," Nguyen and Tettamanzi introduce a significant advancement in RDFMiner by incorporating a Multi-Objective Genetic Evolution (MOGE) approach

[267].

Introduction of Multi-Objective Optimization This paper marks a shift in RDFMiner's development by introducing a multi-objective approach to the evaluation of candidate axioms. The focus is on refining the evaluation framework to simultaneously optimize two independent criteria: the credibility and generality of axioms. Additionally, the paper introduces a novel measure, termed "similarity," enhancing the diversity of the axioms obtained. This approach ensures higher accuracy and generality in the axioms mined from RDF datasets.

New Objective Functions The paper proposes new objective functions in the multi-objective framework to evaluate the fitness of axioms. The fitness of an axiom is determined based on three key measures: possibility, generality, and similarity. The possibility and generality measures were introduced in Section 3.2, while the similarity measure is a new addition, quantifying the uniqueness of an axiom within the population. The objective functions used are:

$$\text{Maximize } f_1 = \Pi(\phi) \cdot \sqrt{1 - s(\phi)^2} \tag{3.5}$$

$$\text{Maximize } f_2 = g_\phi \cdot \sqrt{1 - s(\phi)^2} \tag{3.6}$$

where $\Pi(\phi)$ is the possibility measure, g_ϕ is the generality measure, and $s(\phi)$ is the similarity measure of the axiom ϕ.

Similarity Measure The similarity measure $s(\phi)$ is introduced to quantify the uniqueness of an axiom within the population. It is calculated as the average similarity between the axiom ϕ and each axiom a_i in the population, excluding ϕ itself. This is expressed mathematically as:

$$s(\phi) = \frac{1}{n-1} \sum_{i=1, a_i \neq \phi}^{n} s(\phi, a_i) \tag{3.7}$$

This measure ensures that the axioms generated are not only credible and general but also diverse, enhancing the overall quality of the knowledge discovery process in RDFMiner.

Implications of MOGE in RDFMiner Integrating MOGE into RDFMiner allows for a more nuanced approach to axiom discovery, balancing multiple objectives to produce a diverse and robust set of axioms. This advancement demonstrates the ongoing evolution of RDFMiner as a leading tool in semantic web research, particularly in the automated discovery of complex and relevant axioms.

3.3.3 Mining Complex Class Expressions in RDFMiner

Nguyen and Tettamanzi go on to write "Grammatical Evolution to Mine OWL Disjointness Axioms Involving Complex Concept Expressions" and "Using Grammar-Based Genetic Programming for Mining Disjointness Axioms Involving Complex Class Expressions" which extend RDFMiner's capabilities in mining disjointness axioms involving more complex class expressions [268, 266].

Extension of Axiom Discovery This work presents a significant extension in the application of Grammatical Evolution (GE) within RDFMiner. The focus is on mining axioms that incorporate the relational operators of existential quantification ($\exists$) and value restriction ($\forall$). This advancement enables RDFMiner to handle a broader range of complex class expressions, thereby enriching its capability to mine from a variety of topics within the DBpedia dataset.

Updated BNF Grammar for Complex Class Expression Mining Here, the BNF grammar for RDFMiner is updated to generate well-formed OWL class disjointness axioms, especially focusing on complex class expressions involving existential quantification and value restriction. The grammar is organized into static and dynamic parts:

- **Static Part:** The static part of the grammar defines the syntax for disjointness axioms, specifying axioms that involve complex expressions with relational operators. These include:

  DisjointClasses(C1 C2)

  where C1 and C2 can be atomic or complex classes like

  DisjointClasses(Building ObjectSomeValuesFrom(hasWings Animals))

  The form of complex axioms is defined as $\exists r.C$ or $\forall r.C$, where r is a property and C is an atomic class. The static part of the grammar includes the following rules:

  (r1) Axiom := ClassAxiom

  (r2) ClassAxiom := DisjointClasses

  (r3) DisjointClasses := 'DisjointClasses' '(' ClassExpression ' ' ClassExpression ')'

  (r4) ClassExpression := Class | ObjectSomeValuesFrom | ObjectAllValuesFrom

  (r5) ObjectSomeValuesFrom := 'ObjectSomeValuesFrom' '(' ObjectProperty ' ' Class ')'

  (r6) ObjectAllValuesFrom := 'ObjectAllValuesFrom' '(' ObjectProperty ' ' Class ')'

  (r7) Class := [SPARQL query generated classes]

  (r8) ObjectProperty := [SPARQL query generated object properties]

- **Dynamic Part:** The dynamic part contains production rules for low-level non-

terminals, filled at runtime by querying the RDF dataset. The primitives in this part are ‘Class‘ and ‘ObjectPropertyOf‘. The rules for these primitives are:

(r9) Class := dbo:Plant | dbo:FloweringPlant | dbo:WrittenWork

(r10) ObjectPropertyOf := dbprop:spouse | dbprop:artist

These rules are dynamically generated based on the actual contents of the RDF dataset.

The updated grammar significantly enhances RDFMiner's capability to mine complex class expressions, enabling the extraction of more intricate and informative disjointness axioms from RDF datasets.

3.3.4 Enhancing RDFMiner for Complex Class Subsumption Axioms

Felin et al. further advanced RDFMiner's capabilities by focusing on mining subsumption axioms involving complex class expressions [263]. This work signifies a notable extension in RDFMiner's functional scope, specifically targeting subsumption axioms, which are vital for detailed and accurate ontology representations.

Integration of Complex Class Expressions The research introduces an approach to enrich subsumption axioms with complex class expressions, using relational operators like existential quantification ($\exists$) and universal quantification ($\forall$). This enables RDFMiner to process more intricate class structures, significantly enhancing the detail and depth of the ontology learning process.

BNF Grammar for Complex Class Subsumption Axioms The BNF grammar for generating complex class subsumption axioms in RDFMiner is constructed with

the following static rules:

(r1) Axiom := ClassAxiom

(r2) ClassAxiom := subClassOf

(r3) subClassOf := 'subClassOf' '(' classExpression ' ' classExpression ')'

(r4) classExpression := ObjectSomeValuesFrom |

ObjectAllValuesFrom | ObjectIntersectionOf | Class

(r5) ObjectIntersectionOf := 'ObjectIntersectionOf' '(' Class ' ' Class ')'

(r6) ObjectSomeValuesFrom := 'ObjectSomeValuesFrom' '(' ObjectPropertyOf ' ' Class ')'

(r7) ObjectAllValuesFrom := 'ObjectAllValuesFrom' '(' ObjectPropertyOf ' ' Class ')'

Dynamic rules are also incorporated, retrieved via SPARQL queries for Classes and ObjectPropertyOf from the RDF dataset.

Acceptance/Rejection Index (ARI) for Axiom Evaluation The fitness of axioms in RDFMiner is evaluated using the Acceptance/Rejection Index (ARI) from the Possibilistic Heuristic described in Sec 3.2, which is calculated based on the possibility and necessity measures. The ARI for an axiom ϕ is defined as:

$$\text{ARI}(\phi) = N(\phi) + \Pi(\phi) - 1 \tag{3.8}$$

where $N(\phi)$ is the necessity of the axiom, and $\Pi(\phi)$ is the possibility of the axiom. The ARI value lies in the range of $[-1, 1]$, with higher values indicating a stronger acceptance of the axiom and lower values indicating rejection.

Contribution to Semantic Web Ontology Learning This extension of RDFMiner marks a significant contribution to the field of Semantic Web ontology learning, particularly in the automated discovery of complex and nuanced axioms. The integration

of complex class expressions in subsumption axioms opens new avenues for detailed and accurate ontology construction in the Semantic Web.

3.3.5 Conclusion

The enhancements made to RDFMiner, as discussed in this section, directly contribute to the framework of this thesis. These enhancements enable RDFMiner to handle more sophisticated axiom structures in addition to simple atomic ones. Both are necessary for our research which focuses on semantically analysing axioms. By utilizing RDFMiner's capabilities, our thesis aims to develop more efficient and accurate axiom evaluation techniques that can be applied to large-scale semantic datasets. We utilize RDFMiner in our thesis in the phase of axiom generation, axioms which we use to train and test our models. RDMiner uses the possibilistic heuristic detailed in Section 3.2 to evaluate it's generated axioms. The RDMiner framework encompasses a real-world ontology learning system which we aim to enhance with our work by substituting it's current evaluation method with a much faster yet equally accurate model.

3.4 Bridging Model-Theoretic Concepts and Practical Machine Learning

3.4.1 Fuzzy Implication and Modified Support Vector Clustering

In their work "Predicting the Possibilistic Score of OWL Axioms through Modified Support Vector Clustering," Malchiodi and Tettamanzi propose a novel approach to ontology learning [240]. This method centers on predicting the possibilistic score of OWL axioms, crucial for ontology construction and maintenance.

Method Overview The authors present a method based on support vector clustering, originally designed for inferring the membership functions of fuzzy sets. This

technique is adeptly adapted for predicting the possibilistic score of candidate OWL axioms. The possibilistic score is a measure of an axiom's compatibility with the recorded facts in a knowledge base, grounded in the principles of possibility theory.

Semantic Similarity Measure A pivotal aspect of this approach is the semantic similarity measure between axioms, which is essential for the support vector clustering model. The similarity measure, inspired by the Jaccard index, is defined as follows:

$$\text{sim}(\phi, \psi) = \min\{\text{Impl}(\phi, \psi), \text{Impl}(\psi, \phi)\} \tag{3.9}$$

where ϕ and ψ are OWL axioms. The function *Impl* denotes a fuzzy implication operator, and the similarity is expressed as the minimum of the implications of ϕ by ψ and vice versa, encapsulating a logical conjunction.

Fuzzy Implication Operator The fuzzy implication operator is based on Herbrand semantics:

$$\text{Impl}(\phi, \psi) = \frac{\|\{\mathcal{I} : \mathcal{I} \vDash \neg\phi \vee \psi\}\|}{\|\Omega\|} \tag{3.10}$$

where Ω is the universe set, and $\mathcal{I}$ represents interpretations. This definition requires an approximation due to computational inefficiency in real-world datasets.

Practical Implementation For practical implementation, the focus is on subsumption axioms of the form $A \sqsubseteq B$, where A and B are OWL class expressions, and their negations. The similarity measure is approximated by considering individuals in the RDF dataset that confirm or contradict these axioms:

$$\text{sim}(A \sqsubseteq B, C \sqsubseteq D) = \min\left\{\frac{\|[A] \cap [B] \cup [C] \cap [D]\|}{\|[A] \cup [B]\|}, \frac{\|[C] \cap [D] \cup [A] \cap [B]\|}{\|[C] \cup [D]\|}\right\}, \tag{3.11}$$

$\downarrow \phi\psi \rightarrow$	$C \sqsubseteq D$	$C \not\sqsubseteq D$
$A \sqsubseteq B$	$\frac{\|[A] \cap [B] \cup [C] \cap [D]\|}{\|[A] \cup [B]\|}$	$\frac{\|[A] \cap [B] \cup [C] \cap [\overline{D}]\|}{\|[A] \cup [B]\|}$
$A \not\sqsubseteq B$	$\frac{\|[A] \cap [\overline{B}] \cup [C] \cap [D]\|}{\|[A] \cup [B]\|}$	$\frac{\|[A] \cap [\overline{B}] \cup [C] \cap [\overline{D}]\|}{\|[A] \cup [B]\|}$

Table 3.9: Formulas for computing similarity between positive or negated subsumption axioms [240].

where $[A]$, $[B]$, $[C]$, and $[D]$ denote the extensions of the class expressions in the RDF dataset. This approximation, aligned with the Jaccard index, provides a computationally feasible method for calculating similarity between axioms.

Given that in order to predict the ARI of subsumption axioms we will need to compute similarities between positive or negated subsumption axioms, the formulas to apply in all four cases, as summarized in Table 3.9, are:

The similarity between two candidate OWL axioms of the form $A \sqsubseteq B$ and $C \sqsubseteq D$ can be computed using SPARQL counting queries. For instance, the denominator $\|[A] \cup [B]\|$ may be computed by the following SPARQL query:

```
SELECT (count(DISTINCT ?x) AS ?n)
WHERE { { ?x a A . } UNION { ?x a B . } }
```

This query counts the distinct individuals that are instances of either class A or B, effectively computing the union of their extensions in the dataset.

For computing the numerator $\|[A] \cap [B] \cup [C] \cap [D]\|$ of the similarity measure, a SPARQL query can be utilized to count the distinct individuals that are instances of both classes A and B or both classes C and D. This effectively computes the intersection of the extensions of these class expressions in the dataset. The corresponding SPARQL query is as follows:

```
SELECT (count(DISTINCT ?x) AS ?n)
```

```
WHERE { { Q([A]) . Q([B]) . }
        UNION
        { Q([C]) . Q([D]) . } },
```

where $Q([X])$ is $?x\ a\ X$ for the positive case and $Q([\overline{X}])$ is represented by $FILTER\ NOT\ EXISTS\ \{\ ?x\ a\ X\ \}$ for the negated case.

Possibilistic Axiom Scoring This work utilizes the Possibilistic Heuristic detailed in Sec 3.2. The possibilistic score of an axiom is assessed using a machine learning model trained on a dataset of axioms with precomputed possibilistic scores. The training process involves predicting the possibility of a candidate axiom and its negation, which then facilitates the estimation of the axiom's ARI.

Modified Support Vector Clustering The model employed in the paper is based on a modified version of support vector clustering. This technique, originally developed for learning the membership functions of fuzzy sets, is adapted for the specific task of predicting the possibilistic score of candidate OWL axioms.

Support vector clustering is a method that typically operates by finding a sphere in a high-dimensional space, which encloses the data points. In this modified version, the support vector clustering is adapted to work with the similarity measure between OWL axioms and to handle the specific characteristics of the possibilistic scores.

The modification lies in how the model interprets the possibilistic scores and axiom similarities. Instead of directly clustering the axioms based on their semantic content, the model uses the derived similarity measures (based on the semantic overlap of the axioms) and the possibilistic scores to train a clustering model. This approach allows the model to predict the possibilistic score of a new, unseen axiom based on its similarity to already scored axioms.

Conclusion This paper contributes significantly to ontology learning by proposing a scalable and efficient method for evaluating OWL axioms. The integration of machine learning with a semantically driven similarity measure offers a practical alternative to direct possibilistic scoring. The approach emphasizes the importance of adapting sophisticated theoretical models to practical applications, particularly in managing large volumes of semantic web data.

3.4.2 Predicting Possibilistic Scores with Support Vector Regression

In the paper "Predicting the Possibilistic Score of OWL Axioms through Support Vector Regression" by Malchiodi et al. extend their previous work on ontology learning, focusing this time on using support vector regression (SVR) for predicting possibilistic scores of OWL axioms [238].

The Shift to Support Vector Regression The main novelty of this paper compared to the previous one lies in the application of support vector regression (SVR). Unlike the modified support vector clustering used previously, SVR provides a more direct and simplified approach to axiom scoring, enhancing both efficiency and scalability.

Experimentation and Results The authors conducted experiments with 722 subClassOf axioms and their negations, involving a total of 1444 formulas. These were tested against the DBpedia dataset to compute similarity measures and the possibility of each formula. The experiments revealed that ridge regression, a variant of SVR, showed better results than ε-insensitive regression in terms of root mean square error (RMSE), median, and standard deviation of errors, both for acceptability and ARI (Acceptance/Rejection Index) prediction. Notably, the training time for ridge re-

gression was dramatically lower compared to the fuzzy-based system used previously, highlighting the efficiency of the SVR approach.

Performance Comparison The results indicated that while the original fuzzy-based approach outperformed the SVR method in terms of accuracy, the SVR method was significantly faster. Specifically, the training and tuning of the fuzzy-based system took eight hours, compared to just two minutes for the ridge regression, representing a substantial reduction in computational time.

3.5 Syntactic VS Semantic Similarity Measures and Does Dimensionality Reduction Matter?

In "Classifying Candidate Axioms via Dimensionality Reduction Techniques", Malchiodi et al. explore the effectiveness of different similarity measures in classifying OWL axioms [239].

Dimensionality Reduction Techniques Two key dimensionality reduction techniques were employed:

1. **Kernel-based Principal Component Analysis (PCA)**: This technique applies PCA to data nonlinearly mapped onto a higher-dimensional space, extracting principal components to represent the data efficiently.

2. **t-distributed Stochastic Neighbor Embedding (t-SNE)**: t-SNE minimizes the Kullback-Leibler divergence between two distributions, one representing the similarities of data points in the high-dimensional space and the other in the reduced space.

These techniques were crucial in mapping axioms into a lower-dimensional space for effective application of machine learning algorithms for classification.

Syntactic vs Semantic Similarity Measures The authors tested various similarity measures, categorized into syntactic and semantic approaches:

- **Syntactic Measures:**
  - *Length-based Similarity (simlen)*: This measure compares the textual representation length of two formulas. It is a basic measure, focusing only on string lengths, and is not expected to capture meaningful semantic information. However, it serves as a baseline for comparison.

$$simlen(\phi_1, \phi_2) = 1 - \frac{|\#\phi_1 - \#\phi_2|}{\max\{\#\phi_1, \#\phi_2\}} \tag{3.12}$$

  - *Hamming Similarity (simH)*: This measure is a bit more sophisticated and considers the normalized Hamming distance between the textual representations of formulas. It evaluates the fraction of positions where the strings contain different characters, aligning them on the left and ignoring extra characters in the longer string.

$$simH(\phi_1, \phi_2) = \begin{cases} H(\phi_1, \phi_2), & \text{if } \operatorname{sgn}(\phi_1) = \operatorname{sgn}(\phi_2) \\ 1 - H(\operatorname{abs}(\phi_1), \operatorname{abs}(\phi_2)), & \text{otherwise} \end{cases} \tag{3.13}$$

  - *Levenshtein Similarity (simedit)*: This measure utilizes the normalized Levenshtein distance between strings, representing the smallest number of atomic operations needed to transform one string into the other. It captures more complex syntactical and some simple semantic similarities.

$$simedit(\phi_1, \phi_2) = \begin{cases} Lev(\phi_1, \phi_2), & \text{if } \operatorname{sgn}(\phi_1) = \operatorname{sgn}(\phi_2) \\ 1 - Lev(\operatorname{abs}(\phi_1), \operatorname{abs}(\phi_2)), & \text{otherwise} \end{cases} \tag{3.14}$$

- **Semantic Measure:**

– *Jaccard Similarity (simJ)*: This is the only measure among those tested that takes into account the specific form of the formulas and their meaning within a dataset. It is closer to a semantics-based approach and uses the Jaccard similarity index formula [238, 239, 240]. It is the same measure developed in the works introduced in Sec 3.4.

$$simJ(\phi_1, \phi_2) = \frac{\|[A] \cap [B] \cup [C] \cap [D]\|}{\|[A] \cup [B]\|} \tag{3.15}$$

where ϕ_1 is $A \sqsubseteq B$ and ϕ_2 is $C \sqsubseteq D$.

Comparative Performance of Similarity Measures The study's statistical analysis revealed a significant disparity in performance between the syntactic and semantic similarity measures. The semantic-based Jaccard similarity ($simJ$) consistently outperformed the syntactic measures across various machine learning models and dimensionality reduction techniques. This was evident in the superior clustering and classification accuracy achieved by $simJ$, regardless of the computational approach used.

The statistical analysis involved a comparison using several machine learning models, including Random Forests (RF), Naive Bayes (NB), Linear Discriminant Analysis (LDA), Multi-Layer Perceptron (MLP), and Support Vector Classifiers (SVC). These models were tested across different dimensions, but in Table 3.10 we focused on $d = 2$ for PCA, as PCA generally outperformed t-SNE, and the number of dimensions (d) showed minimal influence on results. Notably, the length-based similarity measure ('simlen') consistently yielded poor results and is thus excluded from the table.

Implications and Future Directions These findings suggest that semantic understanding plays a critical role in ontology learning and axiom classification. While syntactic measures offer computational simplicity, they may lack depth in contexts

Model	simH	simedit	simJ
RF	0.67	0.67	**0.83**
NB	0.62	0.69	**0.82**
LDA	0.52	0.68	**0.82**
MLP	0.62	0.66	**0.82**
SVC	**0.60**	**0.60**	**0.60**

Table 3.10: Median test set accuracy of algorithms using PCA (d=2) with different similarity measures [239].

requiring nuanced semantic analysis. The prominent performance of the Jaccard measure indicates the need for semantic-aware approaches in automated ontology learning tools.

3.5.1 Conclusion

The exploration of merging model-theoretic concepts with practical machine learning techniques, inspires a core innovation in our thesis. These concepts are necessary for developing a sophisticated, machine learning-based framework for axiom evaluation that surpasses traditional methods in both efficiency and scalability while maintaining accuracy. By predicting possibilistic scores using a machine learning model, the approaches presented in this section offer a novel way to assess axiom validity against existing knowledge bases. Although this work presents challenges, such as suboptimal accuracy and slow similarity calculation processes, it demonstrates that such approaches are achievable and provide viable alternatives to much slower traditional evaluators. In this thesis, we use this work to benchmark and gauge the performance of our models as well as the effectiveness of our innovative similarity measures.

The evaluation and representation of axioms within the domain of ontology have advanced significantly through the application of embedding techniques. These techniques, which transform complex, high-dimensional information into compact, lower-dimensional, and semantically rich representations, have become pivotal in the realm of semantic web and knowledge representation. This section delves into various state-of-the-art embedding methods, as discussed in recent literature, highlighting their unique approaches and contributions to axiom evaluation.

3.6.1 Onto2Vec

Method Overview Onto2Vec, as presented in the work "Onto2Vec: joint vector-based representation of biological entities and their ontology-based annotations", combines symbolic inference (automated reasoning) with statistical representation learning to generate vector-based representations of ontology classes and biological entities annotated with these classes [336]. This approach leverages the HermiT OWL reasoner to infer new logical axioms (like equivalent class, subclass, and disjointness axioms), thus enhancing the richness of the ontology's semantic content.

Representation Learning using Word2Vec The core of Onto2Vec's method is the application of the skip-gram model from Word2Vec [249] to learn representations of each class or property in an ontology. Given an ontology O, the deductive closure O^* is first computed, which includes both the axioms in O and the set of inferred axioms. Every axiom in O^* is treated as a sentence, forming a corpus, with the vocabulary consisting of classes, relations, and keywords used in the OWL axioms.

The skip-gram model is then used to learn the representation of each word in the corpus. Formally, given a sequence of training words $\omega_1, \omega_2, \ldots, \omega_T$, the skip-gram

model aims to maximize the average log likelihood:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(\omega_{t+j}|\omega_t) \tag{3.16}$$

Here, c is the size of the training context, T is the size of the set of training words, and ω_i is the i-th training word in the sequence.

Similarity Measures in Onto2Vec In Onto2Vec, the primary similarity measure employed is cosine similarity, which is crucial for evaluating relationships between entities in the vector space. Cosine similarity quantifies the cosine of the angle between two vectors, reflecting their degree of alignment. Given two vector representations $\mathbf{v}_1$ and $\mathbf{v}_2$, cosine similarity is defined as:

$$\text{cosine\_similarity}(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|\|\mathbf{v}_2\|} \tag{3.17}$$

Here, $\mathbf{v}_1 \cdot \mathbf{v}_2$ denotes the dot product of the two vectors, and $\|\mathbf{v}_i\|$ is the norm of vector $\mathbf{v}_i$. This measure effectively captures the degree of alignment between two vectors, with a value of 1 indicating perfect alignment and -1 indicating perfect opposition.

Additionally, for baseline comparison, the paper uses Resnik similarity, a well-known semantic similarity measure in bioinformatics. It provides a standard against which the effectiveness of Onto2Vec's embeddings can be evaluated, particularly in the context of predicting protein-protein interactions.

Applications and Evaluations Onto2Vec has been applied to the Gene Ontology (GO), producing dense vector representations of proteins, GO classes, and the axioms that govern these classes. The method's effectiveness is demonstrated through applications in predicting protein-protein interactions and clustering protein families.

Method	AUC Value (Yeast)	AUC Value (Human)
Resnik	0.7942	0.7891
Onto2Vec	0.7701	0.7614
Onto2Vec NoReasoner	0.7439	0.7385
Binary GO	0.6912	0.6712
Onto BMA	0.6741	0.6470
Onto AddVec	0.7139	0.7093
Onto2Vec LR	0.7959	0.7785
Onto2Vec SVM	**0.8586**	**0.8621**
Onto2Vec NN	**0.8869**	**0.8931**
Binary GO LR	0.7009	0.7785
Binary GO SVM	0.8253	0.8068
Binary GO NN	0.7662	0.7064

Table 3.11: AUC values of ROC curves for PPI prediction. The best AUC value among all methods is shown in bold [336].

Table 3.11 shows an example of the performance of Onto2Vec when compared to a baseline (Resnik) and using different configurations. In particular, Onto2Vec's representations have shown potential in significantly improving the accuracy of protein-protein interaction predictions compared to traditional semantic similarity measures.

Conclusion Onto2Vec represents a significant advancement in the representation of biological entities and ontology-based annotations. Its ability to encode both the ontology structure and an entity's annotations in a single representation offers a novel approach to ontology-based machine learning applications in bioinformatics.

3.6.2 OPA2Vec

Method Overview OPA2Vec (Ontologies Plus Annotations to Vectors), as described in "OPA2Vec: combining formal and informal content of biomedical ontologies to improve similarity-based prediction", progresses beyond Onto2Vec by integrating both the formal logical content of ontologies (asserted and inferred logical axioms) and the meta-data present as annotation axioms [337]. This approach enriches the embedding process by including natural language descriptions, labels, and other annotations associated with ontology entities, thereby providing a more comprehensive semantic representation.

Combining Formal and Informal Content The key innovation in OPA2Vec lies in its ability to process and vectorize a mixture of formal ontology axioms and informal, human-readable annotations. This is accomplished through the following steps:

1. Generating sentences from OWL annotation axioms to form a corpus, thereby translating both the formal and informal content into a format suitable for vectorization.

2. Applying a Word2Vec model, pre-trained on PubMed abstracts, to this combined corpus. This use of transfer learning allows OPA2Vec to leverage existing knowledge in biomedical literature for better encoding of natural language phrases and statements.

Differences from Onto2Vec and Novel Contributions While Onto2Vec focuses on embedding the logical axioms of ontologies, OPA2Vec extends this by:

1. Incorporating the ontology's meta-data, such as natural language annotations,

thus capturing a broader semantic spectrum.

2. Utilizing transfer learning with a pre-trained Word2Vec model, which significantly enhances the representation of natural language components within the ontology.

3. Demonstrating improved performance in applications like protein-protein interaction prediction and gene-disease association prediction, surpassing both Onto2Vec and traditional semantic similarity measures.

Similarity Measures and Evaluations Similar to Onto2Vec, OPA2Vec uses cosine similarity for comparing vector representations. Additionally, OPA2Vec's enriched embeddings are evaluated against traditional semantic similarity measures, like Resnik similarity, showing its superior performance in predicting protein-protein interactions and gene-disease associations. Tables 3.12 and 3.13 illustrate the performance of OPA2Vec compared with Onto2Vec with different configurations.

Method	**AUC (Human)**	**AUC (Yeast)**
OPA2Vec (Labels)	0.7973	0.7973
OPA2Vec (Description)	0.8298	0.8298
OPA2Vec (Synonyms)	0.7769	0.7769
OPA2Vec (Creation Date)	0.7494	0.7494
OPA2Vec (Created By)	0.7697	0.7697
OPA2Vec (OBOnamespace)	0.7587	0.7587
Onto2Vec	0.7614	0.7614

Table 3.12: AUC values of ROC curves for PPI prediction for different annotation properties in human and yeast datasets [337].

Method	AUC (Human)	AUC (Mouse)
Onto2Vec	0.7841	0.8431
OPA2Vec (No Pre-training)	0.8104	0.8755
OPA2Vec	0.8379	0.9013
OPA2Vec (NN)	0.8676	0.9304
Resnik	0.8291	0.9041

Table 3.13: ROC curves and AUC values for gene-disease association prediction performance of different methods for human and mouse datasets [337].

Conclusion OPA2Vec represents a considerable advancement in the field of ontology embeddings by effectively combining formal ontology content with rich annotation meta-data. Its approach provides a more nuanced understanding of biological entities, illustrating the power of integrating structured and unstructured data in ontology-based applications.

*3.6.3 OWL2Vec**

Method Overview OWL2Vec*, as introduced in "Embedding OWL Ontologies with OWL2Vec?" and detailed in "OWL2Vec*: Embedding of OWL Ontologies", represents a significant evolution in the field of ontology embeddings [170, 70]. Unlike its predecessors, Onto2Vec and OPA2Vec, which focus on embedding logical axioms and annotation metadata, respectively, OWL2Vec* integrates three dimensions: the graph structure of the ontology, the lexical information, and logical constructors. This multifaceted approach allows OWL2Vec* to capture a broader and more nuanced spectrum of ontology semantics.

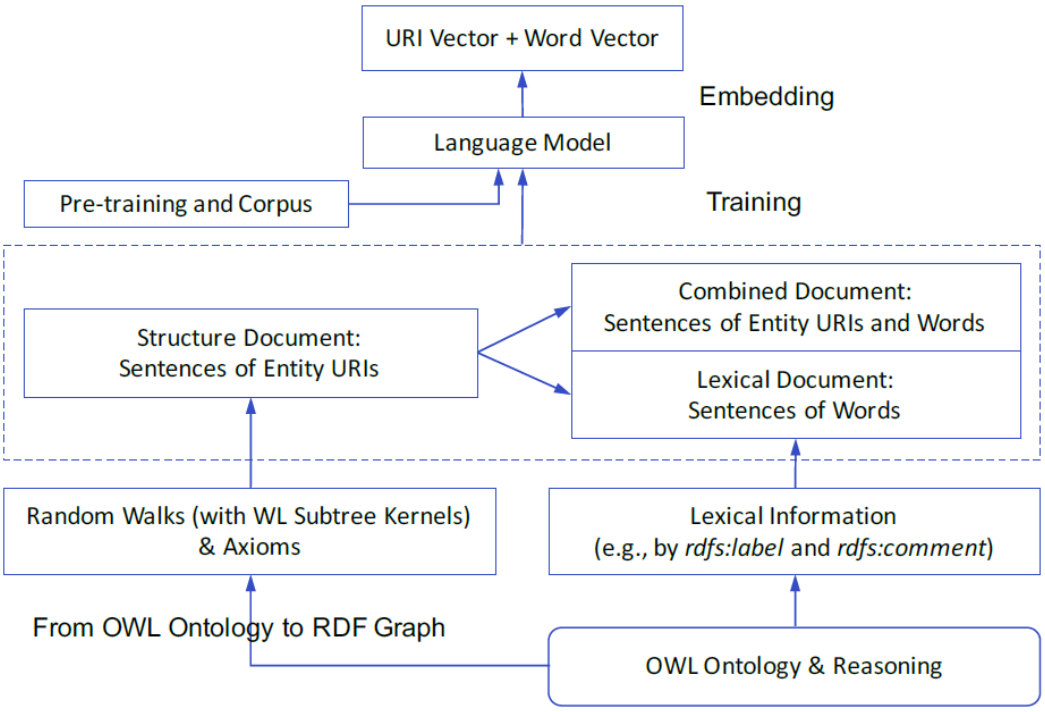


Figure 3.1: The overall framework of OWL2Vec*, illustrating the process from OWL Ontology to RDF Graph transformation, corpus generation with structural, lexical, and combined documents, and the subsequent embedding training using a language model [70].

Enhanced Embedding Techniques The core innovation in OWL2Vec* lies in its ability to extract and learn from a rich variety of data sources within an ontology:

1. **Graph Structure**: The approach includes not only the direct translation of ontology axioms but also an exploration of the ontology's graph-based structure, offering a more comprehensive view of the relationships and hierarchies present in the ontology.

2. **Lexical Information**: OWL2Vec* delves deeper into the lexical content of the ontology, extracting and vectorizing natural language annotations, labels, and comments, thus enriching the embeddings with human-readable context and

providing higher dimensional vectors.

3. **Logical Constructors**: The inclusion of logical constructors in the embedding process allows OWL2Vec* to account for the inherent logical complexity of ontologies, a feature not fully explored in Onto2Vec or OPA2Vec.

OWL2Vec* Strategies and Ontology Projection OWL2Vec* employs a sophisticated framework to compute semantic embeddings from OWL 2 ontologies, involving several interconnected strategies shown in Figure 3.1:

1. **Ontology Projection (From OWL Ontology to RDF Graph)**: The ontology is projected into a graph form, where nodes represent concepts, and edges are labeled with relations between these concepts. This projection is critical as it translates complex OWL 2 axioms into a graph structure, making it easier to navigate and analyze the ontology's semantic relationships. For instance, a triple $A \,\mathrm{Ro}\, B$ in the graph is justified by one or more axioms entailed by the ontology, semantically relating two concepts A and B via a property Ro [170].

2. **Walk Strategy**: Diverse strategies are applied to traverse the projected ontology graph. A modified RDF2Vec [300] approach with weighted edges is used to highlight specific relationships. A node2vec [144] inspired strategy offers scalability and flexibility, allowing for the biasing of walks to enhance semantic similarities across different ontology structures. These strategies are essential for generating varied and rich corpora that capture the ontology's complexity from different angles.

3. **Corpus Creation**: Using the walk strategies, corpora of sentences are created that reflect the diverse paths through the ontology graph. The variety in walking

strategies leads to different types of corpora, each offering unique insights into the ontology's structure and semantics.

4. **Concept Embeddings**: The resulting corpora are then used to generate concept embeddings with distinct characteristics. Techniques like Word2Vec and FastText are employed for this purpose. The choice of walking strategy and corpus type directly influences the nature of these embeddings, allowing for tailored representations that emphasize different semantic aspects of the ontology.

The implementation of these strategies in OWL2Vec* is specifically designed to address the multifaceted nature of ontologies. The use of ontology projection simplifies the complex structure of OWL 2 axioms for better navigability and analysis. Diverse walking strategies increase the number and diversity of the generated corpora, ensuring that the embeddings capture a wide range of semantic relationships and structures inherent in the ontology. This comprehensive approach is vital for creating embeddings that accurately reflect the rich and varied semantics of ontologies.

Comparison with Onto2Vec and OPA2Vec OWL2Vec* advances beyond Onto2Vec and OPA2Vec by:

- Merging structural, lexical, and logical elements into a cohesive embedding framework, thereby providing a more holistic representation of ontologies.
- Demonstrating superior performance in tasks such as class membership and subsumption prediction across various ontologies, including HeLis, FoodOn, and GO.
- Offering enhanced versatility and adaptability to different ontological structures and requirements, as evidenced by its empirical performance.

Technical Details and Performance Metrics OWL2Vec* showcases its effectiveness through a range of performance metrics. For instance, in class membership prediction and subsumption prediction tasks, OWL2Vec* significantly outperforms both Onto2Vec and OPA2Vec, as well as other baseline methods. Specific performance metrics, such as Mean Reciprocal Rank (MRR) and Hits@N, indicate the superior accuracy of OWL2Vec* embeddings in capturing the complex semantics of ontologies.

	FoodOn				**GO**			
Method	**MRR**	**Hits@1**	**Hits@5**	**Hits@10**	**MRR**	**Hits@1**	**Hits@5**	**Hits@10**
RDF2Vec	0.078	0.053	0.097	0.119	0.043	0.017	0.057	0.087
TransE	0.029	0.011	0.044	0.065	0.015	0.005	0.018	0.030
TransR	0.072	0.044	0.093	0.130	0.048	0.016	0.076	0.113
EL Embedding	0.040	0.014	0.067	0.099	0.018	0.005	0.021	0.036
Onto2Vec	0.034	0.014	0.047	0.064	0.024	0.008	0.031	0.053
OPA2Vec	0.093	0.058	0.117	0.156	0.075	0.032	0.106	0.157
OWL2Vec	0.091	0.052	0.121	0.152	0.031	0.012	0.040	0.067
Pre-trained Word2Vec	0.136	0.089	0.175	0.227	0.123	0.055	0.177	0.260
OWL2Vec*	**0.213**	**0.143**	**0.287**	**0.357**	**0.170**	**0.076**	**0.258**	**0.376**

Table 3.14: Subsumption Prediction Results for FoodOn and GO [70]

Evaluation and Comparison of Subsumption Prediction The results for subsumption prediction, as presented in Table 3.14, indicate that OWL2Vec* significantly outperforms other methods such as Onto2Vec and OPA2Vec in both the FoodOn and GO datasets. Specifically, OWL2Vec* achieves the highest MRR of 0.213 and 0.170, and also leads in Hits@1, Hits@5, and Hits@10 when compared to the baselines for both datasets. These results highlight the effectiveness of OWL2Vec*'s embedding strategies, which are capable of capturing complex semantic relationships within

and across ontology hierarchies. This is especially notable in the case of FoodOn, where OWL2Vec* nearly doubles the MRR of OPA2Vec, the next best performing method. Similarly, in the GO dataset, OWL2Vec* outperforms OPA2Vec in MRR by a substantial margin. The performance leap from traditional embedding methods to OWL2Vec* demonstrates the potential of incorporating comprehensive ontology features and sophisticated embedding algorithms for enhanced semantic prediction tasks.

Conclusion In summary, OWL2Vec* stands as a pivotal development in ontology embedding techniques. By intricately combining the structural, lexical, and logical facets of ontologies, OWL2Vec* not only achieves superior performance in key ontology-related tasks but also sets a new benchmark for comprehensive semantic representation in the realm of ontology embeddings.

3.6.4 Conclusion

The exploration of ontological embedding techniques such as Onto2Vec, OPA2Vec, and OWL2Vec* is important for our thesis for two reasons. The first one, is to figure out if word embedding techniques offer better paths than semantic similarity-based approaches. Recently, and as of the time of writing this thesis, embedding techniques are being used in different domains and are gaining much attention. This is amplified by the appearance of Large Language Models (LLMs) and the rise of Natural Language Processing (NLP) systems and AIs. Which brings us to the second reason, to show the strength of our work and proposed models and compare them to the state-of-the-art approaches which would not be complete without including those that are based on embeddings. These embedding techniques are viable as they provide a means to convert complex ontological structures into a form that machine

learning models can process. However they fall very short in expressing the semantics of logical formulas and are strictly limited to atomic axioms, as well as being slow. For this, our goal is to propose models that surpass all the shortcomings of methods mentioned in this section.

Chapter 4

LEARNING TO CLASSIFY LOGICAL FORMULAS BASED ON THEIR SEMANTIC SIMILARITY

An important task in logic, given a formula and a knowledge base which represents what an agent knows of the current state of the world, is to be able to guess the truth value of the formula. Logic reasoners are designed to perform inference, that is, to decide whether a formula is a logical consequence of the knowledge base, which is stronger than that and can be intractable in some cases. In addition, under the open-world assumption, it may turn out impossible to infer a formula or its negation. In many practical situations, however, when an agent has to make a decision, it is acceptable to resort to heuristic methods to determine the probable veracity or falsehood of a formula, even in the absence of a guarantee of correctness, to avoid blocking the decision-making process and move forward. In this chapter, we propose a method to train a classification model based on available knowledge in order to be able of accurately guessing whether an arbitrary, unseen formula is true or false. Our method exploits a kernel representation of logical formulas based on a model-theoretic measure of semantic similarity. The results of experiments show that the proposed method is highly effective and accurate.

4.1 Introduction

A defining feature for intelligent agents is their ability to reason, that is to draw logical conclusions from the available premises, which constitute their knowledge [18, 47]. While this capability is very important, an equally important and useful, but weaker, capability for an agent would be to be able to recognize if a given formula

is likely to be true or false, given the current knowledge, *in the current state of the world*, even though not necessarily in general.

As a matter of fact, we often experience situations where our incomplete knowledge would not allow us to make exact inferences, yet this does not prevent us to make decisions, because even when we don't know a fact that we need in order to move forward, we are able to make an educated guess (i.e., a prediction based on what we already know) about the veracity of that fact and proceed with our decision making.

It is this weaker task that we are interested in studying here. We propose a simple but effective idea, which is to train a classifier against the knowledge base of the agent, which may be viewed as a set of formulas labeled with their truth value. The formulas are represented as vectors of similarities to the labeled formulas; to this end, we propose a model-theoretic semantic similarity measure which can be computed efficiently. This kernel-representation and its associated similarity measure are the key ingredients of our proposal.

Recently, a rise of interest in developing connectionist methods for reasoning can be observed, with proposals such the so-called Logic Tensor Networks [29], or the Logic-Integrated Neural Network [326] to integrate the power of deep learning and logic reasoning, or approaches that employ state-of-the-art methods for training deep neural networks to learn to perform some basic ontology reasoning tasks [168]. As a further witness of the attention this research field is attracting, some conferences are beginning to feature tutorials on it, like KDD'21 [353] and there is even an upcoming Dagstuhl seminar on "Machine Learning and Logical Reasoning: The New Frontier" [1]

.

Unlike these approaches, what we propose does not require sophisticated neural architectures or resource-intensive deep learning; in addition, we do not attack the

[1]Dagstuhl Seminar 22291, July 17–22, 2022.

more ambitious challenge of logical deduction, but just that of heuristically guessing (as human beings do), the truth value of a formula, independently of its being logically entailed by the available knowledge.

Actually, what people do in case of incomplete knowledge is to somehow measure the similarity between known/familiar situations and unknown/unfamiliar situations [370]. Several cognitive tasks, such as *learning* and *interpolation* require the concept of similarity to be performed [116]. There exists a vast literature on similarity measures, with many proposal arising in the field of machine learning [73]. However, it appears that the problem of measuring the similarity of logical formulas has been less investigated and, when it has, that's often in relation with specific contexts.

While not directly addressing the problem of defining similarity among logical formulas, Bowles [49] studies the nature of relevance and irrelevance of a proposition with respect to another. His work shares with the definition of semantic similarity that we propose here, a basic intuition, which is that the probability that one proposition is true given that another one is true should play a central role. The definition of relevance proposed by Makinson in [236], instead, is not in line with what we are proposing here because it is defined in terms of letter-sharing. A way of measuring the similarity of a Boolean vector to a given set of Boolean vectors, motivated in part by certain data mining or machine learning problems, was proposed by Anthony and Hammer [16]. A similarity measure for Boolean function was proposed by Fišer *et al.* [126] in a quite different context, that of circuit synthesis, which explains the differences with our proposal. One measure of similarity between functions is the existence of a Lipschitz mapping (with small constant) between them [175]. A problem somehow related to the one we are dealing with is the problem of measuring the similarity between logical arguments, which has been studied by Amgoud and David [13].

Through an empirical validation we show that the framework we propose allows a number of quite standard and unsophisticated classification techniques, like support-vector machines, to learn very accurate models that are capable of "guessing" whether a given, unseen formula is true or false in the current state of affairs, without the need to perform any logical deduction.

The rest of the chapter is structured as follows: Section 4.2 states the problem of formula classification; Section 4.3 defines a semantic similarity measure for formulas that is the cornerstone of the proposed approach. Section 4.4 provides an empirical validation of the approach and Section 4.5 draws some conclusions and suggestions for future work.

4.2 Problem Statement

Let Φ be a set of formulas in a logical language $\mathcal{L}$ and let $\mathcal{I}$ be an interpretation, which represents a particular state of affairs or the current state of the world. Under interpretation $\mathcal{I}$, the formulas in Φ may be labeled as being true or false. One could thus construct a table

$$\begin{bmatrix} \phi_1, & \phi_1^{\mathcal{I}} \\ \phi_2, & \phi_2^{\mathcal{I}} \\ \vdots & \vdots \end{bmatrix},$$

where $\phi_i \in \Phi \subset \mathcal{L}$, for $i = 1, 2, \ldots$, and $\phi_i^{\mathcal{I}}$ is the truth value of ϕ_i according to interpretation $\mathcal{I}$. This table can be viewed as a representation of a knowledge base K consisting of all the formulas $\phi_i \in \Phi$ such that $\phi_i^{\mathcal{I}} = T$ and all the formulas $\neg\phi_i \in \Phi$ such that $\phi_i^{\mathcal{I}} = F$. K represents what an agent knows (or believes) about the current state of affairs but, of course, $\mathcal{I}$, the actual state of affairs, is not known in full, which is like saying that the open world hypothesis holds.

Consider now the problem of guessing or predicting whether a new formula $\psi \notin \Phi$

is true or false in $\mathcal{I}$, given K. To be sure, one could use a reasoner to check whether $K \vdash \psi$ or $K \vdash \neg\psi$. If the reasoner is sound and complete, this can even allow one to decide whether $K \models \psi$ or $K \models \neg\psi$. However, even in cases where $K \not\models \psi$ and $K \not\models \neg\psi$, which are entirely possible in an open world, it would be useful for an agent to be able to make educated guesses at the truth value of ψ. By an *educated guess* we mean a prediction, based on the truth values of the formulas the agent already knows. If a model to make that type of predictions existed and were fast and accurate enough, the agent might even used it *instead* of the reasoner, for time-critical tasks where having a quick answer is more important than having an answer that is guaranteed to be always correct.

What we have just described is a classification problem, where given a set of labeled examples (here, formulas with their truth value), a model is sought for that is able to accurately predict the label of an unseen case (i.e., a new formula).

To solve this problem, we propose to use a kernel representation, i.e., to represent formulas **as vectors of similarities to a restricted set of formulas whose label is already known** (Φ) and to train a classification model on these labeled examples, later to be used to classify new, unseen formulas. To this aim, we will stick to very standard and unsophisticated classification methods.

4.3 Semantic Similarity

We need to define similarity among logical formulas. It is quite obvious that such a similarity should not be based on syntax, due to the fact that formulas with widely different syntactical forms may be equivalent. Now, the semantics of logical formulas is defined in model-theoretic terms. What we are looking for is, therefore, a model-theoretic notion of formula similarity.

To keep technical complications at a minimum and without loss of generality, let

us consider propositional logic. As a matter of fact, more expressive logical languages can be mapped to the propositional case (e.g., description logics and first-order logic under the Herbrand semantics).

Definition 1 (Language) *Let $\mathcal{A}$ be a* finite *set of atomic propositions and let $\mathcal{L}$ be the propositional language such that $\mathcal{A} \cup \{\top, \bot\} \subseteq \mathcal{L}$, and, $\forall \phi, \psi \in \mathcal{L}$, $\neg\phi \in \mathcal{L}$, $\phi \wedge \psi \in \mathcal{L}$, $\phi \vee \psi \in \mathcal{L}$.*

Additional connectives can be defined as useful shorthands for combination of connectives of $\mathcal{L}$, e.g., $\phi \supset \psi \equiv \neg\phi \vee \psi$.

We will denote by $\Omega = \{0,1\}^{\mathcal{A}}$ the set of all interpretations on $\mathcal{A}$, which we may also call the "universe". An interpretation $\mathcal{I} \in \Omega$ is a function $\mathcal{I} : \mathcal{A} \to \{0,1\}$ assigning a truth value $p^{\mathcal{I}}$ to every atomic proposition $p \in \mathcal{A}$ and, by extension, a truth value $\phi^{\mathcal{I}}$ to all formulas $\phi \in \mathcal{L}$; $\mathcal{I} \vDash \phi$ means that $\phi^{\mathcal{I}} = 1$ ($\mathcal{I}$ is a model of ϕ); if $S \subseteq \mathcal{L}$ is a set of formulas, $\mathcal{I} \vDash S$ means $\mathcal{I} \vDash \phi$ for all $\phi \in S$; $S \vDash \phi$ means that $\forall \mathcal{I} \vDash S$, $\mathcal{I} \vDash \phi$. The notation $[\phi]$ denotes the set of all models of formula $\phi \in \mathcal{L}$: $[\phi] = \{\mathcal{I} \in \Omega : \mathcal{I} \vDash \phi\}$. The semantics of a formula $\phi \in \mathcal{L}$ is the set of its models, $[\phi]$.

We might begin by defining the semantic distance between two formulas ϕ and ψ as the Hamming distance between the two binary string that represent their respective sets of models:

$$d(\phi, \psi) = \sum_{\mathcal{I} \in \Omega} [\phi^{\mathcal{I}} \neq \psi^{\mathcal{I}}], \tag{4.1}$$

where [expr] denotes the indicator function, which equals 1 if expr is true and 0 otherwise.

According to this definition, $d(\phi, \neg\phi) = \|\Omega\|$ and $d(\phi, \phi) = 0$, which is in good agreement with our intuition. Also, two formulas that are totally unrelated [2] , like

[2] Two formulas may be said to be totally unrelated if knowing the truth value of one does not give any information about the truth value of the other.

say, p and q, where $p, q \in \mathcal{A}$, will have a distance which is half-way in between these two extreme cases, $d(p,q) = \frac{1}{2}\|\Omega\|$.

One problem with this notion of distance is that the distance between two given formulas depends on the number of propositional constants in the language, which is a little counter-intuitive. For instance, $d(p,q) = 2$ if $\mathcal{A} = \{p,q\}$, but $d(p,q) = 4$ if $\mathcal{A} = \{p,q,r\}$, and so on. In addition, to compute it, we have to consider all interpretations in Ω, even though many of them might be indifferent when it comes to two given formulas: for example, pqr and $pq\bar{r}$ are indifferent when comparing p to q.

The former problem disappears if, instead of a distance, we define a similarity, ranging between 0 and 1 based on the same idea, as follows:

$$\mathrm{sim}(\phi,\psi) = \frac{1}{\|\Omega\|}\sum_{\mathcal{I}\in\Omega}[\phi^{\mathcal{I}} = \psi^{\mathcal{I}}]. \tag{4.2}$$

The latter problem is also solved by defining $\mathcal{A}_\phi \subseteq \mathcal{A}$ as the set of atoms that occur in formula ϕ and by letting $\Omega_{\phi,\psi} = 2^{\mathcal{A}_\phi \cup \mathcal{A}_\psi}$; Equation 4.2 can now be rewritten as

$$\mathrm{sim}(\phi,\psi) = \frac{1}{\|\Omega\|}\sum_{\mathcal{I}\in\Omega}[\phi^{\mathcal{I}} = \psi^{\mathcal{I}}] = \frac{1}{\|\Omega_{\phi,\psi}\|}\sum_{\mathcal{I}\in\Omega_{\phi,\psi}}[\phi^{\mathcal{I}} = \psi^{\mathcal{I}}]. \tag{4.3}$$

According to this definition, for any formula $\phi \in \mathcal{L}$, $\mathrm{sim}(\phi,\phi) = 1$, $\mathrm{sim}(\phi,\neg\phi) = 0$ and, no matter how many atoms are involved, if $\mathcal{A}_\phi \cap \mathcal{A}_\psi = \varnothing$, $\mathrm{sim}(\phi,\psi) = \frac{1}{2}$.

Another interesting property of this semantic similarity is the following, which ensures that the proposed similarity is consistent with logical negation.

Theorem 1 *Let ϕ ψ be any two formulas of $\mathcal{L}$. Then*

$$\mathrm{sim}(\phi,\psi) = 1 - \mathrm{sim}(\neg\phi,\psi).$$

Proof: For all interpretation $\mathcal{I}$, $\phi^{\mathcal{I}} = \psi^{\mathcal{I}} \Leftrightarrow \neg\phi^{\mathcal{I}} \neq \psi^{\mathcal{I}}$ and $\phi^{\mathcal{I}} \neq \psi^{\mathcal{I}} \Leftrightarrow \neg\phi^{\mathcal{I}} = \psi^{\mathcal{I}}$.

Therefore, $\{\mathcal{I} : \phi^{\mathcal{I}} = \psi^{\mathcal{I}}\} = \{\mathcal{I} : \neg\phi^{\mathcal{I}} \neq \psi^{\mathcal{I}}\} = \Omega \setminus \{\mathcal{I} : \neg\phi^{\mathcal{I}} = \psi^{\mathcal{I}}\}$ and we can thus write

$$\begin{aligned}
\mathrm{sim}(\phi,\psi) &= \frac{1}{\|\Omega\|}\sum_{\mathcal{I}\in\Omega}[\phi^{\mathcal{I}} = \psi^{\mathcal{I}}] = \frac{1}{\|\Omega\|}\|\{\mathcal{I} : \phi^{\mathcal{I}} = \psi^{\mathcal{I}}\}\| \\
&= \frac{1}{\|\Omega\|}\|\Omega \setminus \{\mathcal{I} : \neg\phi^{\mathcal{I}} = \psi^{\mathcal{I}}\}\| = \frac{\|\Omega\|}{\|\Omega\|} - \frac{1}{\|\Omega\|}\|\{\mathcal{I} : \neg\phi^{\mathcal{I}} = \psi^{\mathcal{I}}\}\| \\
&= 1 - \frac{1}{\|\Omega\|}\sum_{\mathcal{I}\in\Omega}[\neg\phi^{\mathcal{I}} = \psi^{\mathcal{I}}] = 1 - \mathrm{sim}(\neg\phi,\psi).
\end{aligned}$$

□

Another interesting property of the semantic similarity, as we have defined it, is that, if $\Omega_{\phi,\psi}$ is too large, we are not obliged to perform an exact computation of $\mathrm{sim}(\phi,\psi)$, but we can approximate it with acceptable accuracy by randomly sampling n interpretations from $\Omega_{\phi,\psi}$ and counting for how many of them $\phi^{\mathcal{I}} = \psi^{\mathcal{I}}$. Indeed, $\mathrm{sim}(\phi,\psi)$ may be construed as a probability, namely the probability that, in a random interpretation, ϕ and ψ are both true or both false. What we get is an unbiased estimator of $\mathrm{sim}(\phi,\psi)$, which behaves like a binomial parameter $\hat{s}_{\phi,\psi}$, whose confidence interval is given by the Wald confidence interval, based on the asymptotic normality of $\hat{s}_{\phi,\psi}$ and estimating the standard error. This $(1-\alpha)$ confidence interval for $\mathrm{sim}(\phi,\psi)$ would be

$$\hat{s}_{\phi,\psi} \pm z_{\alpha/2}\sqrt{\hat{s}_{\phi,\psi}(1-\hat{s}_{\phi,\psi})/n}, \tag{4.4}$$

where z_c denotes the $1-c$ quantile of the standard normal distribution.

For example, if we set $n = 30$, with a 99% confidence, the actual similarity will be within a deviation of $2.576\sqrt{1/120} = 0.2351$ from $\hat{s}_{\phi,\psi}$, in the worst case, which corresponds to $\hat{s}_{\phi,\psi} = 0.5$; for $n = 100$, the approximation error will be less than 0.1288 and for $n = 1000$ it will be less than 0.0407. As a matter of fact, a precise computation of the similarity between formulas is not really required for the proposed approach to work.

This also suggests a way to deal with non-finite interpretations, which might arise in expressive languages involving variables and functions.

4.4 Experiments and Results

4.4.1 An Example from the Block World

As a first test and example of our proposal, we define a language with four individual constants, A, B, C, *Table*, one unary predicate, $\mathsf{covered}(\cdot)$, and one binary predicate $\mathsf{on}(\cdot,\cdot)$. The Herbrand base of this language is finite and consists of twenty ground atoms, but we can only consider a subset of it, after dropping atoms like $\mathsf{on}(A, A)$, $\mathsf{covered}(\mathit{Table})$, and $\mathsf{on}(\mathit{Table}, A)$, which would always be false in every state of the block world:

$$\begin{aligned}\mathcal{A}_{12} = \{\ & \mathsf{covered}(A), \ \mathsf{on}(A,B), \ \mathsf{on}(A,C), \ \mathsf{on}(A,\mathit{Table}),\\ & \mathsf{covered}(B), \ \mathsf{on}(B,A), \ \mathsf{on}(B,C), \ \mathsf{on}(B,\mathit{Table}),\\ & \mathsf{covered}(C), \ \mathsf{on}(C,A), \ \mathsf{on}(C,B), \ \mathsf{on}(C,\mathit{Table}) \ \ \}.\end{aligned}$$

Notice that, given this $\mathcal{A}_{12}$, $\|\Omega_{12}\| = 2^{12} = 4,096$. By adding another block D to this world we can obtain a larger set of atoms

$$\begin{aligned}\mathcal{A}_{20} = \{\ & \mathsf{covered}(A), \ \mathsf{on}(A,B), \ \mathsf{on}(A,C), \ \mathsf{on}(A,D), \ \mathsf{on}(A,\mathit{Table}),\\ & \mathsf{covered}(B), \ \mathsf{on}(B,A), \ \mathsf{on}(B,C), \ \mathsf{on}(B,D), \ \mathsf{on}(B,\mathit{Table}),\\ & \mathsf{covered}(C), \ \mathsf{on}(C,A), \ \mathsf{on}(C,B), \ \mathsf{on}(C,D), \ \mathsf{on}(C,\mathit{Table}),\\ & \mathsf{covered}(D), \ \mathsf{on}(D,A), \ \mathsf{on}(D,B), \ \mathsf{on}(D,C), \ \mathsf{on}(D,\mathit{Table}) \ \ \},\end{aligned}$$

of size 20, with $\|\Omega_{20}\| = 2^{20} = 1,048,576$, and, similarly, by adding a further block E, a set $\mathcal{A}_{30}$ of 30 atoms, with $\|\Omega_{30}\| = 2^{30} = 1,073,741,824$.

The language may then be completed by a minimal set of logical operators, $\neg$, $\wedge$, $\vee$. Then we select a reference interpretation, for example

$$\mathcal{I}^*_{12} = \{\mathsf{on}(A, \mathit{Table}), \mathsf{on}(C, \mathit{Table}), \mathsf{on}(B, A), \mathsf{covered}(A)\},$$

corresponding to a given state of a very simple block world containing one table and three blocks, arranged as in Figure 4.1a.

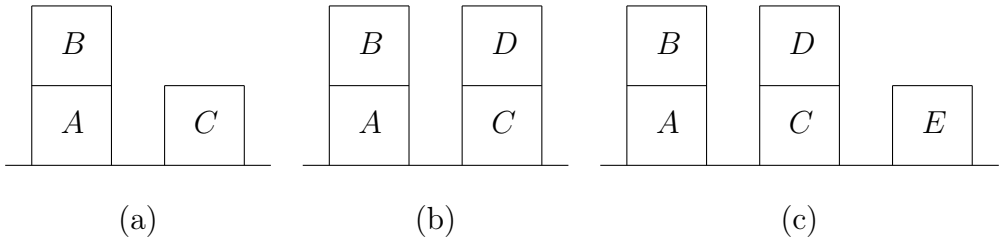


Figure 4.1: The three block worlds corresponding to the reference interpretations, respectively, (a) to $\mathcal{I}^*_{12}$, (b) to $\mathcal{I}^*_{20}$, and (c) to $\mathcal{I}^*_{30}$.

We then generate a set Φ of random logical formulas and we assign them a truth label based on $\mathcal{I}^*$ and train various models on it.

4.4.2 Experimental Protocol

The experiment was divided into two parts. In the first part, we created 3 datasets, each of which is based on a different universe generated using the language explained in Section 4.4.1. The universes are depicted in Figure 4.1. We used these sets to test the performance of models learned using our proposed similarity measure. We considered different universe complexities and used very small training sets to simulate a realistic scenario. For this part, no sampling was done, all interpretations in $\Omega_{\phi,\psi}$ were considered as per Equation 4.3. This can be very time-consuming when the two formulas involve many atoms. For the second part, we created 3 additional sets for each of the universes used in the first part. These additional sets contain the exact same formulas as those in the first part, the only difference being the way the similarity was calculated. To investigate what we mentioned in Section 4.3 regarding the ability to approximate the similarity with acceptable accuracy by randomly sampling

n interpretations, we approximated the similarities for each of the 3 additional sets using $n = 30$, $n = 100$, and $n = 1000$ respectively. We then compared the performance of each of these sets with the base one created in the first part [3] .

Part One: Baseline Experiment.

To see how the proposed method performs, we created the 3 universes depicted in Figure 4.1, and denoted by Ω_{12}, Ω_{20}, and Ω_{30}. The universes consist of 12, 20, and 30 atoms respectively (sets $\mathcal{A}_{12}$, $\mathcal{A}_{20}$, and $\mathcal{A}_{30}$ as defined above). The following are the reference interpretations:

- $\mathcal{I}^*_{12} = \{\mathsf{on}(A, \mathit{Table}), \mathsf{on}(C, \mathit{Table}), \mathsf{on}(B, A), \mathsf{covered}(A)\}$, cf. Figure 4.1a;
- $\mathcal{I}^*_{20} = \{\mathsf{on}(A, \mathit{Table}), \mathsf{on}(C, \mathit{Table}), \mathsf{on}(B, A), \mathsf{on}(D, C), \mathsf{covered}(A), \mathsf{covered}(C)\}$, cf. Figure 4.1b;
- $\mathcal{I}^*_{30} = \{\mathsf{on}(A, \mathit{Table}), \mathsf{on}(C, \mathit{Table}), \mathsf{on}(E, \mathit{Table}), \mathsf{on}(B, A), \mathsf{on}(D, C), \mathsf{covered}(A), \mathsf{covered}(C)\}$, cf. Figure 4.1c;

Following that, we generated 500 random formulas for each of the universes using Algorithm 1. Algorithm 1 generates a given number N of random formulas, taking as input a list of ground atoms $\mathcal{A}$. It is recursive and chooses its next step and which symbols to add at random. It uses a variable that reduces the probability of adding a nested subformula the more complex a formula becomes.

We then labeled each of the formulas with its truth, based on the reference interpretation of its universe. The next step was to create the similarity matrix for each set of formulas. For this part, no sampling was done, in other words all interpretations were taken into account and no noise was added. The similarity between

[3] All the code and data used for the experiments described in this chapter can be found in the following repository: `https://github.com/ali-ballout/Learning-to-Classify-Logical-Formulas-based-on-their-Semantic-Similarity`.

$Truth$	$Formula$	ϕ_0	ϕ_1	$\ldots$	ϕ_m
$Truth_0$	ϕ_0	1	$S_{0,1}$	$\ldots$	$S_{0,m}$
$Truth_1$	ϕ_1	$S_{1,0}$	1	$\ldots$	$S_{1,m}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$Truth_{m-1}$	ϕ_{m-1}	$S_{m-1,0}$	$S_{m-1,1}$	$\ldots$	$S_{m-1,m}$
$Truth_m$	ϕ_m	$S_{m,0}$	$S_{m,1}$	$\cdots$	1

Figure 4.2: Formula similarity matrix with truth labels.

each formula and all other formulas in the set is calculated using Equation 4.3. To simplify, we compare formula ϕ to all other formulas in the set of 500 formulas. At each comparison we check all the unique atoms included in the compared formulas ϕ and ψ, for an example of what atoms are refer to $\mathcal{A}_{12}$ in Section 4.4.1. We then generate all interpretations for this set of unique atoms extracted from both formulas. Then we record the truth for each of the formulas based on each of the generated interpretations. After that, we count the instances where the truth of formulas ϕ and ψ are the same. We divide that number by the total number of generated interpretations and the result obtained is the similarity between ϕ and ψ. We do this, once, for all pairs of formulas to obtain a symmetric similarity matrix of the shape 500×500. Figure 4.2 depicts the similarity matrix between formulas of a set of formulas of size m with S being the similarity between each pair. The diagonal is all 1 since it is the similarity between a formula ϕ and itself. The truth labels of all the formulas are attached as column *Truth* to the similarity matrix to obtain the final product of the process, which is the input to be used for training and testing a machine learning model.

Now that we have created our labeled datasets, we need to choose a machine

learning method that is suitable for the task. Through a process of model selection we decided to use a support vector classifier, it performed the best with the small training sets that we provided. After performing a grid search we determined the best hyper parameters, we set the regularization parameter C to 0.1 and the kernel type to *polynomial*, of degree 3. We ran, for each of the 3 datasets, a 20-fold cross validation processes to establish the baseline performance of our proposed method. All results presented in Table 4.1 are averages of the scores obtained from all the runs for each dataset. Similarly, confusion matrices displayed in Figure 4.3 are the result of summing up all confusion matrices of the 20 runs and then normalizing them. We set the number of formulas included in the training sets of each universe to less than or equal to $\|\mathcal{A}\|$ of said universe, the training set sizes were as follows: 10 for Ω_{12}, 20 for Ω_{20}, and 30 for Ω_{30}. Table 4.2 presents the labeled formulas used in the training set of one of the runs for Ω_{12}.

No agent, human or artificial, has in its knowledge the exhaustive list of all possible formulas. The rationale for using small training sets in our experiments is that the knowledge base of an agent is unlikely to contain many formulas; some of them might be handcrafted and be part of the "background knowledge" of the agent, and the others acquired through sensors or messages received from other agents. In any case, it is a principle of economy that the knowledge of an agent be encoded using as few and as simple formulas as required. We want to test our proposal against a realistic scenario in which the available knowledge (the formulas whose truth value is known) is very small compared to the number of semantically distinct formulas that can be stated in the language. This also has the advantage of demonstrating the generalization capability of our models.

Notice that, for a language whose set of interpretations is Ω, there are $2^{\|\Omega\|}$ semantically distinct formulas, which is a really huge number, although most of them would

be very complicated formulas that one would not expect to find in a real knowledge base. However, even factoring out very complicated formulas, the number of possible formulas would be exceedingly large.

To be sure, there exists a choice of formulas that would make the problem we are studying absolutely trivial. That is when the training set contains at least $\|\mathcal{A}\|$ formulas, each consisting of a single literal (i.e., a positive or negated atom), such that the atoms of these formulas are all distinct. It is easy to see that those formulas, with their truth labeling, would directly give the reference interpretation, from which the truth value of any other formula can be mechanically computed in linear time with respect to the length of the formula, *without performing any logical deduction or reasoning.*

This is the reason why the formulas of each dataset are extracted from a distribution that is skewed in favor of simpler (i.e., realistic), but not too simple formulas. Indeed, we ensure that the training set does not contain literals for all the atoms, by counting the number of literals that are randomly generated and rejecting additional literals once the maximum number of literals has been reached. That maximum is set to $\|\mathcal{A}\|/2$, well below $\|\mathcal{A}\|$.

Since the dataset consists of randomly generated formulas, it is unbalanced [4] , so in addition to the accuracy score, which we report because it gives an intuitive idea of the probability that the prediction is correct, we will provide the Matthews correlation coefficient (MCC) which is a statistical rate between −1 and 1 that produces a high score only if the prediction obtained good results in all of the four confusion matrix categories (true positives, false negatives, true negatives, and false positives), proportionally both to the size of positive elements and the size of negative elements

[4]The dataset for universe Ω_{12} has 272 false formulas and 227 true ones (1 missing because it was a duplicate), for universe Ω_{20} 278 false and 222 true, and for universe Ω_{30} 300 false and 200 true; of course, these figures vary (between training and test set) for every fold obtained from these datasets.

in the dataset. So its a very good metric when we don't have a perfectly balanced dataset [75]. Results of this part of the experiment are presented in Table 4.1.

Part Two: Sampling Experiment.

In this part of our experiment we investigate a property of our proposed similarity, which is the ability to approximate $\text{sim}(\phi, \psi)$ with acceptable accuracy by randomly sampling n interpretations from $\Omega_{\phi,\psi}$. We will name this approximation $\hat{s}_{\phi,\psi}$.

To this end, we created 3 new matrices for each set of formulas used in Section 4.4.2. We ended up with 9 new datasets, 3 for each of the universes depicted in Figure 4.1 and describe in Section 4.4.2. The similarity in the 9 new matrices was calculated differently than in Section 4.4.2. For this part, we approximated the similarity between formulas by randomly sampling a set number n of all interpretations, instead of taking all of them into account as we did in Section 4.4.2. The number of random samples n considered for creating the matrices is $n = 30$, $n = 100$, and $n = 1000$. This allows us to simulate the scenario of having a machine with low computational capacity trying to process a set of interpretations that is too large, and utilizing sampling as a solution. It also allows us to see how the method performs when noise is introduced.

The way the similarity is approximated using sampling is not much different from how it is calculated: we still count the instances where formulas ϕ and ψ have the same truth, but for n randomly sampled interpretations instead of *all* interpretations. Algorithm 2 is used to approximate the similarity between two formulas. In simple terms, when Algorithm 2 compares two formulas ϕ and ψ, instead of generating all interpretations corresponding to the set of unique atoms $\mathcal{A}$ composing those formulas, it generates a number n of these interpretations randomly. This sampling is done with replacement, which means that it is possible that a given interpretation gets sampled

multiple times, especially in case n is larger than the number of all interpretations. We then proceed to count the instances where formulas ϕ and ψ have the same truth out of these sampled interpretations. After that, we divide the obtained number by n, the size of the sample, since we are now dealing with n interpretations and not all of them. The result from that division is the approximation $\hat{s}_{\phi,\psi}$ of the similarity $\text{sim}(\phi, \psi)$ between ϕ and ψ. We do this for the sets of formulas we randomly generated in Section 4.4.2 for each of our universes 3 times, once for each number n of samples we mentioned.

With these 9 new matrices we are able to study how the sample size n might affect the performance of the method when dealing with different universe complexities. It will also show us how well an approximation of the similarity performs. We used the same model to test the performance and the same scoring metrics as the baseline. We used the same training set sizes for each universe as in the baseline. The results of this part of the experiment are available in Table 4.1.

4.4.3 Results and Analysis

Baseline Results

We start our analysis with the first part of our experiment, detailed in Section 4.4.2. The results can be found in Table 4.1 and the corresponding confusion matrices to offer support in Figure 4.3. A small sample of formulas from the test set for the smallest universe, with the labels predicted by the model, is provided in Table 4.3. From this experiment we can determine:

1. The overall performance of our proposed method without sampling while dealing with universes of different complexities, using very small training sets.

2. The effect the training set size has on performance with respect to the complex-

Universe	Training set Size	sample size	Accuracy score	MCC
Ω_{12}	10	no sampling	0.77	0.56
		30	0.76	0.54
		100	0.77	0.56
		1000	0.77	0.55
Ω_{20}	20	no sampling	0.81	0.63
		30	0.81	0.62
		100	0.82	0.63
		1000	0.82	0.65
Ω_{30}	30	no sampling	0.83	0.66
		30	0.79	0.56
		100	0.82	0.62
		1000	0.83	0.64

Table 4.1: Accuracy and MCC for experiments done on each universe.

ity of the universe addressed.

Regarding the first, Table 4.1 shows that the overall performance of our proposed method is good. The highest accuracy achieved was 83% for a training set size of 30 formulas and a universe of complexity 30, MCC being 0.66 which is a very good result when training using an unbalanced set. As a worst case, the method achieved 77% accuracy with a minimal training set size of 10 formulas and universe complexity

of 12, MCC of 0.56 is acceptable considering the small training set relative to the complexity of the universe. Indeed, 30 formulas for a language with 30 propositional symbols is a really sparse training set, when one thinks that this language has $\sim 10^9$ interpretations and there exist $\sim 10^{300,000,000}$ semantically distinct formulas one can construct!

Formula	Label
$(\mathsf{on}(C,B) \wedge \mathsf{on}(B,C)) \vee (\neg\neg\neg\neg\mathsf{on}(A,B) \vee \mathsf{on}(B,A))$	True
$\neg(\neg(\mathsf{on}(A,B) \wedge ((\mathsf{on}(C,Tbl) \vee \neg\neg\mathsf{on}(C,B)) \wedge \mathsf{covered}(A))) \vee (\mathsf{on}(C,A) \vee \mathsf{on}(B,C)))$	False
$\neg(((\mathsf{covered}(B) \vee \neg\mathsf{on}(B,A)) \vee (\mathsf{on}(B,Tbl) \wedge (\mathsf{on}(B,A) \wedge \mathsf{on}(C,A)))) \wedge \mathsf{covered}(B))$	True
$\neg\neg\neg\mathsf{on}(A,C)$	True
$(\mathsf{on}(B,C) \vee \mathsf{covered}(A)) \vee \mathsf{covered}(A)$	True
$\neg(\neg\mathsf{on}(A,C) \wedge \neg\mathsf{covered}(C))$	False
$(\mathsf{on}(A,C) \vee \mathsf{on}(C,A)) \wedge \neg(\mathsf{covered}(C) \wedge \mathsf{on}(A,Tbl))$	False
$\mathsf{on}(C,Tbl) \wedge \mathsf{on}(C,B)$	False
$\mathsf{on}(C,B) \vee \mathsf{on}(B,C)$	False
$\neg\mathsf{on}(C,B) \wedge (\neg\neg\neg\mathsf{on}(C,B) \wedge \mathsf{covered}(B))$	False

Table 4.2: A sample training set made of 10 formulas from universe Ω_{12}.

After demonstrating that our proposed method is capable of achieving good results with very small training sets, we move on to the second point. We can see that the proposed method can achieve an average accuracy of 80% throughout the runs that use a very small training set of 10 formulas for Ω_{12}, 20 formulas for Ω_{20}, and 30 formulas

for Ω_{30}. It would be natural to think that as the universe complexity increases, a model would require a larger training set to maintain performance, which is what the results shown in Table 4.1 and Figure 4.3 confirm. From the results see in Table 4.1 where no sampling was considered, we can see that increasing the number of formulas included in the training set had a very significant effect on performance. This effect was not limited to maintaining performance, but it resulted in an improvement of up to 8% in accuracy and 0.14 in terms of MCC.

To put things into perspective, for Ω_{12} we used for training 10 formulas out of a possible $\sim 10^{1233}$ compared to 30 out of a possible $\sim 10^{300,000,000}$ for Ω_{30}. In other words, the transition from Ω_{12} with 4096 interpretations to Ω_{30} with $\sim 10^9$, resulted in a gain of 8% accuracy by just adding 20 formulas to the training set. The increase in the size of the training set is modest relative to the size of Ω_{30} or the number of semantically distinct formulas that can be constructed, while the gain of accuracy and balance in predictions in terms of MCC is significant.

We observed that the truth value of "simple" formulas turns out to be harder to predict for the trained models than that of "complicated" formulas (see, e.g., Table 4.3). While this must have to do with the geometry of the space of the kernel representation of formulas induced by the semantic similarity, this phenomenon will have to be the object of further investigation.

Sampling Results

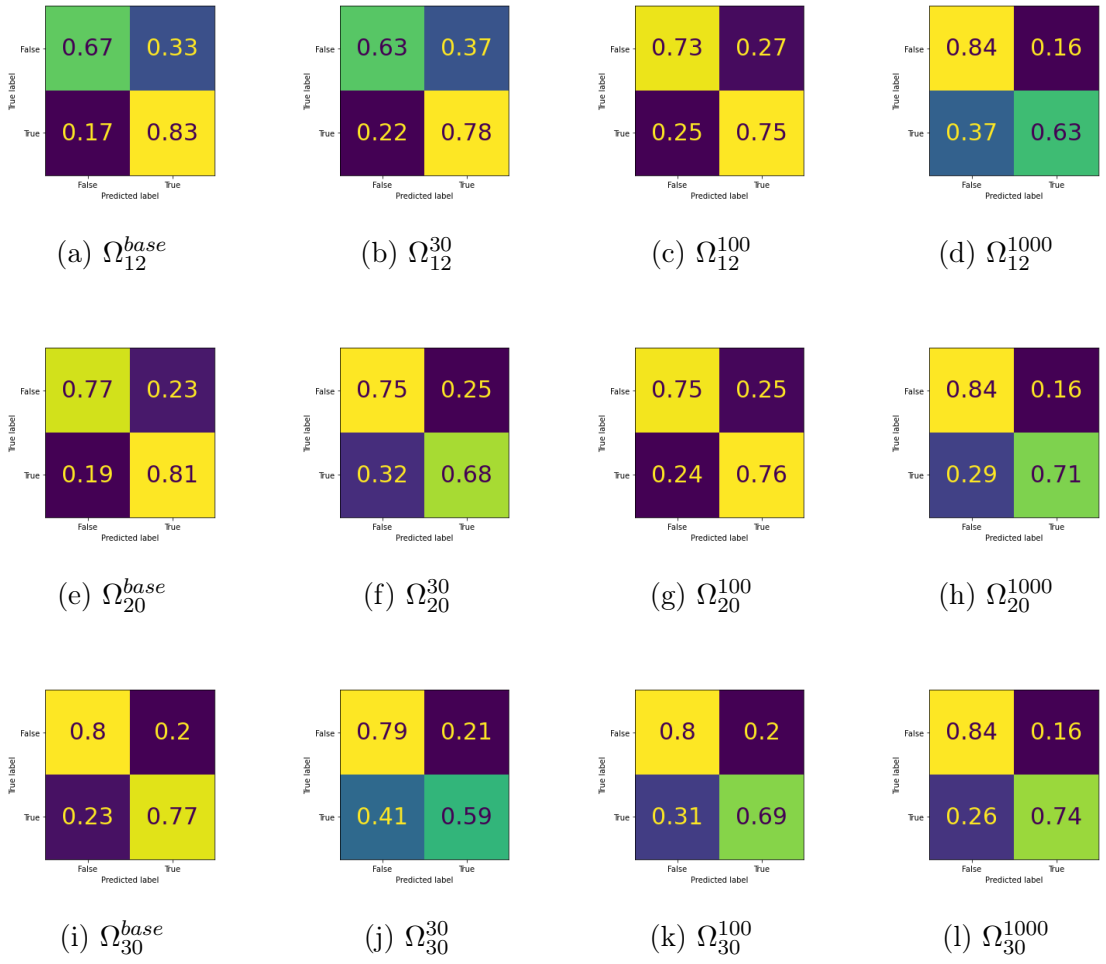
We now shift our attention to part two of the experiment detailed in Section 4.4.2. The results of this experiment are also shown in Table 4.1 and the corresponding confusion matrices to offer support are found in Figure 4.3.

At first glance at Table 4.1, we can tell that the overall performance of the model does not degrade much when the similarity is approximated using the lowest number

of samples n = 30. Indeed, we have a loss of accuracy of almost 4% for 2 of our universes. But this is an acceptable result when considering that in this case we would no longer have to calculate the exact similarity especially when we are limited by computational power. In fact, in this case, we would be looking at 30 random interpretations instead of $\sim 10^9$ for a universe the size of Ω_{30}.

The degradation in accuracy and MCC decreases as we increase the number of samples from 30 to 100 and then to 1000, it even approaches baseline performance. This proves what we mentioned in Section 4.3, we are able to approximate the similarity with very high accuracy even with a low number of sampled interpretations when compared to the number of *all* interpretations.

On the other hand, another increase in the number of samples from 100 to 1000 has no significant effect on performance, which is interesting considering that this introduces noise (since we allow for repetitions) yet it does not degrade performance. It also means that for a universe of complexity $\|\mathcal{A}\|$ there exists an optimal number of samples n that achieves baseline-similar performance.



(a) Ω_{12}^{base} (b) Ω_{12}^{30} (c) Ω_{12}^{100} (d) Ω_{12}^{1000}

(e) Ω_{20}^{base} (f) Ω_{20}^{30} (g) Ω_{20}^{100} (h) Ω_{20}^{1000}

(i) Ω_{30}^{base} (j) Ω_{30}^{30} (k) Ω_{30}^{100} (l) Ω_{30}^{1000}

Figure 4.3: Confusion matrices of the similarity approximation using sampling experiment for each universe. Each row represents 4 cases for each universe, starting with the baseline (no sampling) and then $n = 30$, $n = 100$, and $n = 1000$ respectively. Each sub-figure is captioned by the universe notation $\Omega_{12,20,30}$ and in superscript the sample size n used to approximate similarity.

4.5 Conclusion

We have proposed a framework that allows an agent to train, based on a set of formulas whose truth values are known, a classification model that predicts the truth-

value of a new, arbitrary formula. This framework uses a semantic similarity between formulas, which is a key ingredient of our proposal, to perform a kernel encoding of the formulas, which is then exploited by the classification model. We have tested an implementation of this framework using SVM, showing that the classification model is highly accurate (with accuracy around 80%) even when the similarity is approximated by severely undersampling the interpretations. The practical implications of these results are that the proposed approach is tractable even for languages with a large (or infinite but enumerable) number of atoms; indeed, computing a good approximation of the similarity of two formulas can be done in linear time, as it depends only on the size of the (random) interpretations sampled.

There is no guarantee that all the predictions made by a model be altogether consistent. There is no built-in mechanism to ensure that and the mutual consistency of all the prediction is not part of the measure of the quality of a classifier: every prediction is made by the model and assessed independently of the others. Of course, if the predictions were all correct, they would also be consistent and that's what we observe empirically, that the predictions tend to be mostly consistent.

Since the knowledge of an agent may not be complete, some formulas, which are not entailed by it and whose negation is not entailed either, both predictions would be acceptable, and one might be tempted to count them as correct. However, this is not what we did: for the purpose of testing our method, what we did was to arbitrarily fix one interpretation and say it corresponded to the actual state of affairs; use it to label the training set and evaluate the predictions of the classifiers against the label that would be thus assigned, even for those formulas whose truth value is not constrained by the available knowledge. In a sense, we were as strict as one can be when judging a classifier.

Algorithm 1 Generating random formulas

Require: A set of atoms $\mathcal{A}$

Ensure: $formulas$, a list of generated formulas

$N \leftarrow Number\ of\ random\ formulas\ to\ generate$

$f \leftarrow one\ randomly\ generated\ formula$

$literals \leftarrow 0$ ▷ A counter used to limit the number of literals as sentences

for $i = 1 \rightarrow N$ **do**

  $i \leftarrow i + 1$

  $f \leftarrow$ RANDOM_FORMULA$(\mathcal{A}, 0)$

  if $f\ in\ formulas$ **then**

    $i = i - 1$

    $continue$

  else if $length(f.symbols) == 1$ **then**

    if $literals < length(\mathcal{A})/2$ **then**

      $literals = literals + 1$

    else

      $i = i - 1$

      $continue$

    end if

  else

    $formulas.append(f)$

  end if

end for

function RANDOM_FORMULA($\mathcal{A}$, $level$)

  ▷ Recursive; the nesting $level$, initially $= 0$, is used to progressively reduce the probability of adding nested subformulas

  if RANDRANGE$(level + 4)$ **then** ▷ $\frac{level+3}{level+4}$ probability of stopping

    return CHOICE$(\mathcal{A})$

  end if

  if RANDRANGE$(3) = 0$ **then** ▷ with probability $\frac{1}{3}$

    return $\neg$RANDOM_FORMULA$(\mathcal{A}, level + 1)$

  end if

  return RANDOM_FORMULA$(\mathcal{A}, level + 1) \cdot$ CHOICE$(\wedge, \vee) \cdot$

    RANDOM_FORMULA$(\mathcal{A}, level + 1)$

end function

Note: The generation process is slightly biased towards sentences that are neither too simple, nor too complex.

Algorithm 2 Approximating the similarity by sampling interpretations

Require: 2 formulas ϕ *and* ψ to be compared

Require: a sample size n

Ensure: ϕ *and* ψ, in the same universe Ω

Ensure: $n > 0$

$\mathcal{A} \leftarrow \phi.atoms \cup \psi.atoms$

$interpretations \leftarrow$ SAMPLE_INTERPRETATIONS$(\mathcal{A},\ n)$

$counter \leftarrow 0$

for w in $interpretations$ **do**

  if $\phi.truth(w) == \psi.truth(w)$ **then**

    $counter \leftarrow counter + 1$

  end if

end for

$\hat{s}_{\phi,\psi} \leftarrow \frac{counter}{n}$

function SAMPLE_INTERPRETATIONS$(\mathcal{A},\ n)$

  $interpretations \leftarrow array(size = n)$ ▷ Array to store n interpretations

  for $i = 0 \rightarrow n - 1$ **do**

    $b \leftarrow array(size = \mathcal{A}.length)$ ▷ Array the size of the list of atoms

    for $j = 0 \rightarrow \mathcal{A}.length - 1$ **do**

      $b[j] \leftarrow$ RANDRANGE(2)

    end for

    $interpretations[i] \leftarrow b$

  end for

  return $interpretations$

end function

Formula	Actual	Predicted
$\neg(\neg(\neg(\mathsf{on}(B,A) \wedge \mathsf{covered}(B)) \vee ((\mathsf{covered}(C) \vee \mathsf{on}(A,)B) \wedge \mathsf{on}(C,Tbl))) \vee \neg((\mathsf{on}(B,A) \wedge \mathsf{on}(A,Tbl)) \wedge \neg\mathsf{covered}(A)))$	False	False
$(\mathsf{on}(A,Tbl) \wedge \mathsf{on}(B,A)) \vee \neg\neg((\mathsf{on}(A,C) \wedge ((\mathsf{on}(A,Tbl) \vee \mathsf{covered}(A)) \wedge \mathsf{covered}(A))) \vee \mathsf{on}(C,B))$	True	True
$\mathsf{covered}(B) \wedge \mathsf{on}(A,Tbl)$	False	True
$((\mathsf{covered}(B) \vee (\neg\mathsf{on}(A,B) \wedge \mathsf{on}(C,Tbl))) \wedge (((\mathsf{on}(B,A) \wedge (\mathsf{on}(A,C) \wedge \mathsf{on}(C,Tbl))) \vee (\mathsf{on}(C,A) \vee \mathsf{on}(A,Tbl))) \wedge \mathsf{on}(A,Tbl))) \vee \mathsf{on}(C,A)$	True	True
$((\mathsf{covered}(A) \wedge \mathsf{covered}(B)) \vee (\mathsf{covered}(A) \vee \neg(((\mathsf{covered}(A) \vee \mathsf{on}(B,C)) \vee (\mathsf{on}(B,C) \wedge \mathsf{on}(B,Tbl))) \vee \mathsf{on}(C,A)))) \vee ((\mathsf{on}(B,A) \vee \mathsf{on}(A,C)) \wedge (((\mathsf{on}(C,Tbl) \wedge \mathsf{on}(B,C)) \wedge \mathsf{on}(A,C)) \wedge \mathsf{covered}(A)))$	True	True
$\mathsf{covered}(B) \wedge (((\mathsf{on}(C,B) \vee ((\mathsf{on}(C,B) \wedge \mathsf{on}(B,Tbl)) \wedge ((\neg\mathsf{covered}(B) \vee (\mathsf{on}(A,B) \wedge \neg\mathsf{on}(A,C))) \vee (\neg\neg\mathsf{on}(B,C) \wedge (\mathsf{covered}(C) \vee \mathsf{on}(A,B)))))) \vee \mathsf{covered}(C)) \wedge (\neg(\mathsf{on}(A,C) \vee \mathsf{on}(B,C)) \wedge ((\mathsf{covered}(A) \vee \mathsf{on}(C,B)) \vee (\mathsf{on}(A,Tbl) \wedge (\mathsf{on}(A,Tbl) \vee \mathsf{on}(C,B))))))$	False	False

Table 4.3: A small sample of the test set with formulas varying in complexity from universe Ω_{12}.

Chapter 5

PREDICTING THE SCORE OF ATOMIC CANDIDATE OWL CLASS AXIOMS

Candidate axiom scoring is the task of assessing the acceptability of a candidate axiom against the evidence provided by known facts or data. The ability to score candidate axioms reliably is required for automated schema or ontology induction, but it can also be valuable for ontology and/or knowledge graph validation. Accurate axiom scoring heuristics are often computationally expensive, which is an issue if you wish to use them in iterative search techniques like level-wise generate-and-test or evolutionary algorithms, which require scoring a large number of candidate axioms. In this chapter, we address the problem of developing a predictive model as a substitute for reasoning that predicts the possibility score of candidate class axioms and is quick enough to be employed in such situations. We use a semantic similarity measure taken from an ontology's subsumption structure for this purpose. We show that the approach provided in this chapter can accurately learn the possibility scores of candidate OWL class axioms and that it can do so for a variety of OWL class axioms.

5.1 Introduction

An ontology is the explicit representation of components of a shared conceptualization [145]. In machines, an ontology is a vocabulary which is used by said machine in the representation of knowledge. An AI knowledge-based system would take an ontology as its universe of components and their relations and derive new implicit knowledge within that universe. For a detailed description of an ontology and its components please refer to Section 2.1.2 in Chapter 2.

Ontologies play a vital role in data and knowledge integration by making a com-

mon schema available [154]. Unfortunately, ontology construction is extremely expensive, in terms of both time and resources, and dependent on the availability of knowledge experts [154, 331]. The construction of an ontology for a certain field requires the contribution of a knowledge engineer and of an expert in that specific field. This dependency persists throughout the lifetime of an ontology as an expert is required to develop and expand it as new requirements arise. To overcome such a bottleneck in knowledge acquisition, the field of ontology learning was conceived. Ontology Learning [154, 215, 234] is the task consisting of the automatic generation of ontologies. Ontology learning includes a variety of techniques and those are grouped into:

- Linguistic techniques - Natural language processing mostly used in the preprocessing of data, and some learning tasks such as term extraction [23].
- Statistical techniques - such as data mining and information retrieval methods used to extract terms and associations between them [127, 128].
- Inductive logic programming (ILP) is a branch of machine learning that uses logic programming to generate hypotheses based on prior knowledge and a set of examples [119, 213, 255].

Each one of these technique groups is involved in one or more of the stages of ontology learning, those stages being preprocessing, term and concept extraction, relation extraction, concepts and relations hierarchies, axioms schemata and general axioms [23]. Linguistic techniques can have a role in almost all the stages, but as we mentioned they are mostly used for data preprocessing. We have already introduced some statistical and linguistic techniques in Section 3.2 and Section 3.6.

In comparison, ILP techniques are a sub-field of machine learning that follow exhaustive statistical or linguistic processing. One such example is [228], where the au-

thors describe a method for the automated induction of fuzzy ontology axioms which follows the machine learning approach of ILP named `SoftFOIL`. One of `SoftFOIL`'s limits is a result of its sequential covering strategy. As it uses a greedy search to find rules, it does not guarantee to find the smallest or best set of rules that explain the training examples. Another is susceptibility to being trapped in a loop while searching for the best rule.

A new emerging group of techniques, is a hybrid breed that takes advantage of combining classical ILP and statistical machine learning. To stay in the context of the previous examples, a model would be trained with axioms having their scores assigned by a statistical method, as well as using a similarity measure that results from the logical processing of the data. This is exactly what we detail in Section 3.4.

A weakness of the models proposed by Malchiodi et al. is that they were still reliant on the instances in the data set and querying them to construct the similarity measure (not to figure out an axiom's truth). This meant that even though it is an improvement, it still falls victim to the same problems. The authors explicitly mention that a major weakness of their method is that training such a model consumes a significant amount of resources.

Our work addresses the shortcomings of the previous techniques that heavily rely on error-prone instance-dependent statistics. It is also able to predict the scores of multiple types of axioms and is not bound to simply `subsumption`. We propose a method that can be used as a building-block or an extension/plug-in to other existing statistical analysis or ILP options, such as DL-Learner [51], to allow faster execution while maintaining high scoring accuracy, while still having the ability to perform as a simpler stand-alone scorer. The method works by training a model on a set of atomic class axioms scored by an algorithm, in this case [348]. This enables the model to predict the score of any new atomic (consisting of a single concept on each side)

candidate axiom. We experimented using multiple machine learning methods, and compared our work to the state-of-the-art [240] that aims to achieve the same goal.

This chapter is structured as follows: Sect. 5.2 provides some background about both axiom scoring and concept semantic similarity which are both prerequisites to training the models. As for Sect. 5.3 it lays out the method explaining how the axioms were extracted and scored, how the semantic measure we use was developed, and also how an axiom based vector space was modeled leading to the prediction of the scores. We detail our experiments including a comparison with the method presented in [240] in Sect. 5.4, then present the results while listing our observations and findings. We end the chapter with some notes and conclusions.

5.2 Background

5.2.1 Axiom Scoring

Axiom scoring can be done using statistical analysis, we detail in the Section 3.2 the possibilistic heuristic that does this using the theory of possibility described in Section 2.4.3. As a recap, possibility theory [106] is a mathematical theory of epistemic uncertainty. Its central notion is that of a possibility distribution which assigns to each elementary event a degree of possibility ranging from 0 (impossible, excluded) to 1 (completely possible, normal). A possibility distribution π induces a *possibility measure* Π, corresponding to the greatest of the possibilities associated to an event and the dual *necessity measure* N, equivalent to the impossibility of the negation of an event.

5.2.2 Ontological Semantic Similarity

Semantic similarity is a notion used to define a distance between terms or concepts based on meaning or semantics. It includes in its calculation only relations of the type **IS-A** [32]. It is often confused with semantic relatedness, for example a *train* and *train tracks* are functionally complementary, where as a *train* and an *airplane* are functionally similar. The latter is an instance of semantic similarity where the relatedness of both terms is based on the defining features they share where both are vehicles [32, 79]. Most semantic similarity measures that rely on a structured ontology, are based on path lengths between concepts as well as depth of concept nodes in an **IS-A** hierarchy. As for information-based measures they use information content (**IC**) of concept nodes derived from the ontology hierarchy structure and corpus statistics [4].

The similarity measure we utilize and extend in our work is the one created by Corby et al. [80, 79], under the subsection titled *Ontological Approximation*. The idea is to calculate the ontological distance between two concepts by using the subsumption path length. Following is the general definition:

$$\begin{gathered} \forall (t_1, t_2) \in H^2, \\ D_H(t_1, t_2) = min_t(l_H(\langle t_1, t \rangle) + l_H(\langle t_2, t \rangle)) \\ = min_t \left(\sum_{\{x \in <t_1, t>, x \neq t_1\}} 1/2^{d_H(x)} + \sum_{\{x \in <t_2, t>, x \neq t_2\}} 1/2^{d_H(x)} \right) \end{gathered} \tag{5.1}$$

Formula 5.1 translates to: for all type pairs t_1 and t_2 in an inheritance hierarchy H, the *ontological distance* between t_1 and t_2 in the inheritance hierarchy H is the minimum of the sum of the lengths of the subsumption paths between each of them and a common super type. And the length of the subsumption path between a type t_1 and its direct super type t is equal to $1/2^{d_H(t)}$ with $d_H(t)$ being the depth of t in

H.

The authors implemented this measure as part of a larger ontology-based search engine tool named `Corese` [78] [1] . This is the tool that has been used as means of extracting the semantic similarity between concepts in our method.

We propose an extension to this similarity to present similarities between axioms, this process is detailed in Sect. 5.3.2.

5.2.3 Instance Semantic Similarity

This similarity is used in the state-of-the-art method detailed in Section 3.4.1. The support vector clustering method the authors used requires a kernel function which was assumed to return the similarity between two axioms.

Similar to the ontological similarity, it is based on the semantics of axioms and not on their syntax. The measure S is defined with values in $[0, 1]$, satisfying the following desirable properties: for all axioms ϕ and ψ,

1. $0 \leq S(\phi, \psi) \leq 1$;

2. $S(\phi, \psi) = 1$ if and only if $\phi \equiv \psi$;

3. $S(\phi, \psi) = S(\psi, \phi)$.

In Section 3.4.1 we detail how the similarity works in theory. One problem is that instead of exactly computing the numerator, which would require to count the models and counter models of both axioms being compared, a not so convincing rough approximation of it is used. This approximation is obtained by replacing models and counter models with instances occurring in the RDF dataset that confirm or contradict the two axioms. This renders the similarity bound to and limited by instance data,

[1] https://project.inria.fr/corese/

which means it is prone to errors as well as unusable in case instance data is scarce or non existent.

In order to predict the ARI of subsumption axioms in this case, similarities between positive and negated subsumption axioms need to be computed. For the implementation of the similarity's computation in terms of SPARQL queries please refer to Section 3.4.1.

This is a slow process, and for every candidate axiom you would need to calculate the similarity for it and its negation. Since the queries used are counting queries similar to those used in the scoring heuristic this means it also suffers from the same limitations such as heavy computation cost and instance dependency.

5.3 Method

In an **OWL** ontology containing an inheritance hierarchy of concepts formed by the subsumption axiom *rdfs:subClassOf*, our aim is to predict an acceptability score for a candidate atomic class axiom by learning a set of previously scored axioms of the same type, the score used is the one detailed in Sect. 5.2.1. A model is built for each type of axiom to be predicted, this means that the method is repeated for the number of axiom types being dealt with. To measure the similarity between (candidate) axioms, we construct a similarity measure by extending the *ontological distance* discussed in Sect. 5.2.2, which is defined among concepts, not axioms. To this end, we consider the following necessary steps:

1. **Axiom extraction and scoring**: This step constitutes the creation of the set of scored axioms of a certain type to be learned. We either use a ready scored set such as we did for our comparison, or we score a new generated set.

2. **Semantic measure construction and assignment**: This step involves the

retrieval of concepts used in our set of axioms, and their ontological distance from the ontology. Followed by extending that similarity to those axioms. This was done by calculating a single value that represents the similarity between each pair of axioms, by applying a function such as *Average* to the ontological distances of concepts in those axioms.

3. **Axiom base vector space modeling**: This step focuses on using axiom similarity measures as weights/vector components, each axiom can be represented as a vector in an axiom based vector space.

4. **Score prediction**: This step is dedicated to training a Machine Learning model with the data set (vector space model in addition to the scores) and predicting the acceptability score of new candidate axioms.

We start by collecting the set of scored axioms we plan to train and test our method with. After that, we query the ontology to retrieve the ontological distances between all concepts. Then similarity measures between axioms will be derived from those distances and assigned as weights to represent the axioms as vectors in an axiom-based vector space. Machine learning models are used in the end to learn the set of scored axioms with their similarity weights and predict the acceptability score of a candidate axiom. We will consider subClassOf axioms for the comparison with [240] since that is the only type of axiom they can address, and their dataset which we use for said comparison is made of subClassOf axioms. We will also use disjointWith axioms for our experiment to show that we can apply our method to all atomic OWL class axiom types, as well as highlight that no leakage or bias is present from utilizing the subclass of hierarchy.

5.3.1 Axiom Extraction and Scoring

In this chapter, we consider two approaches. In the first one, we generate a number of atomic class candidate axioms randomly. The generated candidates are filtered for duplicates. We then make sure none of the candidates already exists in the ontology explicitly or implicitly, using the search engine and its reasoner, then we score them.

The other approach, which we prefer and use for controlled tests, is to query existing axioms. To make sure we have positive scores, we query the ontology for existing axioms and score them. The same can be said for negatively scored axioms, we can query the counter type and score it as if it were the first. For example, subClassOf and disjointWith are counter types so if $disjointWith(C_1C_2)$ has a positive score, $subClassOf(C_1C_2)$ will have a negative one. This is bound by how many axioms exist to be queried (after inferring implicit axioms).

Query 5.1 is used to extract existing axioms of both types and labeling them to be scored after with the heuristic. The search engine used is Corese [78] and the ontology is Dbpedia. After the extraction of the axioms, we select an equal amount of both classes.

```
0  SELECT ?class1 ?class2 ?label WHERE {
1  { ?class1 a owl:Class . ?class2 a owl:Class . ?class1
      rdfs:subClassOf ?class2
2  filter (!isBlank(?class1) && !isBlank(?class2))
3  filter (?class1 != ?class2)
4  bind(1.0 as ?label) }
5  Union{ ?class1 a owl:Class . ?class2 a owl:Class . ?class1
      owl:disjointWith ?class2
6  filter (!isBlank(?class1) && !isBlank(?class2))
```

```
7    filter (?class1 != ?class2)
8    bind(0.0 as ?label) }}
```

Query 5.1: Axiom extraction

The second step is to score, this is done by simply inputting the extracted axioms as a text file using the possibilistic heuristic [348], and receiving an output file containing the scores (ARI), it should be noted that the process is very slow thus the need for a method such as ours.

For disjointWith axioms, possibility only is considered since necessity is meaningless in the case of this axiom and incalculable. So the score is between 0 and 1.

5.3.2 Semantic Measure Construction and Assignment

To be able to assign similarity measures between axioms, we need to retrieve the ontological distances between all classes. Using *Corese* in which the ontological distance metric is implemented, this translates into a function added to the *SPARQL* query. Query 5.2 retrieves three columns, the first two contain the combination of all classes with the third containing the ontological distance denoted by similarity. Blank nodes are ignored.

```
0    select * (kg:similarity(?class1, ?class2) as ?similarity)
1    where {
2        ?class1 a owl:Class . ?class2 a owl:Class
3    filter (!isBlank(?class1) && !isBlank(?class2))
4    }
```

Query 5.2: Class ontological distance retrieval

Concepts	C_0	C_1	$\dots$	C_n
C_0	1	$S_{0,1}$	$\dots$	$S_{0,n}$
C_1	$S_{1,0}$	1	$\dots$	$S_{1,n}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
C_{n-1}	$S_{n-1,0}$	$S_{n-1,1}$	$\dots$	$S_{n-1,n}$
C_n	$S_{n,0}$	$S_{n,1}$	$\cdots$	1

Figure 5.1: Concept similarity matrix.

Scores	*Axioms*	A_0	A_1	$\dots$	A_m
$score_0$	A_0	1	$S_{0,1}$	$\dots$	$S_{0,m}$
$score_1$	A_1	$S_{1,0}$	1	$\dots$	$S_{1,m}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$score_{m-1}$	A_{m-1}	$S_{m-1,0}$	$S_{m-1,1}$	$\dots$	$S_{m-1,m}$
$score_m$	A_m	$S_{m,0}$	$S_{m,1}$	$\cdots$	1

Figure 5.2: Axiom similarity matrix with labels.

After retrieving the table of similarities, we pivot it resulting in a symmetric $n \times n$ matrix where the first column and the first row are the classes and the cells are the similarities between them with a diagonal of only *1's* since as we mentioned the similarity between a class C and itself is *1*. The shape of the concept similarity matrix is illustrated in Figure 5.1.

From this matrix, we can derive the similarity between axioms. The goal is to end up with a similar square symmetric matrix of the shape $m \times m$, m being the number of axioms, that has axioms instead of concepts as both first row and column, and the cells would be the similarities between a pair of axioms. Exactly as in the case of the concept similarity matrix, the diagonal of this matrix will contain only *1's* as the similarity between an axiom A and itself is *1*.

In order to construct this axiom similarity matrix M_A depicted in Figure 5.2 alongside the labels, we use Algorithm 3. The algorithm iterates over the set of labeled axioms T_A which we extracted in Section 5.3.1, comparing each axiom A_i to all other axioms A_j in T_A having j increment from i to length of $T_A - 1$ after each iteration of i. This is so we avoid redundant calculations. While comparing axioms A_i and A_j, we first deal with the concept on the left side of each axiom, so the left concept of A_i denoted by $A_i[L]$ and that of A_j denoted by $A_j[L]$. To retrieve S_L the similarity between those concepts from M_C, we search in the first row of the concept similarity matrix M_C for concept $A_i[L]$, and in the first column of M_C for concept $A_j[L]$. $M_C[A_i[L], A_j[L]]$, the intersection between the row where $A_i[L]$ was found, and the column where $A_j[L]$ was found represents the left side similarity between axioms A_i and A_j. The same process is repeated for the right side, and then the axiom similarity S is calculated as a function between S_L and S_R. In this work we applied two functions and they were *minimum* and *average*. After calculating S in each iteration of j it is added to M_A in the cell where the row is the index of A_i and column the index of A_j.

5.3.3 Axiom Base Vector Space Modeling

We define a vector-space model to represent axioms as vectors. The number of dimensions d of our vector space is equal to the number of axioms we have in the

Algorithm 3 Constructing the matrix of axiom similarities

Require: All concepts included in axioms in T_A be present in concept similarity matrix M_C

Ensure: $0 \leq S \leq 1$ ▷ S is the similarity between 2 axioms

$M_C \leftarrow Concept\ similarity\ matrix$

$T_A \leftarrow Set\ of\ labeled\ axioms$

$M_A \leftarrow Axiom\ similarity\ matrix\ to\ be\ filled$

for $i = 0 \rightarrow Length(T_A) - 1$ **do**

  for $j = i \rightarrow Length(T_A) - 1$ **do**

    $S_L \leftarrow M_C[A_i[L], A_j[L]]$

    $S_R \leftarrow M_C[A_i[R], A_j[R]]$

    $S^1 \leftarrow Avg(S_L, S_R)$ ▷ Experiments were done with (min, Avg)

    if Axiom Type is Disjointness or Equivalence **then**

      $S_L \leftarrow M_C[A_i[L], A_j[R]]$

      $S_R \leftarrow M_C[A_i[R], A_j[L]]$

      $S^2 \leftarrow Avg(S_L, S_R)$

    else

      $S^2 \leftarrow 0$

    end if

    $S \leftarrow Max(S^1, S^2)$

    $M_A.Add(A_i, A_j, S)$

  end for

end for

Note: For Symmetric axioms, we have to compare both ways which is what we do inside the if statement for Disjointness and Equivalence axioms.

labeled set T_A. Each axiom can be represented as a vector V in this d-dimensional space. Now that we have the similarities between axioms in our ontology, looking at the axiom-similarity matrix it would be intuitive to consider each row of the matrix as a vector V in a vector space where the dimensions d are the columns which are the axioms themselves having as weights the similarities S. Thus Algorithm 3 is utilized whenever an axiom is generated or a new candidate axiom is suggested and will encode said axiom into a vector V in this vector space.

5.3.4 Score Prediction

After constructing our vector space and representing axioms as vectors, we consider the set of scored axioms as a set of vectors. It is now possible to apply regression machine learning methods on the vector space representation of an axiom base. We used a range of methods throughout our experiments such as random forests, Support vector regressor, and neural networks. Trees and random forests were mostly used to check that there was no bias during the prediction, since they allow us to interpret our predictive model easily and visually analyse the decisions.

To avoid information leakage since the matrix is symmetric, considering the size of the matrix is $n \times n$, all models would be trained using an $m \times m$ sub-matrix of the axiom similarity matrix with the labels, and then tested on a $z \times m$ sub-matrix. This means an axiom vector V will have m dimensions (features) which are the ones used to train the model, regardless what the number of dimensions (features) in the full matrix is. In other words the symmetric part of the matrix equivalent to z (the columns of n outside the range of m) is discarded. The model's goal is to predict the score of a candidate axiom, which is represented as a vector V in our vector space having m dimensions (features), which are the axiom's similarities with the axioms of the same type used to train the model. The score is a number between -1 and

1 for subClassOf and equivalence, and between 0 and 1 for disjointWith. Here, 1 represents the highest acceptability. Any scorer/scoring algorithm can be used, we decided on [348] to be able to compare our results with [240].

5.4 Experiments and Results

For the experiments [2] the workstation that was used had the following hardware configuration:

- Dual CPUs: 2 × Intel(R) Xeon(R) CPU E5-2689 0 @ 2.60GHz base and 3.30 all core boost. With 8 cores and 16 threads per CPU for a total of 16 cores and 32 threads.
- A total of 32 GB of RAM memory with frequency 1600 MHZ distributed as 8 × 4 GB sticks with 4 sticks assigned to each CPU.
- 1 TB of NVME SSD storage with read and write speeds of up to 2000 MB per second.

5.4.1 Dataset Preparation

For our testing and experiments, and to comply with both the scorer and the experiment of [240], the ontology used is Dbpedia. This also allows us to show how our method would perform in a real-world case.

We used two datasets for our experiments, one for axiom type subClassOf and it is the one used in [240], and another for axiom type disjointWith which is a generated set of atomic disjointWith candidate axioms, as described in Sect. 5.3.1, scored using [348].

[2] All the data and code needed to replicate the experiments available at https://github.com/aliballout/axiom-score-prediction

For the axiom set used in [240], they have 722 subClassOf axioms and their negations, making it 1444 axioms. The negations are specific for that similarity and for the support vector clustering method that was applied in [240]. These negations serve no meaning if it is possible to predict the scores of the original 722 axioms. For this reason, and since we had the possibility of an axiom and its negation, we can use that to calculate the ARI using Equation 3.8. Then the dataset we work with will be the 722 axioms and their ARI score, since we will not be using the same machine learning method nor similarity for our proposed solution.

We did not include tests of equivalentClass axiom types since the process of creating the axiom similarity matrix of that axiom type is the same as disjointWith. They are both symmetrical axioms and equivalentClass axioms are basically a two way subClassOf axiom. For that reason we decided to include our experiment on the type disjointWith.

Similarity Method	Type	Set size	ASM	Candidate processing	NN	Random Forest	SVR	Modified SVC	Explained variance
Instance based	subClassOf	1444	649,440	1,799	0.35299	0.30707	**0.26721**	0.572	0.52652
Ontological based	subClassOf	722	**13.7275**	**0.01901**	**0.31442**	**0.30231**	0.33972	NA	**0.88859**
Ontological based	disjointWith	3868	129.9637	0.03359	0.23325	0.21754	0.23771	NA	0.73462

Table 5.1: Time cost in seconds, performance scores in RMSE per model, and the explained variance score of the best performing model per experiment.

For the set of axioms of type disjointWith, we load the Dbpedia OWL file into Corese. The OWL file contains no individual data which means faster processing in the search engine, corese reasoner was applied to it to deduce the maximum number of axioms. Positively scored disjointness axioms were obtained from the results of [264] [3] . Negatively scored disjointWith axioms were existing subClassOf axioms, explained in Section 5.3.1. In case the used ontology is already populated with explicit axioms, this would be done in an attempt to obtain a set of axioms that the scorer will not provide a score close to 0, i.e., a score that implies ignorance for lack of instance data. This allows us to learn a model that makes better predictions and surpasses the limitations a statistical scorer can face. But in our case this would result in a small set of 1120 scored axioms. One of the issues the authors of [240] faced is that the dataset they were working with was too small and they point out that this had prevented them from running certain experiments. For this case we combined both approaches expressed in Section 5.3.1. Randomly generating atomic candidates and checking that they do not already exist, we managed to score an additional 2748 axioms for a total of 3868. This will allow better testing to see how the method performs in real-world cases and on a different axiom type.

We will denote by T_A the axiom set being used. We will not be comparing the time needed to score the set of axioms since they both use the same scorer. Heuristic [348] takes a list of candidate axioms as input and provides as output a list of scored candidate axioms. It is a slow but precise heuristic that depends on instance data and counting queries.

After preparing the set of axioms, we have to produce the concept similarity matrix **(CSM)** shown in Figure 5.1, a process we detailed in Section 5.2.2. The

[3]https://bitbucket.org/RDFMiner/classdisjointnessaxioms/src/master/Results/ClassDisjointnessAxioms/

processing time for creating the *CSM* is dependent on the number of concepts. In our tested dataset, the number of unique concepts was 762 and creating its *CSM* took 2.0362 seconds.

The next step is constructing the axiom similarity matrix **(ASM)**, and encoding each of the axioms as vectors V in our vector space. This step is detailed in the second half of Section 5.3.2 and in Section 5.3.3. The algorithm applied to our axiom data set T_A is Algorithm 3, this is the same algorithm used to encode candidate axioms into the vector space, it is the equivalent of running queries shown in Section 3.4.1 to retrieve the similarity. In contrast the algorithm we use is not based on instance counting and is much faster in comparison. Table 5.1 details the time required to complete the axiom similarity matrix. As observed the time needed to complete the task of building the *ASM* significantly increases as the number of axioms processed increases, but we also know that it depends on the size of the *CSM* as well from the time needed to encode a single axiom into a vector. As the number of concepts increases the *CSM* being searched increases as it has the shape $n_{concepts} \times n_{concepts}$ and the number of cells n^2. We would note that the test scenario (3868 disjointWith axioms) presented in Table 5.1 is extreme, especially the case of Dbpedia, as the number of axioms needed for training a model does not need to be as large to achieve peak accuracy/results. It is possible with a dataset size of subClassOf experiment (722).

While constructing the ASM using the instance similarity method, we had to calculate the denominators of all axiom pairs as in Equation 3.11. We ran into an issue of lack of instance data, this means a 0 in the denominator which is obviously a problem. The authors address this by denoting any similarity between a pair of axiom where the denominator is 0 (lack of instances) with a 0 as well. This highlights the weakness of instance based similarity methods. We do not believe that simply saying

the similarity between a pair of axioms is 0 because they do not share individual data.

5.4.2 Training and Testing

After obtaining our datasets as described in the previous section, we applied different regression methods as mentioned in Section 5.3.4 to check how the similarity measure performs.

The methods we tested include, but are not limited to, Random Forests, Neural Networks and Support Vector Regression. Experiments were with both Average and Minimum functions to get the similarity S between axioms, from S_L and S_R (as explained in Section 5.3.2), we will denote by ASF the axiom similarity function here on out. All this means that the number of experiments performed was large so we will only report on some scenarios.

We used RMSE (root mean squared error) as the score, to be consistent with [240] during the comparison. We applied hyper-parameter tuning using grid search as well as five fold cross validation to infer better models. Table 5.1 shows the best results (in terms of RMSE) for each experiment using each model.

While replicating the experiment of [240], we decided to test other methods in addition to theirs. We will use the authors' best result for the comparison with the original model. We will be using the original 722 formulas for our proposed method, since our similarity does not require negated axioms (no need for the extra 722 negated axioms).

5.4.3 Results and Observations

Comparing ASM creation and candidate axiom encoding time costs in Table 5.1, for both our proposed method and the one in [240], our ontological based similarity seems almost instantaneous. For the instance based similarity it is a problem during

dataset preparation as it needs seven and a half days to prepare a data set of 722 axioms (and their negations), as well as candidate axiom encoding/processing where it takes half an hour to process every candidate axiom we want to predict a score to, whereas our ontological based method requires about fourteen seconds to process and prepare the exact same dataset, and less than 0.1 s to encode a new axiom, making it much more preferable with regards to time cost.

During the experiments we were unable to compare both methods in anything other than subClassOf axioms. This is because the method detailed in [240] can only handle subClassOf axioms making it very limited and constrained, whereas our proposed method can address all atomic class axiom types, all that is needed is training one model for each set. Considering the timing needed to prepare the training data (ASM), as seen in Table 5.1, this is easily doable and very efficient.

We observe in Table 5.1 that the time cost for creating the disjointWith (129 seconds for 3868 axioms) experiment's ASM was almost ten times that of the subClassOf (13 seconds for 722 axioms), even though the number of axioms is not ten times as much, only about five. This is normal since disjointWith axioms are symmetrical, and as shown in Algorithm 3 and explained in Section 5.3.2, the calculation is doubled for every axiom to check forwards and backwards the similarity between the pair of axioms.

It is very important to note that the instance-based similarity method performed better when used with a machine learning method different from the modified support vector clustering method used in [240]. It was able to achieve less than half the RMSE score of what was stated in [240]. This suggests that the modified support vector clustering method is unnecessarily complicated and not the most appropriate method to deal with this problem, which turns out to be relatively easy once a good similarity measure manages to transform it to a suitable representation. The ontological based

method outperformed the instance based method in Neural Network and Random forests models. For the same data set, the subClassOf set, the two methods seem to perform closely in terms of RMSE, while the ontological distance method seems to be achieving very good scores in the larger disjointWith set. This made us look more in-depth to what was happening. Turning our attention to the support vector regression model, we can see in Table 5.1 that for the same dataset (subClassOf) the instance based measure performed considerably better than the ontological based method, but was still outperformed in the larger (disjointWith) set as far as performance goes. So we investigated the reason behind this performance for this specific dataset in this model, and found the best explanation in the explained variance score. We found that for the instance based method, according to the explained variance score of **0.56252** for its best performing model the support vector regressor, this performance is specific to this dataset. This means there exists a bias in the training set and it is expected that any new candidate will suffer from a higher error rate than what is experienced in the training stage. This is explained by the fact that both similarity method and dataset were handcrafted and picked to work with a support vector clustering method modified to work as a regressor. On the other hand for the ontological based method the explained variance score throughout all our experiments ranged between 0.71 up to 0.88 meaning this method will produce better results when predicting scores for new candidate axioms.

All results shown in Table 5.1 were obtained using the Average Axiom Similarity Function (ASF), as it outperformed the Minimum ASF in all our runs.

5.5 Conclusion

We have proposed a method with the aim of learning predictors for the acceptability of an atomic candidate *OWL* axiom of any type. The method relies on a

semantic similarity measure derived from the ontological distance between concepts in a subsumption hierarchy. Extensive tests that covered multiple parameters and settings were carried out to investigate the effectiveness and potential of the method compared with the state-of-the-art.

The results obtained strongly support the effectiveness of the proposed method in predicting the scores of the considered OWL axiom types with a consistently low error rate. This allows us to confidently say that our proposed method can be used in combination with ILP or statistical methods, such as Dl-learner [51] or a Grammatical evolution approach such as [265], as a building block or extension/plugin to allow faster execution while maintaining accuracy.

We attribute the high accuracy and extremely good performance of our method to the close relation between the similarity measure and the model-theoretic semantics of axioms.

Chapter 6

PREDICTING THE ACCEPTABILITY OF ATOMIC CANDIDATE OWL CLASS AXIOMS

The task of evaluating the fitness of a candidate axiom against known facts or data is known as candidate axiom scoring. Being able to accurately score candidate axioms is a prerequisite for automatic schema or ontology induction, but can also be useful for ontology and/or knowledge graph validation. Accurate axiom scoring heuristics are often heavy to compute, which is a big problem if one wants to exploit them in iterative search methods like level-wise generate-and-test or evolutionary algorithms, where large numbers of candidate axioms need to be scored. In this chapter, we tackle the challenge of learning a predictive model as a surrogate to reasoning, that predicts the acceptability of candidate class axioms, that is fast to execute yet accurate enough to be used in such settings. For this purpose, we leverage a semantic similarity measure extracted from the subsumption hierarchy of an ontology. We prove that the method proposed in this chapter is able to learn the acceptability labels of candidate OWL class axioms with high accuracy and that it can do so for multiple types of OWL class axioms.

6.1 Introduction and Motivation

Ontologies, in the context of information science, are concerned with the categorization of entities, their associations, similarities, differences, and hierarchical relationships [161]. They serve as a critical tool in structuring and organizing the vast amount of digital information, providing a rigorous way to develop a general definition of a concept or entity. However, the process of ontology construction is

a labor-intensive and resource-demanding task, heavily reliant on the expertise of knowledge engineers or domain experts. This reliance extends throughout the ontology's lifecycle, as continuous development and expansion are necessary to accommodate evolving requirements. The field of ontology learning emerged as a response to this challenge, aiming to alleviate the bottleneck in knowledge acquisition by facilitating semi-automatic or automatic ontology construction and enrichment [290].

Methods addressing candidate OWL class axiom labeling adhere to the framework detailed in Chapter 4, where a set of labeled formulas (axioms) are used to construct a vector space based on the formulas' semantic similarity. The model is then trained using this set of formulas with the goal of predicting a label for new candidates. We have proven that this approach in ontology learning in Chapter 5, which focuses on predicting a fitness for candidate axioms in the form of a gradual score. Here, the focus is on the binary classification of a candidate axiom as accepted or rejected. In this chapter we present a novel approach that effectively addresses the limitations observed in previous techniques, such as the high computational cost during the training phase and sub-optimal accuracy scores. The solution we propose circumvents the reliance on potentially error-prone, instance-dependent statistics. Furthermore, it boasts the capability to predict labels for a variety of axiom types, including subsumption, disjointness, and equivalence.

The focus of this method is on the binary classification of a candidate axiom as accepted or rejected. We aim to create a method that can be used as a building-block or an extension/plug-in to other existing statistical analysis or ILP options such as DL-Learner [51] to allow faster execution while maintaining high scoring accuracy. The method would still have the ability to perform as a simpler stand-alone labeler.

We compare our proposed model to state-of-the-art description logic reasoners (DL-reasoners) using DBpedia as a real-world case. The aim of this comparison is

to see how our work holds up to similar tools in terms of accuracy and efficiency. We also argue that the ability to reject axioms, which is not available when using reasoners, is crucial for ontology learning.

This chapter is structured as follows: Section 6.2 provides some background about concept semantic similarity which is a prerequisite to train the models. As for Section 6.3 it lays out the method explaining how the axioms were extracted and labeled, how the semantic measure we use was retrieved, and also how an axiom based vector space was modeled leading to the binary classification of axioms. In Section 6.4, we detail our experiments then present the results while listing our observations and findings. We end the chapter with some notes and conclusions.

6.2 Concept Semantic Similarity: A Recap

Our primary goal is to create a model that can accurately *predict* the acceptability labels of axioms, *independent* of potentially error-prone instance data. This requires a training set of labeled axioms and a method to establish relationships between them, serving as features for the model.

Semantic similarity, a measure of distance between terms or concepts based on their meaning or semantics. In the context of ontological concepts, it incorporates the subsumption (IS-A, rdfs:subClassOf) relation [32]. Semantic similarity, based on a subsumption hierarchy, is a suitable candidate due to its close relation with other axiomatic relations between concepts/classes, making it useful for predicting new candidate class axioms of various types.

Notice that semantic similarity is often confused with semantic relatedness, for example a *train* and *train tracks* are functionally complementary, whereas a *train* and an *airplane* are functionally similar. The latter is an instance of semantic similarity where the relatedness of both terms is based on the defining features they share where

both are vehicles [32, 80]. Most semantic similarity measures that rely on a structured ontology are based on path lengths between concepts as well as depth of concept nodes in a subsumption hierarchy. As for information-based measures they use information content of concept nodes derived from the ontology hierarchy structure and corpus statistics [4].

The semantic similarity measure we use for this method is the one we detailed in Section 5.2.2 of the previous chapter.

6.3 Method

In an **OWL** ontology containing an inheritance hierarchy of concepts formed by the subsumption axiom rdfs:subClassOf, our aim is to predict if a candidate atomic axiom (consisting of a single named class on each side) is accepted or not. We do this by training a model on a set of previously labeled axioms of the same type (one of: subsumption, disjointness, equivalence) and their similarity weights. In the absence of a scored set our method creates one using explicit axioms available in the ontology. To measure the similarity between (candidate) axioms, we construct a similarity measure by extending the *ontological distance* discussed in Section 5.2.2, which is defined among classes, not axioms. This enables the model to predict the label of any new atomic candidate axiom of the same type. To this end, we consider the following steps:

1. **OWL ontology closure reasoning**: This step involves using DL-Reasoners the likes of HermiT [138] and Pellet [334] to infer all the axioms (binary relations between named classes) that can be inferred from the knowledge available in the ontology. In this chapter, we use it when testing the ability of our model to accept axioms that are not entailed by a reasoner, or to reject ones that are entailed by a reasoner.

2. **Axiom extraction and labeling**: This step constitutes the creation of the set of accepted and rejected labeled axioms of a certain type to be learned. One approach can be querying existing axioms and labeling them as accepted/rejected. Another is to use a scorer to label a set of generated candidate axioms to learn.

3. **Semantic measure retrieval and assignment**: This step involves the retrieval of concepts used in our set of axioms, and their ontological distance from the ontology, followed by extending that similarity to those axioms. This was done by calculating a single value that represents the similarity between each pair of axioms, by applying a function such as *Average* to the ontological distances of concepts in those axioms.

4. **Axiom base vector space modeling**: This step focuses on using axiom similarity measures as weights, each axiom can be represented as a vector in an axiom based vector space.

5. **Binary Classification**: This step is dedicated to training a Machine Learning model with the data set (vector space model in addition to the extracted labels) and predicting if new candidate axioms are accepted or rejected.

6.3.1 Owl Ontology Closure Reasoning

The first step in our method is optional when the ontology used is rich in explicit axioms. It consists of inferring all axioms that are entailed by the knowledge available in a target OWL ontology. This is useful, and vital for our experiment, for two reasons. The first is providing a larger pool of axioms to use for training the model. In our work this is helpful since a larger sample size means more accurate experimental results. This way we can compare the performance with DL-reasoners in terms of

accuracy using axioms that reasoners can entail. The second reason would be the ability to filter new labeled/scored axioms. These axioms would be checked against our ontology closure. We can then use the newly scored axioms that are not entailed by the reasoner to evaluate the model's performance relative to the reasoner's. This would show how well the model can handle more challenging cases that the reasoners cannot.

6.3.2 Axiom Extraction and Labeling

In this chapter, we employed two approaches to build a set of labeled axioms. The first approach, is to classify axioms using an existing scoring method or use previously scored axioms from other literature. The second way, which is faster, we call the type and counter-type technique.

Scoring Method

We used multiple sets of scored axioms for the ontology DBpedia, one set [1] of axioms generated and scored by Nguyen *et al.* [264] using the possibilistic heuristic [348]. The rest of the sets we scored ourselves using the same heuristic. The scorer though accurate, is extremely slow limiting the size of the sets scored. The scorer currently supports DBpedia and is publicly available [2] . We utilized the version detailed in the work of Felin *et al.* [123]. These scored data sets are what we use for our evaluation and comparison with reasoners in the case of DBpedia. The possibilistic heuristic scores are considered our ground truth and baseline. For a deep analysis of the scoring method please refer to Section 3.2. For experiments using other ontologies we employed the second approach.

[1] https://bitbucket.org/RDFMiner/classdisjointnessaxioms/src/master/Results/ClassDisjointnessAxioms/

[2] https://github.com/RemiFELIN/RDFMining

Type and Counter Type Technique

In this technique, we query an ontology for existing axioms of a certain type, thereby obtaining a set of axioms labeled as *accepted*. We then query axioms of the counter type. For instance, subClassOf and disjointWith can be considered counter types to each other. Axioms of the counter type are assigned the label *rejected*. Since existing disjointWith axioms can be considered false or *rejected* subClassOf, we can construct a sample containing both labels to train a model. We consider disjointWith to be a counter-type for subClassOf and equivalentClass. While subClassOf and equivalentClass are counter-types of disjointWith. However, before querying an ontology for existing axioms, it is advisable to apply a reasoner to obtain the closure of the ontology. The reasoner can be any suitable choice, such as HermiT or Pellet, and not necessarily the reasoner included in the engine used. By applying a reasoner to obtain the closure of an OWL ontology, we ensure that we obtain a complete and consistent set of labeled axioms for our model.

The axiom type designated as *accepted* corresponds to the type that the model will be labeling. For instance, if we assign the *accepted* label to disjointWith axioms, it implies that disjointWith is the axiom type that the model is designed to address. While this may initially appear as a limitation, the efficiency of the process mitigates this concern. As demonstrated in Table 6.2, which details the time consumption, the speed of dataset preparation and model training is sufficiently rapid. This allows us to learn a model for each type of axiom within a relatively short time frame.

We employ Query 6.1 to extract and label existing axioms of both the type and counter-type. Following the extraction, we ensure an equal representation of both type and counter-type axioms. The SPARQL endpoint utilized in this process is Corese [78]. We have selected DBpedia as our test case to illustrate a real-world

application. While other ontologies could be employed, DBpedia offers several advantages. It has been widely used in related work and aligns well with the scoring heuristic we employ for evaluating our candidate axioms.

```
SELECT ?class1 ?class2 ?label
   WHERE {
          {
              ?class1 a owl:Class . ?class2 a owl:Class . ?class1
                 rdfs:subClassOf ?class2
              filter (!isBlank(?class1) && !isBlank(?class2))
              filter (?class1 != ?class2)
              bind(1.0 as ?label)
          }
          Union{
              ?class1 a owl:Class .
              ?class2 a owl:Class .
              ?class1 owl:disjointWith ?class2
              filter (!isBlank(?class1) && !isBlank(?class2))
              filter (?class1 != ?class2)
              bind(0.0 as ?label)
          }
      }
```

Query 6.1: Axiom extraction

6.3.3 Semantic Measure Retrieval and Assignment

To be able to assign similarity measures between axioms, we need to retrieve the ontological distances between all classes and construct the concept similarity matrix **(CSM)**. Using *Corese* in which the ontological distance metric is implemented, this translates into a function added to the *SPARQL* query. Query 6.2 retrieves three columns, the first two contain the combination of all classes with the third containing the ontological distance denoted by similarity. Blank nodes are ignored.

```
0select * (kg:similarity(?class1, ?class2) as ?similarity)
1    where{
2        ?class1 a owl:Class .
3        ?class2 a owl:Class
4        filter (!isBlank(?class1) && !isBlank(?class2))
5    }
```

Query 6.2: Class ontological distance retrieval

After retrieving the table of similarities we povit it resulting in a symmetric $n \times n$ matrix where the first column and the first row are the classes and the cells are the similarities between them with a diagonal of only *1's* since as we mentioned the similarity between a class C and itself is *1*.

6.3.4 Axiom Base Vector Space Modeling

From the CSM, we can derive the axiom similarity matrix **(ASM)** with labels illustrated in Figure 6.1, as explained in Chapter 5 which also highlights that the function used to calculate the similarity S between a pair of axioms can be either *Average* or *Minimum*.

Labels	*Axioms*	A_0	A_1	$\ldots$	A_m
Accepted	A_0	1	$S_{0,1}$	$\ldots$	$S_{0,m}$
Rejected	A_1	$S_{1,0}$	1	$\ldots$	$S_{1,m}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Rejected	A_{m-1}	$S_{m-1,0}$	$S_{m-1,1}$	$\ldots$	$S_{m-1,m}$
Accepted	A_m	$S_{m,0}$	$S_{m,1}$	$\cdots$	1

Figure 6.1: Axiom similarity matrix with labels

We define then a vector-space model to represent axioms as vectors. Indeed, we represent each axiom as a vector whose elements are the kernel values representing the similarity between the considered axiom and the other axioms present in the labeled dataset T_A. We have been inspired by the Kernel trick in Support Vector Machine (SVM) which is used to deal with nonlinear classification [257].

The number of dimensions m of our vector space corresponds to the number of axioms we have in T_A. Each axiom is then represented as a vector V in this m-dimensional space, and corresponds to a row in the axiom similarity matrix as illustrated in Figure 6.1. The only difference between this matrix and Figure 5.2 is that instead of continuous scores we assign labels for each vector.

The axiom similarity matrix is updated whenever an axiom is generated or a new candidate axiom is suggested.

6.3.5 *Binary Classification*

It is now possible to apply classification machine learning methods using our labeled dataset of axioms represented as vectors. We used a range of methods throughout our experiments, including but not limited to trees, random forests, kNN, Support vector classifier, gradient boosting, and neural networks. Trees and random forests were used to check that there was no bias during the classification. The reason for this choice is that such methods allow us to interpret our predictive model easily and visually analyse the decisions. kNN instead was used to test the effect of the similarity measure as a distance (the metric used was Manhattan, due to the large number of dimensions and the weight was distances). To avoid information leakage since the matrix is symmetric, considering the size of the matrix is $m \times m$, all predicting models were trained using an $m' \times m'$ sub-matrix, with $m' < m$, of the axiom similarity matrix with the labels, and then tested on the $(m - m')$ remaining axioms. Our goal is then to predict the label of those axioms which have not been used during the training period. The labels are binary, where 1 represents the label *Accepted* and 0 represents the label *Rejected*.

6.4 Experiments and Results

For the following experiments, the workstation that was used had the following hardware configuration:

- Dual CPUs: 2 × Intel(R) Xeon(R) CPU E5-2689 0 @ 2.60GHz base and 3.30 all core boost. With 8 cores and 16 threads per CPU for a total of 16 cores and 32 threads.
- A total of 32 GB of RAM memory with frequency 1600 MHz distributed as 8 × 4 GB sticks with 4 sticks assigned to each CPU.

- 1 TB of NVME SSD storage with read and write speeds of up to 2000 MB per second.
- Code and ontologies used available in our repository [3] .

6.4.1 Dataset Preparation

As mentioned in Sect. 6.3.5, we have experimented using a range of classification methods, ontologies and types of axioms. We will focus on disjointness axioms for examples given to highlight that no leakage or bias is present from using the subClassOf hierarchy. The first part of our approach was building our datasets as follows:

Ontology Selection

We sought out different ontologies of different sizes and domains to use in our experiments. Some statistics about each ontology used are shown in Table 6.1. Following is a brief description the ontologies that were selected, as described in Section 3.1.1.

- NTNames [4] .
- Pizza, [5] .
- MatOnto, [6] .
- DBpedia, [7] .

[3] https://github.com/ali-ballout/axiom-acceptability

[4] https://semanticbible.com/index.html

[5] https://protege.stanford.edu/ontologies/pizza/

[6] https://github.com/inovexcorp/MatOnto-Ontologies

[7] http://downloads.dbpedia.org/

Table 6.1: Ontology statistics. Reasoner used: Corese built in reasoner: ^{C}, Pellet: ^{P}, Hermit: ^{H}.

Ontology	Classes	subClassOf	disjointWith	equivalentClass
NTNames^{C}	47	278	10	50
Pizza^{C}	101	651	5	101
MatOnto^{C}	847	853	158	9
DBpedia^{HPC}	866	7334	70669	268

Table 6.2: Time cost per step for investigated ontology in seconds for subClassOf.

Ontology	CSM	Axioms processed	ASM	Single Axiom vector encoding	Model training
NTNames	0.02	556	22	0.03	0.2
Pizza	0.04	1302	105	0.05	0.4
MatOnto	2.84	1706	212	0.09	0.6
DBpedia	2.17	8600	2497	0.08	3.9

Axiom Extraction and Labeling

Each ontology was loaded into Corese. Since in some of the ontologies the explicit number of axioms was not big enough for any meaningful experiment, we applied different reasoning methods to obtain the maximum number of deduced axioms. For all ontologies, we used Corese reasoner, which is a rule-based engine that can handle RDF(S) and OWL 2 RL profiles. For DBpedia specifically, we additionaly used

Hermit and Pellet reasoners to compute the closure, a step which took two hours for each run using Protegé. [8] Additionally, we used the disjointness axioms generated and scored by Nguyen *et al.* [264] for DBpedia, as explained in Section 6.3.2. After that, we extracted a balanced set of *Accepted* and *Rejected* axioms from each ontology using SPARQL queries and the technique of type and counter type explained in Section 6.3.2. For instance, for subsumption in DBpedia, we selected 4300 *Accepted* and 4300 disjointWith axioms representing *Rejected* subClassOf axioms, resulting in a total of 8600 axioms for T_A, which is our axiom dataset for this scenario.

Explicit axiom extraction by querying an ontology using a SPARQL endpoint needs minimal time, as for generating a set of labeled axioms as in [264] the time cost would depend on the scorer and the method used to generate candidate axioms.

Concept Similarity Matrix

After preparing the set of axioms, we have to produce the concept similarity matrix **(CSM)**, as explained in Section 6.3.3. The processing time for creating the **CSM** depends on the number of concepts. In our tested dataset, MatOnto had 847 concepts and creating its **CSM** took 2.84 seconds. In Table 6.2 we present the timings for the worst-case scenario, which consists of applying our method to the most dense axiom type that is subsumption for comparison reasons. Other axiom types are naturally much faster due to their lesser population.

Axiom Similarity Matrix and Vector Space

The next step is constructing the axiom similarity matrix **(ASM)** as shown in Figure 6.1 , and encoding each of the axioms as vectors V in our vector space.

Table 6.2 details the time required to complete these operations, all the values

[8] https://protege.stanford.edu/

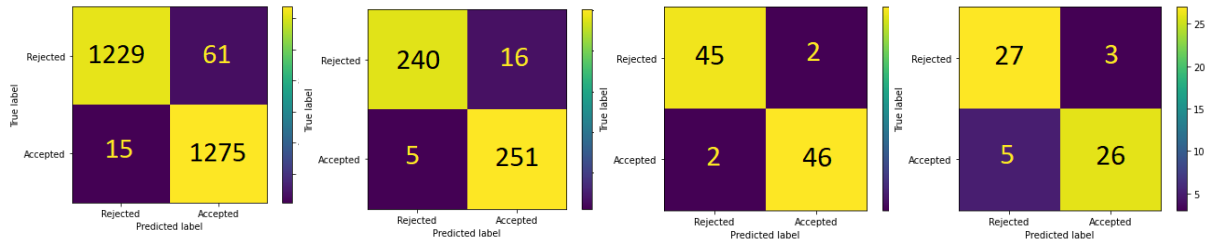
presented are the mean average of *five* consecutive runs. As observed the time needed to complete the task of building the *ASM* significantly increases as the number of axioms processed increases, but we also know that it depends on the size of the *CSM* as well from the time needed to encode a single axiom into a vector. As the number of concept increases the *CSM* being searched increases in size as it has the shape $n_{concepts} \times n_{concepts}$ and the number of cells n^2. We would like to note that the test scenario presented in Table 6.2 is extreme, especially the case of DBpedia, as the number of axioms needed for training a model does not need to be as large to achieve peak accuracy/results.

6.4.2 Training and Testing

Using the dataset obtained as described in the previous section, the classifiers we tested were Decision Trees (DT), Random Forests (RF), kNN, support vector classifier (SVC), Neural Networks (NN) and Gradient boosting (GB). Experiments were performed over every class axiom type, where the axiom population was enough, for every ontology in our dataset some results are shown in Figure 6.1. We also experimented with both Average and Minimum functions to get the similarity between axioms S. The results of such experimentation can be found in Table 6.3 where models were trained using 813 axioms split into 406 *Rejected* and 407 *Accepted* axioms, and tested on 349 axioms split into 175 *Rejected* and 174 *Accepted*.

We would also like to note that the model training times mentioned in Table 6.2, are the average for the above mentioned methods except for the extreme case of DBpedia subsumption using gradient Boosting, where it may take up to 136 *seconds* to train the model on a set of $6,000$ axioms.

In order to prove that the proposed kernel-based representation of axioms is advantageous for addressing the task of predicting the acceptability of candidate axioms,



(a) subClassOf DBpedia (b) subClassOf MatOnto (c) disjointWith MatOnto (d) equivalentClass Pizza

Figure 6.1: Performance by axiom type for investigated ontologies

Table 6.3: F1 scores using DBpedia owl:disjointWith, average and minimum as similarity functions.

Function	GB	kNN	NN	RF	SVC	DT
F1 (AVG)	**0.988**	0.976	0.953	0.985	0.976	0.982
F1 (MIN)	**0.977**	0.974	0.962	0.976	0.962	0.968

we performed an additional experiment. For this experiment we used a simple naive kNN method using our axiom similarity measure *directly* to check the n nearest axioms to a candidate axiom and then assigning that candidate axiom the label that occurs most in the n neighbors. To make things fair, we compare the performance of this naive approach to our proposed kernel-based vector-space representation of axioms, using kNN. We perform the comparison using DBpedia's subClassOf axioms as a dataset. We used as a training set 200 axioms, and for the test set 4000 axioms; both sets are balanced in terms of labels.

Table 6.4 presents the F1 scores of each of the methods as a function of the number n of neighbors. From the results we can see that the naive method performs best when

the number of neighbors is 1, this is because the method does not learn anything and simply retrieves the closest axioms based on the similarity measure. Our proposed method on the other hand maintains performance since it uses the similarity matrix as a kernel to map the neighbors to a plane of dimensions equal to the number of axioms used in training. Our method outperforms the naive one by a significant margin every time. This corroborates the hypothesis that the proposed kernel-based vector-space representation of axioms is capable of capturing useful features of the semantics of candidate axioms, this offering a clear advantage over the simple and direct use of the axiom similarity matrix.

Table 6.4: F1 scores for the naive nearest neighbor method and our proposed method using K-nearest neighbor as the machine learning model, as a function of number n of neighbors.

Model	n_1	n_3	n_{15}
Naive	0.59	0.49	0.33
kNN	0.86	0.87	0.86

Table 6.5: Performance of our proposed model when trained using a scored training set and a training set extracted from the closure of DBpedia as a product of HermiT, Pellet and Corese, compared to reasoners in accepting or rejecting axioms of different types.

		Model scored training set			Model closure training set			Reasoners		
Axiom type	Label	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
subClassOf	Accepted	**1.00**	**0.91**	**0.95**	0.66	1.00	0.79	**1.00**	**1.00**	**1.00**
	Rejected	**0.90**	**1.00**	**0.95**	1.00	0.48	0.65	**1.00**	**1.00**	**1.00**
disjointWith	Accepted	**0.97**	**0.94**	**0.96**	0.27	0.99	0.43	0.00	0.00	0.00
	Rejected	**0.83**	**0.92**	**0.87**	0.99	0.20	0.34	0.23	1.00	0.37

6.4.3 Proof of Concept

In order to prove that the proposed kernel-based representation of axioms is advantageous for addressing the task of predicting the acceptability of candidate axioms, we performed an additional experiment. For this experiment we used a simple naive kNN method using our axiom similarity measure *directly* to check the n nearest axioms to a candidate axiom and then assigning that candidate axiom the label that occurs most in the n neighbors. To make things fair, we compare the performance of this naive approach to our proposed kernel-based vector-space representation of axioms, using kNN. We perform the comparison using DBpedia's subClassOf axioms as a dataset. We used as a training set 200 axioms, and for the test set 4000 axioms; both sets are balanced in terms of labels.

Table 6.4 presents the F1 scores of each of the methods as a function of the number n of neighbors. From the results we can see that the naive method performs best when the number of neighbors is 1, this is because the method does not learn anything and simply retrieves the closest axioms based on the similarity measure. Our proposed method on the other hand maintains performance since it uses the similarity matrix as a kernel to map the neighbors to a plane of dimensions equal to the number of axioms used in training. Our method outperforms the naive one by a significant margin every time. This corroborates the hypothesis that the proposed kernel-based vector-space representation of axioms is capable of capturing useful features of the semantics of candidate axioms, this offering a clear advantage over the simple and direct use of the axiom similarity matrix.

6.4.4 Comparing With Reasoners

In this experiment, we compare the performance of our model with HermiT, Pellet, and Corese in accepting or rejecting candidate OWL class axioms. For this experiment, we prepare a dataset generated and scored by the possibilistic heuristic using DBpedia ontology which is widely used and represents our real-world scenario. The generation and scoring required seven days of processing time. The dataset contains: 811 accepted subClassOf axioms and 745 rejected ones, as well as 1276 accepted disjointWith axioms and 823 rejected ones. We split the dataset into training and test sets. The split was as follows: the subClassOf training set contains 399 accepted and 398 rejected, while the test set contains 412 accepted and 347 rejected. As for disjointWith, the training set contains 524 accepted and 596 rejected axioms, while the test set contains 752 accepted and 227 rejected.

We evaluated our model and the reasoners on test sets, employing two training scenarios for our model. In the first scenario, we trained our model using a set extracted from the closure of DBpedia, produced by the type counter type method. In the second scenario, we used a scored training set independent of the reasoners. We evaluated the reasoners by checking if they could entail the candidate axioms from DBpedia. If a candidate is entailed then it is labeled as accepted otherwise it is labeled rejected. We assessed precision, recall, and F1-score with our ground truth being the labels produced by the possibilistic scorer. The results are presented in Table 6.5.

We noticed that the results of our model were worse when training it using the closure, and almost perfect when using the scored training set. Therefore, we hypothesized that the labels of the training set from the closure (reasoners product) could have wrong labels, while the labels from the scorer were more accurate. To test

this hypothesis, we ran the scored training set through the reasoners to see how they would label it. Indeed, the reasoners were entailing axioms labeled as rejected by the scorer, which appear to have a great impact on how the model labeled the test set. A sample of these axioms can be seen in Table 6.6.

Table 6.6: subClassOf axioms rejected by the scorer and entailed by the reasoners

```
SubClassOf(Tax Genre)

SubClassOf(Organ Aircraft)

SubClassOf(Guitar Locomotive)

SubClassOf(Treadmill ArchitecturalStructure)

SubClassOf(Quote Work)

SubClassOf(Instrument MilitaryVehicle)

SubClassOf(Guitar Aircraft)
```

6.4.5 Key Findings and Observations

In all the experiments, the results for different models trained with the same set were noticeably close. Also, all models consistently achieved very high scores averaging $90-95\%$ accuracy. Below we state some noteworthy observations we made during our experiments:

Axiom Similarity Function

Throughout our experiments, we have observed a consistent trend where most *ASMs* that were constructed using the *Average* function to obtain S gave better results than

those that were built using the *Minimum* function. This can be observed in Table 6.3.

The models are scored based on their performance on the test set (i.e., when classifying axiom candidates never seen before).

We note that Neural Networks were the only model to break this trend. It performed better when the *Minimum* function was used as seen in Table 6.3. However, despite that, it had the worst performance out of all the models presented using both functions, even though different configurations were tested [9] .

Classifier Of Choice

Tree-based models consistently achieved the highest scores. This fact is very evident in Table 6.3, where the leading models when using the better similarity function *Average* were, in order: *Gradient Boosting*, *Random Forest*, and *Tree*. For this reason, and since all models score close to each other with an accuracy greater than 95%, we find *Random Forests* combined with the *Average* function for S to be the classifier of choice when creating the model. We chose *Random Forest* over *Gradient Boost* since the time for training *Random Forest* models is faster while giving up at most 0.1% accuracy.

Model Perfomance Dealing With Variables

We chose the confusion matrix metric to summarize the results while showing the *support*, and how the method performs with different sizes of datasets and ontologies as a whole. Figure 6.1 presents the performance of the method on three different ontologies with different sizes as well as different types of axioms and populations. It is very clear how well the method performs across all those variables. The figure shows results only from the test set, i.e., when the model predicts labels for candidate

[9] Can be found in `keratest.py` in the repository

axioms it has never seen before. This brings us to the conclusion that the method is performing as expected with F1 score between 95% to 99%.

6.5 Conclusion

We introduced a method for predicting the acceptability of various atomic candidate *OWL* class axioms, leveraging a semantic similarity measure derived from ontological distance in a subsumption hierarchy. Comprehensive testing across various ontologies and parameters affirmed the method's robustness.

The method consistently achieved high accuracy in predicting labels for all considered OWL axiom types, validating its effectiveness. This underscores its potential for integration with ILP or statistical methods, such as Dl-learner [51] or a Grammatical evolution approach [265], to enhance execution speed without compromising accuracy.

Chapter 7

SCALABLE PREDICTION OF ATOMIC CANDIDATE OWL CLASS AXIOMS USING A VECTOR-SPACE DIMENSION REDUCED APPROACH

Scoring candidate axioms or assessing their acceptability against known evidence is essential for automated schema induction and can also be valuable for knowledge graph validation. However, traditional methods for accurately scoring candidate axioms are often computationally and storage expensive, making them impractical for use with large knowledge graphs. In this chapter, we propose a scalable method to predict the scores of atomic candidate OWL class axioms of different types. The method relies on a semantic similarity measure derived from the ontological distance between concepts in a subsumption hierarchy, as well as feature ranking and selection for vector-space dimension reduction. We train a machine learning model using our reduced vector-space, encode new candidates as vectors, and predict their scores. Extensive tests that cover a range of ontologies of various sizes and multiple parameters and settings are carried out to investigate the effectiveness and scalability of the method.

7.1 Introduction

Machine learning techniques that tackle the task of candidate axiom scoring face a scalability problem when dealing with large ontologies and the number of facts and entities they include [269]. This is because the process can be intensive in terms of storage and computation, particularly for large and complex datasets. For ontology learning, this would require techniques to step away from instance data when possible and rely more on what has already been established in an ontology's structure. As

a result, scalable techniques and models with the ability of addressing ontologies of different sizes while maintaining satisfactory performance without incurring excessive computational and storage costs become a necessity.

In this chapter, we present the issue of dealing with large ontologies and its effect in terms of storage and computation when attempting to score candidate class axioms. In addition, we propose an approach that scores atomic candidate OWL class axioms of different types for ontologies of different sizes. We do so by utilizing the ontological semantic similarity, described in Section 5.2.2, between concepts and extending it to axioms. We incorporate feature selection techniques and apply them to our dataset to pick the most impactful axioms to act as our dimensions in an axiom-based vector space. We encode candidate axioms into this vector space without the need for any instance data. We experiment using DBpedia, Gene ontology (GO), and Cell ontology (CL) to test the scalability of the approach, as well as the effect of feature selection on performance, storage cost and computation time.

This chapter is structured as follows: in Section 7.2 we give an overview of some related work; Section 7.3 provides a recap on ontological axiom semantic similarity, the possibilistic axiom scorer and feature selection. As for Section 7.4, it lays out the method explaining how the axioms are extracted and scored, how we build the semantic measure, and also how we model an axiom based vector-space leading to the prediction of a candidate axiom's score. We detail our experiments in Section 7.5 then present and analyze the results in Section 7.6. We end the chapter with some notes and conclusions.

7.2 Related Work

Since the current chapter aims at developing a novel approach to scalable prediction of candidate class axiom scores, it is relevant to provide an overview of previous

research on the topic of predicting the score of candidate OWL class axioms. One such research is the work desribed in Section 3.5. It uses methods such as principle component analysis (PCA) to map axioms into a lower-dimension space. This form of dimensionality reduction, which is unrelated to ours in method or goal [1], combined with instance-based similarity measures, resulted in less than satisfactory performance. Indeed, the authors expected to see a clear separation between accepted and rejected axioms, which would have made it possible for unsupervised methods to perform the task of labeling candidate axioms, but their results did not support this hypothesis.

However, an instance-based similarity measure fully relies on an ontology's instance data, and any lack of such data results in ignorance, while an excessive amount of data overwhelms the method. Some follow a different path, such as the works described in Section 3.6, which take the embedding approach utilizing an ontology's class subClassOf hierarchy, also known as the *is-a* hierarchy, to predict subsumers.

These methods and others that use embeddings, like the ones we mentioned before, only address subsumption. They also prove to be computationally complex. For example, the work done by Chen et al. as it follows a breadth-first algorithm when embedding a subsumption relation [69]. This algorithm keeps extracting the subsumers of each of the classes till reaching a leaf, or the superclasses till reaching the root in order to generate a sentence. The authors mention limiting the length of their sentences for the evaluation, highlighting a trade-off between having complete sentence context and redundancy.

We have provided an answer for this challenge in both Chapter 5 and Chapter 6, where we extend the scope to include disjointWith class axioms as well. However,

[1] PCA is a technique used to map data into lower dimensional planes, where as feature selection ranks your dimensions in terms of how useful they are to predict the target value and allows you to drop the low ranking dimensions.

we only experiment using DBpedia, which includes around 760 concepts, positioning it as a smaller ontology when compared with ontologies with tens of thousands of concepts such as GO and CL.

The work presented in this chapter specifically aims to address the shortcomings of the previous models with respect to computational complexity, storage cost, and scalability. We do this by leveraging feature selection techniques like ones described in Section 2.5.3 and performing query optimizations. The results of this method will be compared to those of Chapter 5.

7.3 Background

7.3.1 Ontological Axiom Semantic Similarity and scoring: A Recap

We calculate the axiom semantic similarity as we did in Chapter 5 by performing the following steps:

1. Extract the distances between all concepts in the ontology and store them in a concept similarity matrix.

2. Compare each axiom with all other axioms in the dataset.

3. When comparing two axioms, retrieve from the concept similarity matrix the similarity/distance between the concepts on the left side of the axiom.

4. Repeat the previous step for the right side.

5. In case of symmetric axiom types (disjointness/equivalence) repeat the comparison between the left concept from the first axiom and the right concept of the second axiom, and then between the right concept from the first axiom and the left concept from the second one. Keep the higher values between both comparisons.

6. Take the average of the two values that you have as a result of the previous step.

7. Store that value in an axiom similarity matrix.

As for the scores of the axioms, we use the possibilistic heuristic as well as another method described in Section 5.3.1. For more detail on the heuristic refer to Section 3.2. This process is used for the construction of an axiom-based vector-space, where each axiom or candidate can be represented by a vector of its similarity to all other axioms.

7.3.2 Feature Ranking and Selection: A Recap

In machine learning, feature selection is referred to as the process of obtaining a subset from an original feature set according to a certain feature selection criterion, which selects the relevant features of the dataset. It plays a role in compressing the data processing scale, where the redundant and irrelevant features are removed [54]. It is particularly useful in the case of high-dimensional datasets. It does not involve dimension aggregation, nor attempts to map higher-dimensional spaces to lower ones as done by Malchiodi et al. [239].

According to their relationship with learning methods, feature selection methods can be classified into filter, wrapper, and embedded models. In our work we use the filter model, which has a lesser computational cost than the others [54]. A good feature selection method should have high learning accuracy but less computational overhead (time complexity and space complexity).

Filter feature selection methods typically utilize evaluation criteria to increase the correlation between the feature and the class label and decrease the correlation among features. In addition, the correlation among features is often replaced by either redundancy or diversity (distance). These measures of relevance, redundancy,

and diversity may be identical or distinct. Filter methods involve selecting features based on their individual statistical properties. This can be done using techniques such as correlation analysis or mutual information, which measure the strength of the relationship between a given feature and the target variable. At the end, features are ranked based on their effect on the class label and then a percentage or number of the can be kept while the rest is discarded.

For example, the mutual information gain (also known as mutual information or MI) between two variables x and y can be calculated as follows:

$$MI(x, y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \tag{7.1}$$

where $p(x)$ and $p(y)$ are the marginal probability distributions of x and y, respectively, and $p(x, y)$ is the joint probability distribution of x and y. Mutual information measures the amount of mutual dependence between the two variables, with higher values indicating a stronger relationship. It is often used in feature selection to identify the most relevant features for a given task.

7.4 Method

Our objective is to develop a scalable method to predict a score for atomic candidate OWL class axioms by learning from a set of previously scored axioms of the same type. To this aim, we exploit the hierarchy of concepts formed by the subsumption rdfs:SubClassOf axioms, combined with feature selection. A separate model is required for each type of axiom addressed. Following are the steps of our method:

1. **Axiom extraction and scoring**: This step describes the creation of the set of scored axioms of a certain type to be learned. One approach would be to use a scorer to label a set of generated candidate axioms to learn as done

in the methods we have mentioned in Chapter 5. Another approach can be to query existing axioms and label them as accepted, then generating some random axioms and checking that they are not explicitly available or entailed in the ontology, and labeling them as rejected as done other methods [69].

2. **Axiom similarity calculation**: This step details how we extract the concepts used in our set of axioms and retrieve the ontological distances pertaining to these concepts only. Unlike the method proposed in Chapter 5, we only query concepts present in our set of axioms instead of all concepts present in the ontology. We then calculate the axiom similarity measure using the algorithm detailed in Chapter 5 and briefly explained in Section 7.3.1, but we enhance performance by leveraging the power of multiprocessing.

3. **Axiom-based vector-space modeling**: This step focuses on using the axiom similarity measures as weights; each axiom can be represented as a vector in an axiom-based vector-space.

4. **Vector-space dimensionality reduction**: This step consists of applying feature selection on the axiom similarity matrix to reduce the number of axioms being used as dimensions to a certain pre-defined number. This results in reduced computation and processing when encoding axioms into the vector-space as well as smaller concept and axiom similarity matrices. Unlike the method in Chapter 5, which does not acknowledge the challenge of dealing with a large or rich ontology, we added this step to ensure that the method does not time out or run out of storage no matter what the size of the ontology is.

5. **Candidate axiom encoding**: This step describes how a new candidate axiom is introduced into the vector-space, including the case where the axiom consists

of concepts not available in the concept similarity matrix. The method in Chapter 5 does not include such a step since it naively queries all available concepts. On the other hand, since we added a step that limits our method to query only concepts found in the set of axioms used, we had to add this new step to query any new concept that might be introduced by a candidate axiom.

6. **Prediction**: This step is dedicated to training a machine learning model with the dataset (vector-space model and scores) and predicting the scores of new candidate axioms.

We begin by preparing the set of scored axioms that we want to use to train our model. Then we extract the concepts which our axiom set consists of. After that we query the ontology to retrieve the ontological distances between only the concepts we extracted. We then calculate the axiom similarity between our axioms and use it to model our axiom-based vector-space. We then utilize feature selection to reduce the size of our vector-space by reducing the number of axioms acting as dimensions to those that are most impactful, which leads to a reduction in the size of the concept and axiom similarity matrices. We train a machine learning model using our new reduced vector-space, encode new candidate axioms as a vector, and predict their scores.

7.4.1 Axiom Extraction and Scoring

We adopt two approaches to create our set of axioms. The first approach is discussed in [69]. It dictates that we first query an ontology for existing axioms of a certain type and by that we obtain a set of axioms which would have positive scores. Following that, we generate rejected axioms. We do this by constructing an axiom with a pair of random concepts, the axiom is of the form $\mathsf{subClassOf/disjointWith}(C_1 C_2)$,

with $C_1 \neq C_2$. We then check if the axiom exists in or is entailed by the ontology; if so, the generated axiom is discarded, otherwise it is kept. The resulting generated set of axioms will have a negative score. The rationale is that a randomly generated axiom can be expected to be false with a very high probability. In this method, we add a limit to the number of axioms being selected and generated, while in the method in Chapter 5 every possible combination is generated: this is a critical point when dealing with large ontologies.

Query 7.1 is used to extract a given number of existing axioms and generate a given number of random ones, followed by removing the existing from the generated. We ignore the blank nodes as well as instances where both classes are the same. This produces a balanced set of positive and negative axioms. The server used is Corese [78] which applies reasoning to check entailed axioms.

```
0  select DISTINCT ?class1 ?class2 ?label
1    where {
2        {select ?random ?class1 ?class2 ?label
3            where {
4                ?class1 a owl:Class
5                ?class2 a owl:Class
6                ?class1 rdfs:subClassOf ?class2
7                filter (!isBlank(?class1) && !isBlank(?class2) &&
                     (?class1 != ?class2))
8                bind(1.0 as ?label)
9                BIND(RAND() AS ?random) .
10               }
11       ORDER BY ?random
```

```
12      limit 500}
13  UNION{
14      select ?random ?class1 ?class2 ?label
15          where{
16              ?class1 a owl:Class
17              ?class2 a owl:Class
18              filter (!isBlank(?class1) && !isBlank(?class2) &&
                    (?class1 != ?class2))
19              bind(0 as ?label)
20              BIND(RAND() AS ?random) .
21              }
22          minus{
23              ?class1 a owl:Class
24              ?class2 a owl:Class
25              ?class1 rdfs:subClassOf ?class2
26              filter (!isBlank(?class1) && !isBlank(?class2))
27              bind(0 as ?label)
28              BIND(RAND() AS ?random) .}
29              }
30      ORDER BY ?random
31      limit 500}}
```

Query 7.1: Extraction of an rdfs:subClassOf axiom balanced set with a size of 1000 axioms using random generation.

A second, more judicious approach adopted in Chapter 5 is to only query existing axioms. For example, if we want to train a model to predict subClassOf axioms, we

would query for n subClassOf axioms and consider that as the set of positive subClassOf axioms. We would then query n disjointWith axioms and consider them as the set of negative subClassOf axioms. If one query retrieves a number of axioms lesser than the limit n, we can drop the excess axioms from the other set to maintain balance. The method in Chapter 5 applies no limit and extracts all available axioms. We again provide a limit, as methods that implement the approach described in Chapter 4 perform well even with small datasets. Query 7.2 shows our implementation.

```
SELECT ?class1 ?class2 ?label
   WHERE {
       ?class1 a owl:Class
       ?class2 a owl:Class
       ?class1 rdfs:subClassOf ?class2
       filter (!isBlank(?class1) && !isBlank(?class2) && (?class1 !=
          ?class2))
       bind(1.0 as ?label)
       BIND(RAND() AS ?random) .
   }
   ORDER BY ?random
   LIMIT 500
SELECT ?class1 ?class2 ?label
   WHERE {
       ?class1 a owl:Class
       ?class2 a owl:Class
       ?class1 owl:disjointWith ?class2
       filter (!isBlank(?class1) && !isBlank(?class2) && (?class1 !=
```

```
            ?class2))
17        bind(0 as ?label)
18        BIND(RAND() AS ?random) .
19    }
20    ORDER BY ?random
21    LIMIT 500
```

Query 7.2: Extraction of an rdfs:subClassOf axiom balanced set with a size of 1000 axioms using a negated axiom type.

The following step is to score the axioms. This is done by inputting the extracted axioms into the possibilistic heuristic [348], and receiving an output file containing the scores (ARI). We note that the process is very slow; Malchiodi et al. mention that it took a little less than a year to score 722 axioms [240], whence the need for a method such as ours.

7.4.2 Axiom Similarity Matrix

Like in Chapter 5, the axiom similarity is derived from the concept similarity. This means that we first need to query *Corese*, where the ontological distance metric is implemented as a function, to retrieve this similarity.

Query 7.3 provides our implementation of the retrieval of the concept similarity measure. In contrast with Chapter 5, this approach takes advantage of the ability to run multiple queries at a time and considers only concepts found in our set of axioms instead of all concepts in the ontology. This results in an initial reduction of computational cost and matrix size. We divide our set of concepts into k subsets, then query Corese with k queries, each including one of those subsets and the main set concepts for the distances between them. This allows us to speed up the process

of creating the concept similarity matrix 5.1 by k times. This also drastically reduces the storage space needed to store the matrices, by reducing the values to exactly what is used.

```
select * (kg:similarity(?class1, ?class2) as ?similarity)
   where {
          ?class1 a owl:Class
          ?class2 a owl:Class
          filter (!isBlank(?class1) && !isBlank(?class2) &&
          str(?class1) IN (subset) && str(?class2) IN (concepts))
   }
```

Query 7.3: Concept ontological distance retrieval.

Next, we use the algorithm explained in Section 5.3.2 to calculate the similarity between axioms. We end up with an $m \times m$ axiom similarity matrix 5.2. The diagonal of this matrix will contain only 1s as the similarity between an axiom A and itself is 1.

7.4.3 Axiom Base Vector-Space Modeling

We model our vector space to encode axioms into vectors. The initial number of dimensions d of this vector space is equal to the number of axioms in our scored axiom set. Each axiom can be represented as a vector V in this d-dimensional space. Considering the components of our vectors are the similarities between axioms, it would be intuitive to view our axiom similarity matrix as a representation of our axiom-based vector space. This step is exactly the same as in the previous Chapters, in that the shape and structure are the same for now. However, the time needed for construction is greatly reduced due to optimized querying.

7.4.4 Vector-Space Dimension Reduction

We now shift our focus to ranking and reducing the dimensions of our vector space. By doing so we achieve the following:

- A reduction in the error rate due to the reduction in noise and redundancy.
- A reduction in the size of our vector-space and storage space for the axiom similarity matrix.
- A reduction in the size of our concept similarity matrix. The matrix will only include concepts that constitute axioms acting as dimensions in our vector-space.
- A reduction in the computational complexity when encoding new candidate axioms into the vector-space. We will be comparing the new axiom to a subset of the initial axioms acting as dimensions d.
- A reduction in the look-up time when retrieving the concept similarity value from the new smaller concept similarity matrix.
- A better dataset for our machine learning model to train on with regards to redundancy.

To this aim, we consider our dimensions as features and apply a supervised filter-type feature selection method such as mutual information. This works by taking as input the axiom similarity matrix along with the scores of the axioms and returning a ranking of the dimensions from the most to the least impactful. We then keep a percentage of these dimensions according to their ranks and discard the rest.

Our new axiom similarity matrix is of size $m \times z$, z being the number of dimensions selected from the original dimensions d. This in turn affects the concept similarity

matrix, which will become of size $n \times f$, f being the number of concepts included in the selected axioms acting as dimensions in our reduced vector space. New candidate axioms will be encoded into the vector space with the reduced number of dimensions. This means lower processing complexity in terms of computation cost and storage cost, which are the keys for scalability.

7.4.5 Candidate Axiom Encoding

The candidate axiom encoding process includes two cases. In the first case, the candidate axiom is made up of two concepts already found in our concept similarity matrix. If so, the candidate axiom goes straight through the algorithm mentioned in Section 7.3.1, as we did with the training set. The candidate then becomes a new vector in our vector-space, ready for score prediction.

The second case covers candidate axioms that contain concepts not found in our concept similarity matrix. Such candidates invoke a new similarity measure retrieval query 7.3. In this query the *subset* variable includes the new concepts and the *concepts* variable includes the concepts found in the new reduced set of axioms acting as dimensions. This produces at most two new rows to be added to the matrix, if none of the candidate's concepts are in the concept similarity matrix. After that, the candidate proceeds normally through the vector encoding algorithm.

7.4.6 Prediction

Now that we have our reduced vector space, we can apply machine learning methods. A simplistic method such as k-NN can be used to highlight the strength of our similarity measure, so we use it along with more sophisticated methods such as random-forest regressors. We choose this method to be able to compare with the method proposed in Chapter 5, since it achieves it's best results using that method.

7.5 Experimentation Protocol

We use the following hardware configuration for our experiments:

- CPU: Intel(R) Xeon(R) CPU W-11955M @ 2.60GHz base and 4.5 GHz all core boost. With 8 cores and 16 threads.
- A total of 128 GB of RAM memory with frequency 3200 MHZ.
- 1 TB of NVME SSD storage with read and write speeds of up to 2000 MB per second.

In addition, the code [2] uses the Python multiprocessing package to distribute the dataset-building and querying tasks over all available cores.

We searched for ontologies of different sizes and domains to use in our experiments and selected the following ones. Each described in Section 3.1.1.

- DBpedia [3] .
- GO [4] .
- CL [5] .

We will be addressing the method proposed in Chapter 5 as baseline.

For DBpedia, and to be able to compare with the baseline, we use the same scored subClassOf dataset consisting of 722 axioms. The dataset is scored using the possibilistic heuristic explained in Section 3.2. As for GO and CL, and since the baseline does not experiment using these ontologies, we create our own disjointWith

[2] https://github.com/ali-ballout/Scalable-Prediction-of-Atomic-Candidate-OWL-Class-Axioms-Using-a-Vector-Space-Dimension-Reduced-Appr

[3] https://www.dbpedia.org/resources/ontology/

[4] http://geneontology.org/docs/download-ontology/

[5] https://www.ebi.ac.uk/ols/ontologies/cl

datasets using the process described in Section 7.4.1. For each ontology, we create a balanced set of 600 axioms. For the sake of experimentation and since scoring 722 axioms in our smallest tested ontology took little under a year [240], we gave the score of 1 to all positive axioms and 0 to all negative axioms. This would turn the task in those cases to classification, which is not an issue, as long as the approach proves scalable and with good accuracy. After all, the work that is the foundation for all these vector-space approaches (Chapter 4) is presented as a formula classifier.

Following the preparation of our dataset, we are now able to train a regressor, in the case of DBpedia, for performance experimentation in terms of error rate. To compare with the baseline, we use a random forest regressor, with which the baseline achieves its best results. In our experiments we consider **Processing time** as the time needed in seconds to construct the concept similarity matrix (CSM) and the axiom similarity matrix (ASM) and **Encoding time** the time needed to encode cone new candidate axiom. And in regards to storage cost, we consider the size in mega bytes (MB) for the ASM and in number of values stored for the CSM.

With the smallest ontology, DBpedia, we perform experiments to analyse the effect of feature selection on prediction accuracy. Since the DBpedia datasets are correctly scored using [348], they are the most suitable to use for this type of analysis. The results for these experiments are presented in Table 7.1 and Figure 7.1.

Using GO and CL we perform experiments to analyse the impact of our current approach on storage usage and computational cost. For these experiments we set our feature selection percentage at 40% based on the results from Figure 7.1. We also note that in these experiments we do not include the performance metric for lack of space and since its not our main concern, but would like to highlight that an average F1 score of 0.86 was achieved in both experiments. We compare the feasibility of both our current approach and the baseline when applied to these ontologies. The

Table 7.1: Comparison of computational cost in seconds as well as storage cost in number of values for CSM using using the DBpedia scored subClassOf dataset.

Approach	Number of axioms processed	Initial CSM size	Concepts queried	Processing time	Encoding time
baseline	722	580,644	762	13.72	0.019
current approach	722	**85,264**	**292**	**3.86**	**0.005**

results are presented in Table 7.2 for GO and Table 7.3 for CL.

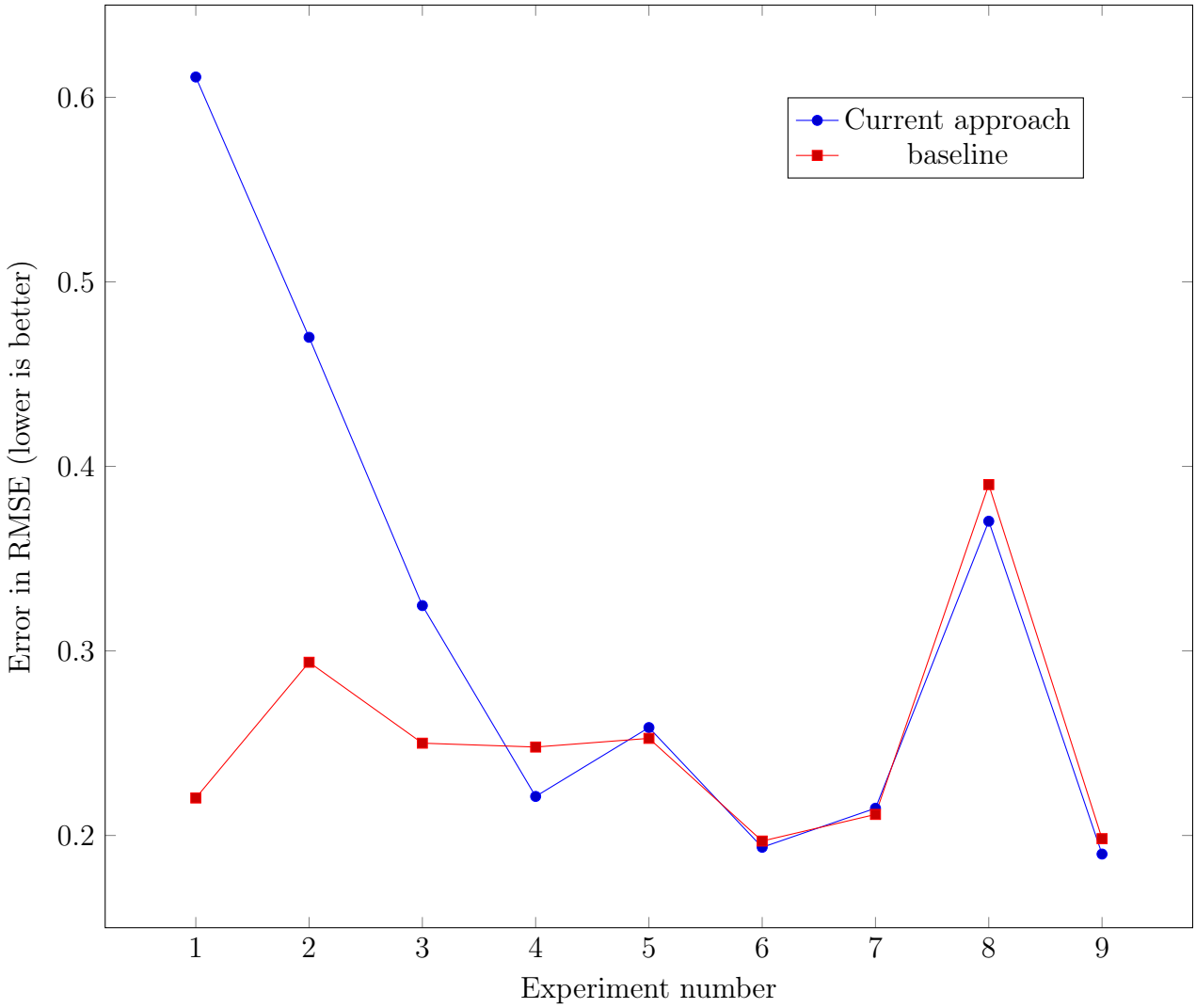


Figure 7.1: A graph comparing the performance of our current proposed approach and the baseline in RMSE using the subClassOf dataset containing 722 axioms throughout 9 experiments (train/test splits), in each experiment the baseline is trained using 100% of the dimensions, while our model is trained using a selected percentage of the dimensions equivalent to the experiment's number × 10% so from 10% to 90%.

Table 7.2: Comparison of computational cost in seconds as well as storage cost in MB for ASM and number of values for CSM using the GO disjointWith dataset. Time out error: TO.

Approach	Number of axioms processed	ASM size	Initial CSM size	Concepts queried	Processing time	Encoding time
baseline	600	62,000	6,214,957,225	78,835	TO	TO
proposed approach	600	**5.1**	**95,481**	**309**	**312.54**	**0.034**

Table 7.3: Comparison of computational cost in seconds as well as storage cost in MB for ASM and number of values for CSM using the CL disjointWith dataset. Time out error: TO.

Approach	Number of axioms processed	ASM size	Initial CSM size	Concepts queried	Processing time	Encoding time
baseline	600	8,000	874,680,625	29,575	TO	TO
proposed approach	600	**2.05**	**216,225**	**465**	**103.7**	**0.015**

7.6 Results and Analysis

Figure 7.1 depicts the performance of both the current proposed method and the baseline. It compares the error rate in terms of RMSE for both methods through out a series of 9 experiments where the percentage of dimensions selected in our method is incremented by 10% each run. This plot shows the effect of feature selection on the prediction accuracy of the model. We can see that when a very small number of dimensions is selected (< 30%) the method cannot make accurate predictions. The error rate decreases as we increase the number of selected dimensions until we reach 40%. Here we can see that our current approach performs better with fewer dimensions that the baseline. After the 40% mark, we get a similar or slightly better performance. We can conclude from this that 40% to 50% can be considered an optimal percentage of dimensions to remove redundancy and improve performance. This is exactly the reason why we set the feature selection percentage parameter to 40% for the GO and CL experiments.

Table 7.1 highlights the effects of our current approach on computational cost and storage cost. Due to our method querying only selected concepts instead of all concepts, we can see that the initial size of the concept similarity matrix is almost seven times smaller than that in the baseline while processing with the same set of axioms. This smaller size results in faster look-ups to calculate the axiom similarity measure, which leads to a reduction in time cost from 13.7 seconds to 3.8 seconds. Also, fewer dimensions in our vector-space lead to a faster encoding time for a candidate axiom as we can see a reduction from 0.019 seconds to 0.0053 seconds.

In Table 7.2, we see from the size of the CSM and the concepts queried, that our method has scaled well from a small-size ontology such as DBpedia to a larger one such as GO. Even though the number of concepts in the ontology addressed

changes from 762 to 78,835 our method is able to maintain almost the same size of the CSM by dealing with a relatively small number of concepts (309). When compared to the baseline, we notice that it is unfeasible to apply the method to the ontology. Concerning computational cost, the baseline times out and crashes without completing the task. As for storage cost, the size of the CSM for the baseline would be approximately 62 Gbytes compared to 5.1 Mbytes for our method.

We notice that our current approach consumes an increased amount of processing time up to 312 seconds but maintains a very short axiom encoding time of 0.034 seconds. This increase in processing time is attributed to the querying of the semantic similarity measure in such a large ontology. It is dependant on the capability of the SPARQL endpoint and the size of the ontology. However, this is well within acceptable time.

Similarly, when dealing with the medium-size CL, having 29,575 concepts, our current approach displays consistency and stability in terms of storage and computation. Processing time is 103 s, which falls within expectation when compared to the processing time of GO, the same can be said for the encoding time. Again, the baseline times out and crashes proving neither feasible nor scalable.

7.7 Conclusion

We have proposed a scalable approach for the score prediction of atomic candidate *OWL* class axioms of different types. The method relies on a semantic similarity measure derived from the ontological distance between concepts in a subsumption hierarchy, as well as feature selection for vector-space dimension reduction. Extensive tests that covered a range of ontologies of different sizes as well as multiple parameters and settings were carried out to investigate the effectiveness and scalability of the method.

The results obtained support the effectiveness of the proposed method in predicting the scores of the considered OWL axiom types with lower error rates than the baseline. More importantly, it does so while being scalable, consistent and stable when dealing with ontologies of different sizes. This allows us to confidently say that our proposed method is feasible and able to address large real-world ontologies.

Chapter 8

OCASP OWL CLASS AXIOM SCORE PREDICTOR WITH ACTIVE LEARNING

Semantic applications require expressive schemas, or ontologies, to exploit the full potential of knowledge graphs, which have become ubiquitous to express and organize knowledge in many domains. Existing knowledge graphs are rich in factual information, but the available schemas, handcrafted by knowledge engineers, are often minimal and not always complied with by the contributors of factual data. A key challenge to bridge this gap is to develop methods to induce schema axioms from factual data. This task can be broken down to two steps: (i) generating candidate axioms and (ii) scoring them against the available evidence for acceptability. Here, we focus on the latter, crucial step. Current methods such as OWL2Vec*, Onto2Vec, and OPA2Vec have shown promise, but accurately predicting the acceptability of axioms not logically deducible from an ontology remains a challenge. This chapter introduces OCASP, **O**WL **C**lass **A**xiom **S**core **P**redictor, a novel active learning approach to address this issue. The approach exploits a semantic Web reasoner, a data-driven axiom scoring heuristic, and an existing semantic similarity, which we extend to include complex class axioms, this defining an axiom-based embedding space. Our approach aims at providing a more efficient and robust solution for expressive schema induction, overcoming the limitations of existing methods.

8.1 Introduction

Now that we have addressed scalability, we turn our focus to the challenge of maximizing accuracy. Active learning emerges as a promising solution to this challenge. By involving the model in the learning process, it can selectively query the instances

for which it needs labels, leading to improved accuracy with less data [325].

We introduce a novel active learning approach for predicting the acceptability of OWL class axioms in ontology completion. We leverage the semantic similarity measure introduces in Section 5.3.2 and extend it to include complex class axioms. Our model aims to overcome existing method limitations, offering a more robust solution for ontology completion.

The rest of the chapter is organized as follows: Section 8.2 provides a review of related work, Section 8.3 describes how we extended our axiom semantic similarity measure, Section 8.4 presents the method and model structure, Section 8.5 details our experimental protocol and results, and Section 8.6 concludes the chapter.

8.2 Embedding Methods and Active Learning: A Recap

Embedding Methods: Methods like Onto2Vec, OPA2Vec, and OWL2Vec* rely on embeddings [70, 336, 337]. They are resource-intensive, domain-specific, and sensitive to input quality. They also struggle with complex class relations and differentiating similar sentences [170, 199, 248]. Their limitations suggest the need for semantically-based solutions. For more information on these embedding based methods please refer to Section 3.6.

Active Learning: As a learning paradigm, active learning offers efficiency and accuracy but needs proper integration with ontology completion methods [318, 325]. For more information on Active Learning and its techniques please refer to Section 2.5.4.

Proposed Approach: We propose an active learning approach using an ensemble model and two-layered oracle, including Hermit [138]. Our method extends the semantic similarity introduced in Section 5.3.2 to address complex axioms and aims to overcome limitations in previous methods, such as computational cost, robustness against incomplete or noisy ontologies, and handling of complex class relations. In

this chapter we will be comparing our new method to the one we proposed in Chapter 7. We will be addressing that method as baseline. We will also be comparing to state-of-the-art embeddings based methods described in Section 3.6.

8.3 Axiom Semantic Similarity

We extended the axiom similarity measure proposed in Section 5.3.2 to address some OWL constructs, namely intersection, union, and complement. This implementation makes the calculation of this distance faster and more scalable. We do not change the way the measure is extended from classes to atomic axioms.

8.3.1 Class similarity

In our approach, we wrote our own implementation of the concept distance measure described by Gandon in his thesis [133]. In this implementation we take into consideration equivalence axioms. Where in a class a has the same distance from two equivalent classes. And those equivalent classes have a distance of 0 between each other. We also optimized the calculation to compute distances incrementally and only on demand. This means less time need since no useless distance calculations are taking place, as well as a much lesser need for storage since only required distances are being calculated.

We first compute the depth of each class in the ontology. The depth of a class is defined as the length of the longest path from the class to the root of the IS-A hierarchy, $D(t) = 1 + \max(D(p))$, for all parent p of class t. The distance between each class and its ancestors is then computed as:

$$Dist(t, a) = \sum_{i=D(t)}^{D(a)} \frac{1}{2^i} \tag{8.1}$$

For each pair of classes, the distance is computed as:

$$Dist(t1, t2) = Dist(t1, a) + Dist(t2, a) \tag{8.2}$$

With class a being their deepest common ancestor. The distance is normalized by:

$$NormDist(t1, t2) = \frac{Dist(t1, t2)}{MaxDist} \tag{8.3}$$

With $MaxDist$ being the distance between the deepest class and the root multiplied by two. Finally, the similarity between the two classes is computed as:

$$Sim(t1, t2) = \frac{1}{1 + k \times NormDist(t1, t2)} \tag{8.4}$$

Where k is a scaling constant given by the equation $k = \frac{2^{\text{MaxDepth}}}{100}$ having MaxDepth as the depth of the deepest class in the hierarchy. If two classes are the same or equivalent, the distance is 0 and the similarity is 1.

8.3.2 Axiom Similarity

We extend the class similarity to axioms, defining the similarity between two atomic axioms candidate axiom ϕ and feature axiom ψ where L_ϕ, R_ϕ, L_ψ, and R_ψ are classes in the axioms (**L**eft and **R**ight sides). We highlight that axioms set as features in our vector space are always exclusively atomic, the concepts on the left and right side of these axioms contain no constructors but only named classes.

$$S(\phi, \psi) = \max\left(\frac{\text{sim}(L_\phi, L_\psi) + \text{sim}(R_\phi, R_\psi)}{2}, \right. \\ \left. \mathbb{I}_{\text{type}}(\phi, \psi) \cdot \frac{\text{sim}(L_\phi, R_\psi) + \text{sim}(R_\phi, L_\psi)}{2}\right) \tag{8.5}$$

where $\mathbb{I}_{\text{type}}(\phi, \psi)$ is an indicator function that is 1 if the type of the axioms ϕ and ψ is either Disjointness or Equivalence, and 0 otherwise.

8.3.3 Complex Axiom Similarity

We further extend the axiom similarity to complex axioms by considering the constructors (union, $\sqcup$, intersection. $\sqcap$, and complement, $\neg$) in the classes. Let ϕ be a complex axiom. Let C_ϕ be the set of classes in the complex concept L_ϕ (or R_ϕ), and c_ψ be the class L_ψ (or R_ψ).

- For $\sqcup$: $\text{sim}(L_\phi, L_\psi) = \max_{c_\phi \in C_\phi} \text{sim}(c_\phi, c_\psi)$;
- For $\sqcap$: $\text{sim}(L_\phi, L_\psi) = \min_{c_\phi \in C_\phi} \text{sim}(c_\phi, c_\psi)$;
- For $\neg$: $\text{sim}(L_\phi, L_\psi) = 1 - \text{sim}(c_\phi, c_\psi)$.

8.4 Method and Model Structure

Our method is inspired by the principles of Active Learning [318]. It involves training three smaller models using pool axiom learning (see Section 8.4.3). Each model predicts a label for the candidate, and if one model disagrees with the other two, we call upon an oracle to check the label. The first layer in our oracle stack is a DL reasoner, and the second layer oracle is a slower but accurate possibilistic heuristic that may be able to label candidate axioms not logically deducible from the ontology. If even the second oracle returns "ignorance" due to insufficient data or evidence, we consider the majority label given by our three models as the designated label. The incorrect models are then retrained by adding the newly labeled candidate to their respective training sets. Figure 8.1 illustrates our ensemble model structure and the process of active learning and candidate axiom labeling.

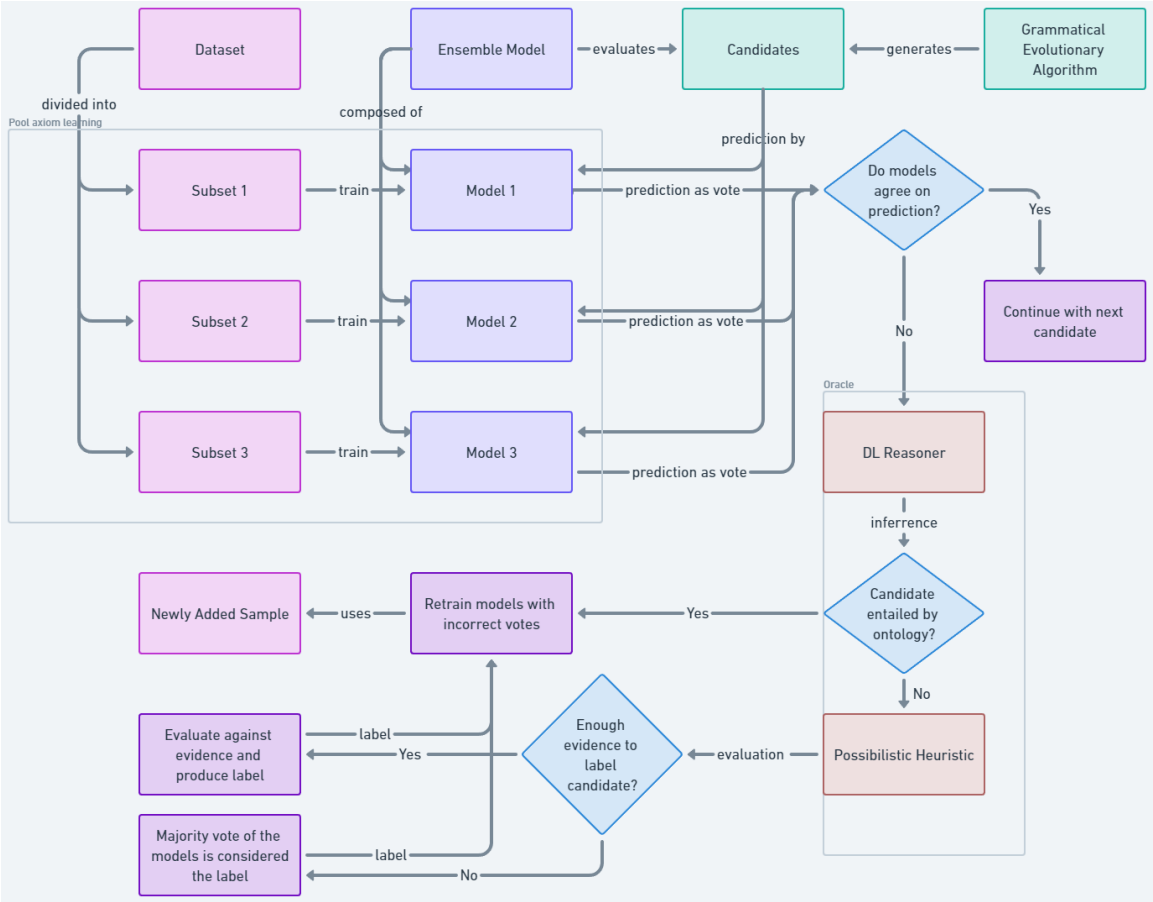


Figure 8.1: Depiction of our model's structure, prediction process and active learning process.

8.4.1 Preprocessing and Feature Extraction

Ontology Parsing and Structure Analysis

Ontology parsing and structure analysis is the initial phase in the preprocessing of the ontology data. It involves understanding the ontology's hierarchical structure.

- **Ontology Parsing**: The raw ontology file is parsed to extract the axioms and classes.
- **Hierarchy Extraction**: The hierarchy within the ontology is identified, this

includes the subClassOf relation between classes.

- **Relationship Analysis**: The relationships between different classes are analyzed, such as equivalence, disjointness and subsumption. This information is used later in the similarity computation in Section 8.4.1.

Axiom Similarity Computation

Axiom similarity computation is crucial in building a semantic understanding of the ontology. It involves the following steps:

- **Semantic Analysis**: Using the relationships extracted from Section 8.4.1, the semantic meanings of axioms are analyzed and the axioms are parsed into trees. In the tree each leaf is a named class and each node is a logical constructor.
- **Similarity Metrics**: The similarity metric we detail in Section 8.3 is applied to compute the similarity between axioms.

Vector Space Construction

Vector space construction is the process of representing axioms in a numerical format suitable for machine learning models. It involves:

- **Vector Space Dimensionality**: All atomic axioms included in our axiom set are considered as dimensions/features in our vector space.
- **Feature Vector Creation**: Based on the similarity computed in Section 8.4.1, feature vectors are created for each axiom. Each axiom is compared with all the axioms that make up our vector space dimensionality.
- **Dimensionality Reduction**: we can use feature ranking and selection techniques to reduce the dimensionality of the vector space.

8.4.2 Ensemble Model Structure

Individual Model Architecture

The ensemble model is constructed by integrating various individual models, each of which contributes to the final prediction. The architecture of individual models includes:

- **Model Type**: The model type used in our ensemble is random forests (RF), this is due to their proven effectiveness when used for this task by the state-of-the-art [70, 336, 337] (SOTA).
- **Configuration**: We set the hyper parameters of the random forest models to the following: number of estimators 200, empirically between 100 and 200 (200 being best) produces the best results, in our model and the SOTA. A max depth is not set allowing node expansion until all leaves are pure.
- **Training**: The models are trained on the feature vectors derived from Section 8.4.1. We split the sample data into 3 unique non-overlapping sets having the same vector space dimensionality. Each model then receives one of these unique subsets. We do this to guarantee that we create 3 distinct sub-models especially when using the same classifier, thus ensuring the ensemble is robust and able to capture as much information from the training data as possible without noise or redundancy.

Model Integration

The integration of the individual models into an ensemble follows the Query by Committee approach [319]. Here three sub-models contribute equally to the final prediction. The process includes:

- **Equal Weighting**: Each sub-model has an equal weight in the voting process.
- **Voting Scheme**: The three sub-models make predictions, and a majority vote is considered as the final decision. If there is unanimous agreement among the sub-models, the prediction is finalized.
- **Oracle Consultation**: In the case where only one sub-model disagrees with the others, an oracle is consulted to determine the true label as described in Section 8.4.3.
- **Oracle Decision Making**:
  - *Definitive Oracle Decision*: If the oracle returns a label, its decision is considered final.
  - *Oracle Ignorance*: In case the oracle returns ignorance due to lack of data, the majority vote among the sub-models is considered as the final prediction.

8.4.3 Active Learning Process

Our active learning process plays a crucial role in the continuous improvement of our model and assuring that we get the best performance we can out of our model. This process includes the following stages:

Pool-Based Sampling

Pool-based sampling is employed using the sub-model's assigned data subset. With this technique the model starts out with a very small *initial* sample to learn and then iteratively adds more. In our case, we already split our data into 3 smaller subsets, and so we increase the size of the initial sample so that our sub-models have a meaningful sample to initially learn before iteratively querying for more samples.

The process is as follows:

- **Initial Training Set Selection**: We do this for each individual sub-model in our ensemble, and using its respective subset of the data. A subset of instances is randomly chosen to form the initial training set, we set this to 25% of the sub-model's data subset size. In pool-based active learning, the initial training set is randomly selected from the available data, and should be a small percentage of the whole. The percentage can be decided based on the amount of instances available, the number of iterations and the samples to query per iteration. Since we are splitting our data into 3 subsets, taking a very small percentage from an already small set is not enough to train a model. Also, to optimize the iterative training process we select a quarter of the data and then iterate to learn the other most informative quarter by querying. This insures that our models are adequately trained using sparse instances from the data, and then reinforced with data they find as most informative via uncertainty sampling.

- **Sample Selection**: This is done over n iterations with a set m of samples to learn per iteration. Our aim is to learn the second most informative quarter of the dataset For this we hope to learn a total of $|dataset|/4$ instances through out all iterations. The higher the number of iterations, the lower the number of samples learnt per iteration. Higher iterations mean better or more informative samples being selected at the cost of higher training time. In our experiments we set the number of iterations to 10 to learn 2.5% of the dataset per iteration.

- **Uncertainty Sampling**: Instances where the model has the greatest uncertainty are prioritized. The most informative instances are selected as samples to learn and added to the training set. So instances with the probability of being classified as *Accepted* or *Rejected* is close to 50/50 get the highest priority

to be queried.

Oracle Integration

The oracle is used to determine the true label of candidates when at least one sub-model disagrees with the others, and the process includes:

- **Multi-Layer Oracle Calling**: Different layers of oracles are called upon depending on the responses.
  - *Description Logic Reasoner*: A DL reasoner is a tool used to infer logical consequences from a set of asserted facts or axioms [242]. In our method, the DL reasoner serves as the first-layer oracle to check the label of a candidate when there is disagreement among models. It is faster than the second-layer oracle but is only able to determine if a candidate is entailed.
  - *Possibilistic Heuristic*: Our second-layer oracle, the possibilistic heuristic, is a slower but accurate oracle used to check the label of a candidate. This heuristic is capable of labeling axioms that are undeducible from the ontology using a DL reasoner. Based on possibility theory [104, 153], it considers confirmations, counterexamples, and cases of ignorance within an RDF dataset, and defines the possibility and necessity measures of the axiom. These measures are combined into an acceptance/rejection index (ARI) that ranges from -1 to 1, reflecting the axiom's acceptance, rejection, or a state of ignorance [348].
- **Label Acquisition**: The oracle structure, consisting of two layers, is designed to efficiently and effectively handle a variety of candidates. The first layer employs a DL reasoner. Its role is to evaluate whether a candidate is entailed by the existing knowledge base (KB). If so, the candidate is labeled as *Accepted*.

This approach is particularly efficient for candidates that are direct logical consequences of the available axioms. However, a DL reasoner operates within certain limitations. It can only determine if a formula is a consequence of the available axioms, and it does not necessarily imply that a non-consequence formula is false or impossible, unless a contradiction is explicitly present within the set of available axioms. Moreover, OWL syntax does not support expressing the negation of an axiom directly, posing a challenge for the reasoner to definitively refute a candidate formula. To compensate for these limitations, a second layer is invoked when the DL reasoner cannot entail a candidate. This layer uses an RDF mining tool, RDFMiner, [1] to provide a more definitive *Accepted*, *Rejected*, or *Ignorance* label based on the evidence present in the RDF data. While RDFMiner's heuristic for negation is not as exact as a reasoner, it offers a more comprehensive evaluation, particularly for candidates that are not directly deducible through reasoning. The architecture of this two-layer oracle is thus a balanced and pragmatic approach that leverages the strengths of both DL reasoning and RDF mining while addressing their limitations. This enables the model to efficiently and effectively handle a wide variety of candidates, enhancing its overall performance and utility in knowledge acquisition and reasoning tasks.

- **Ignorance Handling**: In case of lack of data or uncertainty, the oracle may return ignorance, and the process detailed in Section 8.4.2 under *Oracle Decision Making, Oracle Ignorance* is followed.

[1] `https://github.com/RemiFELIN/RDFMining`

Stream-Based Learning and Model Retraining

With the new labeled data obtained from the oracle or majority vote, the models are updated and retrained as follows:

- **Incorporation of New Data**: The newly labeled instances are added to the respective training set of the sub-models whose prediction is different from the obtained label.
- **Model Retraining**: The individual sub-models, within the ensemble, whose prediction is different from the obtained label are retrained using their respective updated training set.
- **Continuous Improvement**: This iterative process continues, allowing the model to learn from the most informative instances and adapt to the underlying patterns in the data.

8.5 EXPERIMENTS & RESULTS

For the following experiments, the machine that was used had the following hardware configuration:

- Processor: Intel(R) Xeon(R) CPU W-11955M @ 2.60GHz.
- Memory: A total of 128 GB of RAM.

All our code and the data needed to replicate the experiments can be found in our repository. [2]

[2]https://github.com/ali-ballout/ocasp-complex-active

8.5.1 Dataset Preparation

Ontologies

For our experiments we use 3 ontologies of different sizes covering various domains as described in Section 3.1.1.

- **GO** [3] .
- **FoodOn** [4] .
- **DBpedia** [5] .

We use GO and FoodOn to compare our work with OWL2Vec*, OPA2Vec and Onto2Vec for the task of class subsumption prediction. As for DBpedia, we use it to compare our work with the baseline in the task of class subsumption and disjointness prediction. We also use DBpedia to demonstrate our models performance when classifying complex axioms.

Axiom Datasets

We used two different processes to produce datasets for our experiments.

- **FoodOn and GO**: We start out by randomly selecting n rdfs:subClassOf axioms from the ontology. We proceed to remove these axioms from the original ontology and make sure that they can not be entailed by the edited ontology using a DL reasoner. This means that we now have a set of real *Accepted* axioms that are not deducible by the DL reasoner from the available ontology. After that, we generate random rdfs:subClassOf axioms of equal number and make

[3] http://geneontology.org/docs/download-ontology/

[4] https://foodon.org/

[5] https://www.dbpedia.org/resources/ontology/

sure that they are not present in the ontologies nor entailed according to the DL reasoner. This means that we now have a set of *Rejected* axioms. This is not the ideal way to obtain negative samples, but it is the approach followed by OWL2Vec* and in the aim of keeping the comparison fair we follow the same approach. We now have a balanced set of labeled axioms of size $2n$. We split this set into two sets of equal size, one used for training and one used for testing.

- **DBpedia**: For these experiments we obtained a set of axioms generated by the grammatical evolution algorithm [122, 123, 265, 264], and labeled using the possibilistic heuristic [348]. We produced 4 different datasets:
  - Atomic disjointWith axioms.
  - Atomic subClassOf axioms.
  - Complex disjointWith axioms.
  - Complex subClassOf axioms.

Vector Space

After producing our axiom sets we create our axiom-based vector space (embedding) following the process detailed in Section 8.4.1. For axiom sets having atomic axioms, complex axioms or a combination of both, only the atomic axioms are considered as features. Each axiom in our axiom set is compared to all the feature axioms of the vector space and the similarity described in Section 8.3.2 is calculated and set as the weight of that feature. All axioms are encoded this way to obtain our vector space.

8.5.2 Ensemble Setup

The setup of the ensemble model involves the configuration of individual models and the determination of their interaction within the ensemble structure. The process

of ensemble setup is detailed as follows:

- **Model Selection**: The ensemble is composed of three Random Forest (RF) models as mentioned in Section 8.4.2 under *Model Type*.
- **Model Configuration**: Each RF model is configured with 200 estimators and a maximum depth is not set as described in Section 8.4.2 under *Configuration*.
- **Integration Mechanism**: The models are integrated into an ensemble using a majority voting scheme. This democratic approach gives equal weight to each model's predictions as described in Section 8.4.2 under *Equal Weighting* and *Voting Scheme*.
- **Oracle Setup**: Both layers described in Section 8.4.3 are set up as services that receive the candidate and ontology as input and output a label (Accepted, Rejected, Ignorance).

8.5.3 Ensemble Training

The training process of the ensemble model takes place in multiple stages, including data splitting, initial training, pool-based training, prediction, oracle consultation, and retraining. This last three stages are iterative and follows the principles of Active Learning, allowing the model to learn and adapt to the underlying patterns in the data over time.

- **Data Splitting**: The available labeled data is split into three unique subsets and assigned to the RF models as described in Section 8.4.2 under *Training*.
- **Initial Model Training**: Each sub-model is initially trained on 25% of its assigned data subset. This percentage is chosen to provide each sub-model with

a sufficient initial training set, while also leaving enough data for the iterative learning process as detailed in Section 8.4.3 under *Initial Training Set Selection.*

- **Iterative Pool-Based Learning**: After the initial training, the sub-models enter an iterative learning phase. In each iteration, the sub-models identify and select the most informative instances from their remaining data subsets based on uncertainty as detailed in Section 8.4.3 under *Uncertainty Sampling.*

- **Stopping Criterion**: We define our stopping criterion to be when we have learned a quarter of the dataset size across all iterations as in Section 8.4.3 under *Sample Selection.*

- **Continuous Stream-Based Learning**: During testing/prediction, if an oracle is called due to a disagreement between the sub-models according to Section 8.4.3 and after a label is acquired, the sub-models whose predictions are inconsistent with the newly obtained label are updated and retrained using their respective updated training sets, as described in Section 8.4.3.

8.5.4 chapter5testing

We performed multiple experiments, which we will split into two groups. The first group includes the experiments done on GO and FoodOn for predicting the acceptability of atomic subClassOf axioms. The second group includes experiments done on DBpedia for predicting the acceptability of atomic and complex subClassOf and disjointWith axioms.

Group 1

For the GO dataset, the training set contained contained 4,593 instances of *Rejected* axioms and 5,693 instances of *Accepted* axioms, while the test set contained 4,705

instances of *Rejected* axioms and 5,580 instances of *Accepted* axioms.

For the FoodOn dataset, the training set contained 3,170 instances of *Rejected* axioms and 2,995 instances of *Accepted* axioms, while the test set contained 3,143 instances of *Rejected* axioms and 2,962 instances of *Accepted* axioms.

For this group of experiments we compare our model to the embeddings based models, as well the baseline (our method from Chapter 7).

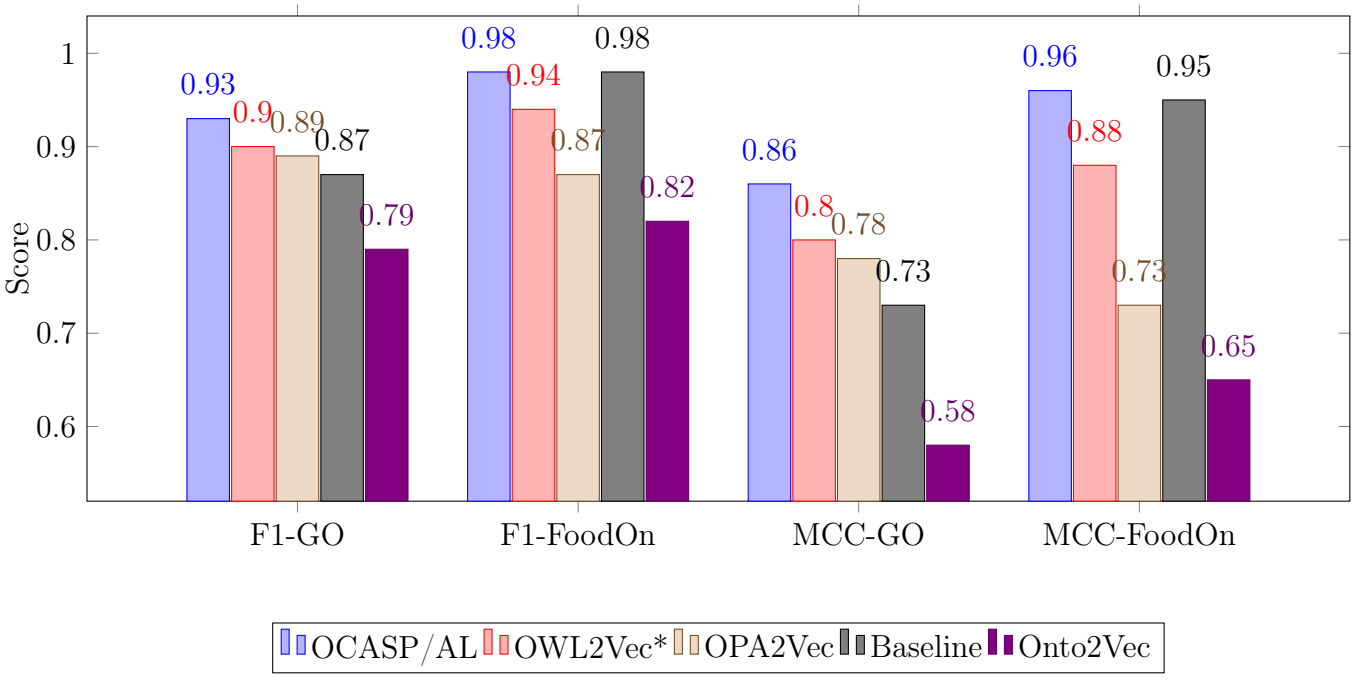


Figure 8.2: Performance of the models in classifying candidate subClassOf axioms for the GO and FoodOn datasets in terms of F1 score and MCC (Matthews Correlation Coefficient)

Group 2

For DBpedia we did 4 different experiments:

- **Atomic disjointWith**: The training set contains 596 *Rejected* and 524 *Accepted* axioms. While the testing set contains 227 *Rejected* and 752 *Accepted*

axioms.

- **Atomic subClassOf**: The training set contains 398 *Rejected* and 399 *Accepted* axioms. While the testing set contains 347 *Rejected* and 412 *Accepted* axioms.

- **Complex disjoinWith**: For this experiment we have a training set containing complex axioms along atomic axioms with labels distributed as follows: 736 *Rejected* and 817 *Accepted* axioms. We also have a features set which includes atomic axioms from the training set with labels distributed as follows: 596 *Rejected* and 524 *Accepted* axioms. As for the testing set, we included only complex axioms to focus on the models ability to label those axioms. The label distribution of our testing set was as follows: 206 *Rejected* and 1618 *Accepted* axioms.

- **Complex subClassOf**: Producing *Accepted* subClassOf axioms is difficult especially when complex constructs are involved. For this, we had to use only atomic axioms in the training set while we use the test complex axioms in the test set. The training set we used was the same used for the atomic subClassOf experiment. While the testing set contains 101 *Rejected* and 17 *Accepted* axioms.

For this group of experiments we compare our work with the baseline.

The testing process involved feeding the test data to the trained models and comparing the predicted labels with the actual labels. The performance of the models was evaluated using various metrics such as F1-score and MCC score. The results of the first group of experiments are presented in Figure 8.2. The results of the second group are presented in Figure 8.3. Table 8.1 gives some insight on how the model performs in terms of MCC and number of oracle queries.

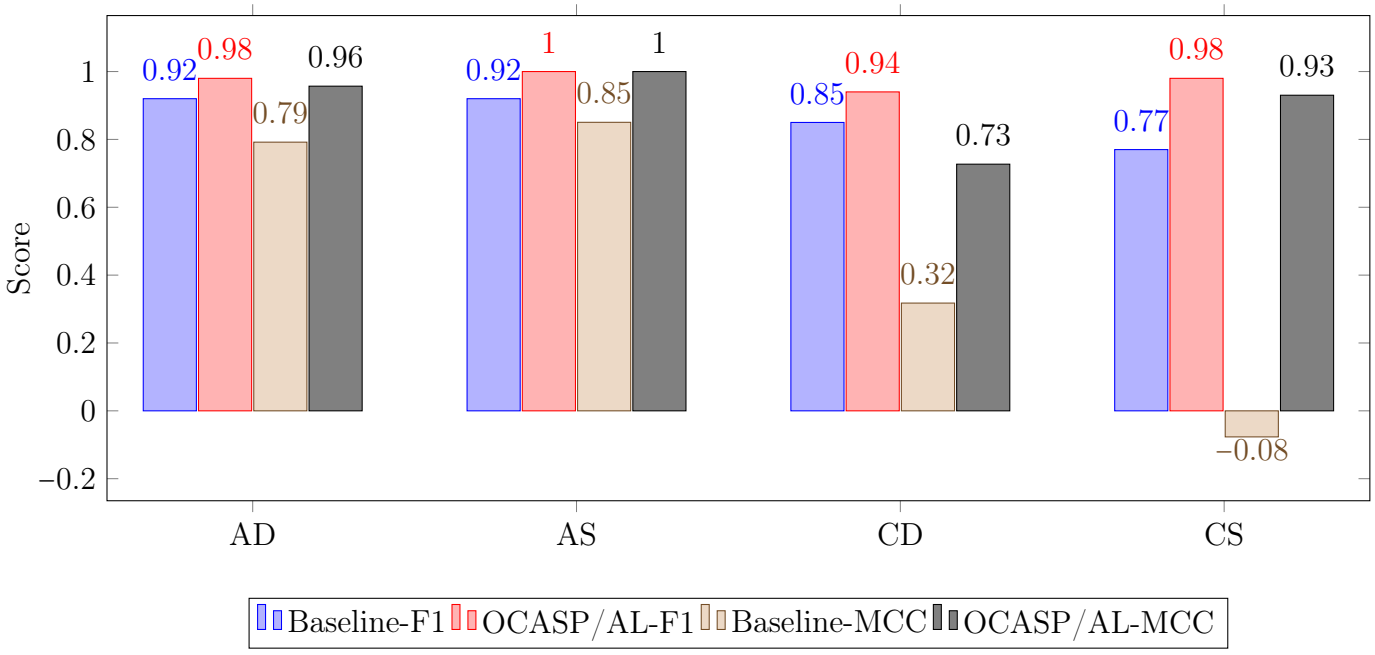


Figure 8.3: Comparison of F1 and MCC Scores for DBpedia Ontology (AD: Atomic Disjoint, AS: Atomic Subclass, CD: Complex Disjoint, CS: Complex Subclass)

8.5.5 Results and Analysis

Performance Analysis of OCASP/AL

Group 1: GO and FoodOn Datasets The performance of OCASP/AL on the GO and FoodOn datasets (Figure 8.2) demonstrates a strong capability in classifying atomic subClassOf axioms.

- **High F1-Scores**: OCASP/AL achieved an F1-score of 0.93 for GO and 0.98 for FoodOn, reflecting high precision and recall.
- **Strong MCC**: With MCC scores of 0.86 for GO and 0.96 for FoodOn, OCASP/AL's predictions were in robust correlation with the actual classifications.

Table 8.1: MCC score and number of queries made to the oracle over the total number of instances predicted by OCASP/AL per experiment.

Experiment	**MCC**	**Queries/Total**
GO Subclass	0.8596	1747/10285
FoodOn subclass	0.9616	89/6105
Atomic Disjoint	0.9568	210/979
Atomic Subclass	1	117/759
Complex Disjoint	0.7269	144/1824
Complex Subclass	0.9301	63/118

Group 2: DBpedia Datasets The DBpedia experiments (Figure 8.3) provide insights into OCASP/AL's versatility:

- **Atomic Experiments**: OCASP/AL showed near-perfect performance with F1-scores of 0.98 (AD) and 1.00 (AS), and MCC scores of 0.9568 and 1 respectively.
- **Complex Constructs**: In the complex disjoint experiment, OCASP/AL achieved an F1-score of 0.94 and an MCC of 0.7269, indicating good performance but room for improvement.
- **Contrast in Complex Subclass**: The substantial contrast with the negative baseline MCC achieving 0.9301 with OCASP/AL emphasizes the model's capability and adaptability.

Efficiency in Active Learning Table 8.1 provides a detailed look at the efficiency of OCASP/AL in terms of the number of queries made to the oracle. The model demonstrates an impressive balance between query reduction and prediction accuracy:

- **FoodOn Subclass**: OCASP/AL achieved an MCC score of 0.9616 while making only 89 queries out of 6105 total instances, translating to about 1.46% of queries.
- **Atomic Disjoint**: Similarly, for the Atomic Disjoint experiment, the model made 210 queries out of 979 total instances, resulting in an MCC of 0.9568, thus efficiently leveraging the oracle's input.

Critical Evaluation of OCASP/AL and Future Directions

- **Handling Complex Disjoint Axioms**: The Complex Disjoint experiment's MCC score of 0.7269, while commendable, indicates room for improvement. Further refinement in handling complex axioms may lead to even better performance.
- **Efficiency with Small Datasets**: The model's ability to perform well even with small datasets (e.g., Complex Subclass) suggests robustness but also opens avenues to explore optimization in scenarios with limited data availability.

8.6 conclusion

This chapter presented OCASP/AL (**O**WL **C**lass **A**xiom **S**core **P**redictor with **A**ctive **L**earning), a novel approach that enhances expressive schema induction in semantic applications. OCASP/AL generates and scores candidate axioms, leveraging a semantic Web reasoner, a data-driven heuristic, and an extended semantic similarity

for complex class axioms. It offers a more efficient and robust solution compared to existing methods such as OWL2Vec*, Onto2Vec, and OPA2Vec.

The experimental results highlight OCASP/AL's querying efficiency and ability to handle both atomic and complex axioms, with a distinctive capability to minimize oracle interaction without losing accuracy, as demonstrated in the FoodOn Subclass and Atomic Disjoint experiments.

The success of OCASP/AL in predicting axioms not logically deducible from an ontology contributes significantly to bridging the gap between factual data and hand-crafted schemas in knowledge graphs. It offers a promising avenue towards the full exploitation of knowledge graphs' potential, paving the way for more expressive and compliant ontologies across various domains.

Chapter 9

QUANTIFYING SEMANTIC SIMILARITY IN DESCRIPTION LOGIC FORMULAS FOR MACHINE LEARNING-BASED TRUTH LABEL PREDICTION: A MODEL-THEORETIC APPROACH

In the domain of ontology representation and reasoning, accurately determining the semantic similarity between axiomatic formulas and predicting their truth labels is essential. This chapter presents a method based on model-theoretic semantics to compute semantic similarity between axioms described in Description Logic. Using tree structures to represent axioms and strategies reminiscent of the tableaux method, our approach evaluates the co-satisfiability of axiom pairs across various interpretations, represented as ABoxes. Leveraging this computed semantic similarity, we introduce a machine learning model trained to predict the truth labels of unseen formulas. By integrating traditional logic-based reasoning with machine learning techniques, our approach offers a novel perspective on knowledge representation and automated reasoning. Empirical results demonstrate the effectiveness of our method in both semantic similarity computation and truth label prediction.

9.1 Introduction

This chapter introduces a novel method that leverages the principles of model-theoretic semantics to compute semantic similarity between OWL axioms. By representing axioms as tree structures and utilizing strategies reminiscent of the tableaux method, our approach evaluates the co-satisfiability of axiom pairs across various interpretations, represented as ABoxes. Building on this foundation, we further integrate a machine learning model trained on these computed semantic similarities

to predict the truth labels of unseen axioms. This innovative blend of traditional logic-based reasoning with contemporary machine learning techniques offers a new perspective on ontology-based knowledge representation and automated reasoning. In theory, our proposed method works for all DL axiom types, rather all DL formulas.

The empirical results of applying our method underscore its effectiveness not only in accurately computing semantic similarity between OWL axioms but also in predicting their truth labels with high precision. Through this work, we aim to contribute to the advancement of ontology representation and reasoning, paving the way for more sophisticated applications in the semantic web and beyond.

The remainder of this chapter is organized as follows. Section 9.2 provides a short recap on concepts of model-theoretic semantics and Description Logic, with a focus on OWL axioms. In the same section we give a brief review on methods presented in this thesis that influence the method we propose in this chapter. Section 9.3 details the proposed approach for computing semantic similarity between OWL axioms and describes the machine learning model developed for truth label prediction. In Section 9.4, we present the empirical evaluation of our method, including the dataset, experimental setup, and results. Section 9.5 discusses the implications of our findings, explores the limitations of the current study, and suggests avenues for future research. Finally, Section 9.6 concludes the chapter, summarizing the key contributions and highlighting the potential impact of our work on the field of knowledge representation and automated reasoning.

9.2 Recap and Review

Description Logic (DL) serves as the formal foundation for constructing and reasoning about ontologies, encompassing a range of expressivities including quantifier-

s/restrictions, which allow for more complex knowledge representation. More information on DL is provided in Section 2.2.3.

OWL axioms, fundamental to defining relationships and constraints within Semantic Web ontologies, can be quite complex and vary widely in types, such as class axioms and property axioms. These intricacies are explored in Section 2.2.7.

Model-theoretic semantics offer a framework for interpreting the symbols and sentences of a logic system and are directly linked to reasoning methods such as tableaux methods, which are utilized for semantic evaluations. Details on model-theoretic semantics are outlined in Section 2.2.4, with a specific focus on tableaux methods in Section 2.4.4.

In Chapter 4, we present a model-theoretic semantic similarity approach applied to propositional logic, which was found to be challenging to extend to description logic due to its increased complexity. This is mentioned in Section 4.3 .

Further exploration of model-theoretic approaches is discussed in the Related Work chapter, where attempts by Malchiodi et al. to approximate a model-theoretic approach using instance data are covered. Despite their efforts, they were unable to achieve satisfactory results and were limited to atomic axioms of the **subClassOf** type. This limitation is highlighted in Section 3.4.

In Chapter 5, we propose another method to approximate semantic similarity using the distance between concepts in the is-A hierarchy. This method was successfully extended to encompass complex axioms that include negation, union, and intersection of **subClassOf**, **Disjointness**, and **Equivalence** axioms. The details of this extension are elaborated in Chapter 8.

However, this extension method was found to be insufficient as it could not be further extended to include quantifiers/restrictions, which are necessary for a complete description logic evaluation.

9.3 Method

Our method, inspired by the principles of tableaux reasoning, seeks to innovate a similarity measure between formulas by assessing whether the negations of said formulas are satisfiable within a given ABox. Tableaux methods are well-regarded for their effectiveness in checking the satisfiability of logical formulas by systematically attempting to construct a model that satisfies the formula's conditions. By adapting this approach to measure similarity, our method evaluates the extent to which any two formulas, including axioms, share semantic properties by exploring their behavior under diverse interpretations. Using this new similarity, a machine learning method can be trained with a set of labeled formulas and then utilized to predict the label of **any** unseen formula. In this method there would be no need to train multiple model for each type, rather one model for any logical formula.

9.3.1 Model theoretic Semantic Similarity

Theoretically, our approach is designed to capture the similarity between any two logical formulas by examining the consequences of their negations within simulated environments. This section will detail each step of our method, from the initial representation of axioms as logical trees to the computation of similarity scores based on their satisfiability within generated ABoxes. For better comparability we will using OWL Class axioms as our logical formulas.

Process Overview

Following are the steps for our semantic measure calculation.

Axiom Initialization: Each axiom is conceptualized as an instance within the theoretical framework of an *Axiom class*. An axiom is initiated with its literal string

representation, from which a structured parse tree is constructed. This tree serves as a formal representation of the axiom's logical structure, encapsulating the elemental relationships and entities involved.

Furthermore, to facilitate logical assessments, each axiom is also transformed into its negated form, represented similarly as a tree (*negated_tree*). This negation allows for subsequent evaluations of the axiom's satisfiability within various interpretative scenarios, providing a foundation for rigorous logical analysis.

Interpretation Creation: An *Interpretation* object is formulated encompassing a set of axioms. In this object we generate an ABox, which consists of a specified collection of individuals, each assigned to classes and properties. These assignments are not arbitrary but are derived from the logical constituents and relational structures identified within the axioms' trees. The generation of such an ABox is critical as it simulates a potential real-world scenario or "possible world" where the logical validity of axioms can be tested. An interpretation object is created for only one pair of axioms.

Semantic Similarity Calculation: The core of our method lies in the calculation of semantic similarity between axioms, which is achieved by evaluating the satisfiability of their respective negated forms within the context of the generated ABox.

We approach this in two ways, both of which act as an approximation for the model theoretic semantic-based similarity as done in Chapter 4 through sampling. The first way is to generate a specific number of ABoxes with the minimum number of individuals needed (enough to have one for each class in a pair formulas/axioms). This approach aims to check if a pair of axioms are co-satisfied or not in the same k of n worlds. The other approach is to generate one ABox with a specific number of

individuals. This approach aims to check if a pair of axioms share the same number k of combined examples and counterexamples out of n instances.

In theory the similarities from both approaches, should be almost equal. We do not necessarily care about the actual truth of an axiom in any of these ABoxes, rather how similar the axioms are.

This similarity score, therefore, is computed by systematically iterating either through each individual in a large ABox and comparing the satisfiability outcomes for the axioms in question, or through multiple ABoxes and checking if both axioms hold in each. Such a measure not only reflects the logical coherence between the axioms but also their semantic alignment under varied interpretations.

Axiom Representation and Negation

Each axiom is initially represented as a structured logical tree, facilitating the application of logical operations such as negation and the evaluation of satisfiability. Given an axiom represented by the formula ϕ, its tree representation $T(\phi)$ is constructed to reflect the syntactic structure of ϕ.

The negation of the axiom, $\neg\phi$, is then processed to generate a corresponding tree $T(\neg\phi)$. This transformation involves the application of logical negation rules, which for classical logical connectors, are defined as follows:

$$\neg(\neg A) \to A$$

$$\neg(A \wedge B) \to \neg A \vee \neg B$$

$$\neg(A \vee B) \to \neg A \wedge \neg B$$

$$\neg(\forall x\, P(x)) \to \exists x\, \neg P(x)$$

$$\neg(\exists x\, P(x)) \to \forall x\, \neg P(x)$$

Below are the transformations applied to common logical constructs found in ontological axioms:

$$\neg(\text{SubClassOf}(A, B)) \rightarrow \text{ObjectIntersectionOf}(A, \text{ObjectComplementOf}(B))$$

$$\neg(\text{DisjointClasses}(A, B)) \rightarrow \text{ObjectIntersectionOf}(A, B)$$

$$\neg(\text{EquivalentClasses}(A, B)) \rightarrow \text{ObjectUnionOf}(\text{ObjectIntersectionOf}(A, \text{ObjectComplementOf}(B)), \text{ObjectIntersectionOf}(\text{ObjectComplementOf}(A), B))$$

$$\neg(\text{ObjectAllValuesFrom}(P, B)) \rightarrow \text{ObjectSomeValuesFrom}(P, \text{ObjectComplementOf}(B))$$

$$\neg(\text{ObjectSomeValuesFrom}(P, B)) \rightarrow \text{ObjectAllValuesFrom}(P, \text{ObjectComplementOf}(B))$$

$$\neg(\text{ObjectIntersectionOf}(A, B)) \rightarrow \text{ObjectUnionOf}(\text{ObjectComplementOf}(A), \text{ObjectComplementOf}(B))$$

$$\neg(\text{ObjectUnionOf}(A, B)) \rightarrow \text{ObjectIntersectionOf}(\text{ObjectComplementOf}(A), \text{ObjectComplementOf}(B))$$

These negation rules ensure that the negation of complex axioms is handled systematically, preserving the logical integrity while enabling a thorough exploration of their semantic properties within diverse interpretative contexts.

Example Consider the following complex class axiom:

$$\text{SubClassOf}(\text{ObjectIntersectionOf}(\text{ObjectSomeValuesFrom}(\text{hasPart}, \text{Wheel}), \text{ObjectSomeValuesFrom}(\text{hasPart}, \text{Engine})), \text{Vehicle})$$

This axiom asserts that any entity that possesses both a wheel and an engine as parts is considered a subclass of vehicles.

Tree Representation The tree structure for this axiom, denoted as $T(\phi)$, represents the logical construction where both 'ObjectSomeValuesFrom' constraints must be satisfied for the subclass relationship. Figure 9.1 visually represents the structure of $T(\phi)$:=.

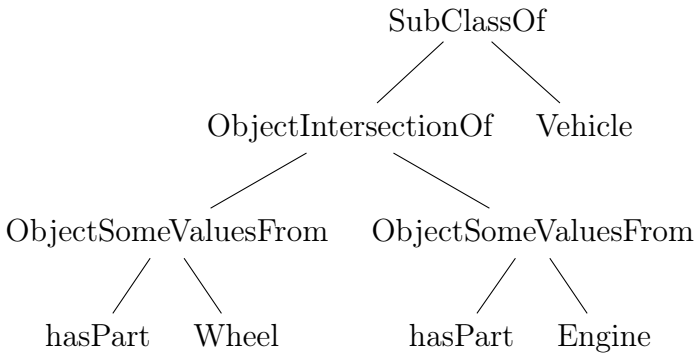


Figure 9.1: Tree representation of the axiom

Negation of the Axiom The negation of this axiom is expressed by inverting the subclass relationship, indicating that there exist instances having both a wheel and an engine that are not vehicles:

$$\neg(\text{SubClassOf}(\text{ObjectIntersectionOf}(\text{ObjectSomeValuesFrom}(\text{hasPart}, \text{Wheel}),$$
$$\text{ObjectSomeValuesFrom}(\text{hasPart}, \text{Engine})), \text{Vehicle}))$$
$$\rightarrow \text{ObjectIntersectionOf}(\text{ObjectIntersectionOf}(\text{ObjectSomeValuesFrom}(\text{hasPart}, \text{Wheel}),$$
$$\text{ObjectSomeValuesFrom}(\text{hasPart}, \text{Engine})), \text{ObjectComplementOf}(\text{Vehicle}))$$

Negated Tree Representation The tree structure for the negated axiom, denoted as $T(\neg\phi)$, is depicted in Figure 9.2. It visually represents the logical structure after negation, highlighting the addition of the "ObjectComplementOf" node to denote entities that are not vehicles:

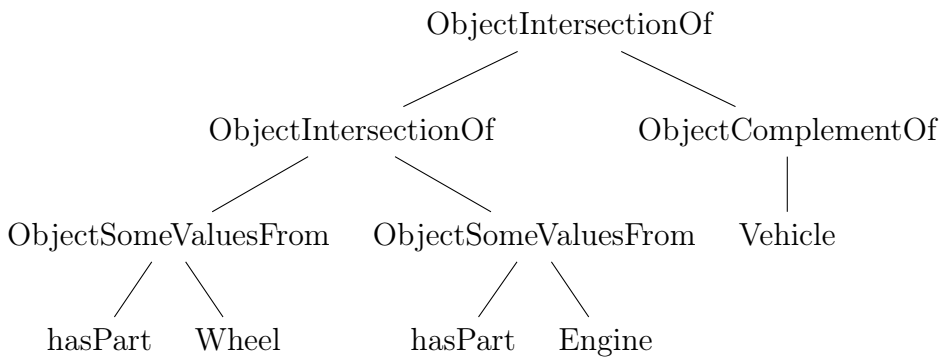


Figure 9.2: Tree representation of the negated axiom

ABox Generation

The ABox (Assertional Box) is the second component of our method, representing a specific state or configuration of the world under which the logical truths of axioms can be tested. The generation of an ABox involves creating a set of individuals and assigning them to various classes and properties derived from a pair of axioms. This simulated environment allows us to test the implications of axioms under different logical scenarios.

Algorithm for ABox Generation The ABox is generated through the following steps, designed to ensure a comprehensive and realistic simulation of possible worlds:

1. **Identify Classes and Properties**: Extract all unique classes and properties from the axioms' tree representations. This is critical as it determines the kinds of entities and relationships that need to be simulated in the ABox.

2. **Create Individuals**: Initialize a predefined number of individuals. Each individual in the ABox represents a potential entity that could exist in the ontology's domain.

3. **Assign Classes Randomly**: Each individual is randomly assigned to classes based on the classes identified from the axioms. This random assignment simulates the variability and diversity of real-world entities. A condition is placed such that each individual has at least one class.

4. **Assign Properties Randomly**: Properties are randomly assigned to individuals, establishing relationships between them. This step involves not only selecting which properties an individual possesses but also linking these properties to other individuals, thereby creating a network of relationships that mirror complex ontological structures.

5. **Axiom Separation**: For the method to work, the ABox generation has to disregard any logical constraints of the axioms, for this everything is done at random. The only role for the axioms in this step is to provide classes and properties, nothing else.

Importance of ABox in Semantic Similarity The generated ABox serves as the testing ground for evaluating axiom satisfiability. We can observing whether an axiom's negated form holds true across various individuals and scenarios within the ABox. The consistency of satisfiability results across multiple ABoxes or with variations within the same ABox can indicate the semantic similarity between different axioms, as axioms that consistently yield similar results under diverse conditions are considered to be semantically closer.

Satisfiability Checking

The satisfiability checking of negated axioms within an ABox is a critical step in determining the semantic similarity between axioms. This process involves evaluating

whether the conditions expressed by the negated axioms are met by the individuals in an ABox. Here, we outline the theoretical foundation and the procedural steps for satisfiability checking.

Theoretical Foundation Satisfiability checking is rooted in model-theoretic semantics, where the goal is to determine if there exists an interpretation under which a given logical formula is true making it a model. In the context of our method, an ABox serves as this interpretation, providing a specific instantiation of individuals, classes, and properties that represent possible real-world entities and their relationships.

Procedure for Satisfiability Checking The procedure for checking the satisfiability of a negated axiom involves the following steps:

1. **Traversal of the ABox**: Each individual in the ABox is examined to determine if it satisfies the conditions specified by the negated axiom. This involves a logical evaluation of the axiom's tree structure against the properties and class memberships of the individual.

2. **Evaluation of Logical Conditions**: The negated tree of the axiom is traversed, and for each node (representing a logical operator or class/property), a check is performed to see if the individual meets the criteria specified by that node. This includes checking class memberships, property relationships, and the application of logical operators such as conjunctions, disjunctions, and negations. Algorithm 4 depicts an implementation of the recursive function responsible for this evaluation.

3. **Returning the Result**: If we are checking for examples and counter examples of the axiom, once all individuals have been evaluated, the results are stored in

a vector. On the other hand if we are checking if an interpretation is a model or counter model, then if at any point any individual contradicts the negated axiom, we return that the interpretation is a model of the axiom. And if we evaluate all the individuals with out running into any contradiction we can say that the interpretation is a counter model. When checking models and counter models we store the results from evaluating an axiom against all generated ABoxes into a vector.

Implications for Semantic Similarity The results of the satisfiability checking process are then used for calculating the semantic similarity between axioms. If two axioms consistently result in similar patterns of satisfiability across multiple ABoxes or within a diverse ABox, it suggests a strong semantic correspondence between them. Conversely, discrepancies in satisfiability results may indicate semantic divergence.

In the case of examples and counterexamples the vectors containing the results of evaluating each of the axioms against all individuals in the ABox are compared. We denote by k the number of instances where an individual is an example or a counterexample for both axioms at the same time. The similarity between the axioms then is k divided by the total number of individuals in the ABox n. This procedure is depicted in Algorithm 5.

In the case where we are checking for models and counter models, the vectors containing the results of evaluating each of the axioms against all generated ABoxes are compared. We denote by k the number of instances where an interpretation is a model or counter model for both axioms at the same time. The similarity between the axioms then is k divided by the total number of generated ABoxes n. This process is depicted in Algorithm 6.

9.3.2 Model

For this method there will be no difference in the creation of the model than in Chapter 6. We have an axiom based vector space and our axiom similarity matrix is produced by invoking either one of Algorithm 5 or Algorithm 6 for every pair of axioms in our set. The set of axioms may be scored in any of the methods mentioned in the previous Chapters.

Regarding Active Learning and Scalability, Since in this method we want to know the raw performance of the new similarity measure we will not be incorporating said techniques.

9.4 Experiment and Result

For this method our experimental goal is simple. We want to evaluate how it performs in predicting labels for a set of unseen complex axioms in a real-world Scenario. For this end, we plan to use DBpedia (Section 3.1.1). We provide two sets of axioms that also include constructs of the type ObjectUnionOf, ObjectIntersectionOf, ObjectAllValuesFrom and ObjectSomeValuesFrom. The sets contain axioms of both subClassOf and disjointWith types. We also created a set of atomic subClassof axioms to test the performance of the method on purely atomic formulas. This is also because we can not evaluate this method with our last method's subClassOf set, since its extremely small with less than 17 positive axioms to test with.

We plan on experimenting with the difference between the model counter model approximation and the example counterexample approximation. We also plan on testing the effect of n the number of ABoxes or individuals generated depending on the case. For more information on n and sampling please refer to Section 4.3.

For all experiments conducted on our new generated sets of axioms, we will be

using a $70-30$ train test split paradigm. We use three different machine learning methods similar to previous chapters, these methods are random forests, support vector machines and K-nearest neighbor.

9.4.1 Experiment: The effect of sample size

For this experiment we will be using our generated axiom sets. We will call them *Complex* – 1, *Complex* – 2 and *Atomic*. *Complex* – 1 is made up of a combination of atomic and complex subClassOf and disjointWith axioms. The total number of axioms in this set is $1,465$ axioms split into 662 disjointness axioms and 803 subsumption axioms. *Complex* – 2 is of a similar composition to *complex* – 1 in terms of types and constructs, but has $1,102$ disjointness axioms and 343 subsumption axioms. *Atomic* on the other hand is made up of purely atomic subsumption axioms numbered 797. We will test the model counter model approach with multiple sampling sizes specifically $10, 100, 1,000$. This experiment will highlight the effect of the sample size on the accuracy of the similarity produced which shall be evident through the model prediction score. Table 9.1 presents the results of the experiment.

9.4.2 Experiment: Model Counter Model VS Example Counterexample

For this part of the experiment, we used the same three sets of axioms but this time calculated the similarities using the example and counterexample approach. For the complex sets we specified the number of samples (individuals generated in one ABox) to $1,000$ to be able to compare with the model and counter model approach. For the *atomic* set we specified $10,000$ samples, so we can check the consistency of how a higher sample size affects the similarity and prediction accuracy. The detailed results of this experiment are found in Table 9.1.

Algorithm 4 Check Satisfiability of a Negated Axiom

Require: *node* (a leaf node or a tree structure of an axiom), *individual* (and individual from an ABox)

Ensure: Returns true if the axiom is satisfiable the individual, else false

```
1: function IsSatisfiable(node, individual)
2:     if node.type is LEAF then
3:         return node.class ∈ individual.classes
4:     else if node.type is COMPLEMENT then
5:         return ¬IsSatisfiable(node.child, individual)
6:     else if node.type is INTERSECTION then
7:         return IsSatisfiable(node.left, individual)
8: ∧ IsSatisfiable(node.right, individual)
9:     else if node.type is UNION then
10:        return IsSatisfiable(node.left, individual)
11: ∨ IsSatisfiable(node.right, individual)
12:    else if node.type is ALLVALUESFROM then
13:        return ∀v ∈ individual.properties[node.property]:
14: IsSatisfiable(node.body, v)
15:    else if node.type is SOMEVALUESFROM then
16:        return ∃v ∈ individual.properties[node.property]:
17: IsSatisfiable(node.body, v)
18:    end if
19: end function
```

Algorithm 5 Compare Two Axioms for Examples and Counterexamples

Require: $axiom1$, $axiom2$ (negated axioms), A (the ABox)

Ensure: Returns the similarity score between two axioms

```
1: function CompareAxioms(axiom1, axiom2, A)
2:     count ← 0
3:     for each individual ∈ A do
4:         result1 ← IsSatisfiable(axiom1, individual)
5:         result2 ← IsSatisfiable(axiom2, individual)
6:         if result1 == result2 then
7:             count ← count + 1
8:         end if
9:     end for
10:    similarity ← count/size of A
11:    return similarity
12: end function
```

Algorithm 6 Compare Two Axioms Across Multiple ABoxes

Require: $axiom1$, $axiom2$ (negated axioms), $\mathcal{A}$ (set of ABoxes)

Ensure: Returns the average similarity score between two axioms across all ABoxes

```
function CompareAxiomsMultiple(axiom1, axiom2, 𝒜)
    count ← 0
    num_aboxes ← size of 𝒜
    for each A ∈ 𝒜 do
        result1 ← True
        result2 ← True
        for each individual ∈ A do
            if IsSatisfiable(axiom1, individual) then
                result1 ← False
            end if
            if IsSatisfiable(axiom2, individual) then
                result2 ← False
            end if
        end for
        if result1 == result2 then
            count ← count + 1
        end if
    end for
    similarity ← count/size of 𝒜
end function
```

Table 9.1: Comprehensive Experimental Results Across Datasets and Sample Sizes

Dataset	**Sample Size**	**Method**	**RF (MCC, F1)**	**KNN (MCC, F1)**	**SVM (MCC, F1)**
Complex-1	1000	Model and Counter Model	(0.58, 0.80)	**(0.68, 0.84)**	(0.58, 0.80)
	100	Model and Counter Model	(0.54, 0.78)	(0.55, 0.78)	(0.59, 0.80)
	10	Model and Counter Model	(0.28, 0.65)	(0.32, 0.66)	NA, 0.50
	1000	Example and Counterexample	**(0.59, 0.82)**	(0.59, 0.82)	**(0.60, 0.82)**
Complex-2	1000	Model and Counter Model	(0.58, 0.80)	**(0.68, 0.84)**	(0.58, 0.80)
	100	Model and Counter Model	(0.32, 0.71)	(0.48, 0.77)	(0.06, 0.65)
	1000	Example and Counterexample	(0.55, 0.80)	(0.50, 0.78)	**(0.56, 0.81)**
Atomic	1000	Model and Counter Model	(0.76, 0.88)	(0.77, 0.88)	(0.71, 0.84)
	10000	Example and Counterexample	(0.76, 0.88)	**(0.80, 0.90)**	(0.71, 0.84)

9.5 Discussion

The experimental evaluation of our method using various datasets and experimental setups offers valuable insights into its performance and applicability in real-world scenarios. This discussion aims to highlight key findings, draw critical observations, and underline the best-performing models based on the results presented in Table 9.1.

9.5.1 Impact of Sample Size

The results from the experiments on different sample sizes reveal significant insights into the scalability and sensitivity of the model. For instance, in the $Complex{-}1$ dataset, as the sample size increases from 10 to $1,000$, there is a noticeable improvement in both MCC and F1 scores across all methods. This trend underscores the importance of a sufficient sample size in achieving reliable and accurate predictions. The diminished performance at a sample size of 10 suggests that the model requires a larger context of individuals to accurately capture and predict the underlying semantic relationships. However, a sample size of 100 seems to offer an acceptable trade off in terms of accuracy and computation time. Considering it is 10 times faster to compute yet at most 0.04 points off in terms of F1 when comparing the best performing model of each experiment.

9.5.2 Comparison of Model Approaches

The experiment comparing the model counter model and the example counterexample approaches demonstrates distinct advantages in certain contexts. For both the $Complex - 1$ and $Complex - 2$ datasets, the example and counterexample approach generally yielded slightly better or comparable results at the highest sample size. However, the highest model performance was achieved with the model counter

model approach with with each dataset when the sample size was the same and the method was KNN. In the case of *Atomic* the example counterexample approach achieved higher performance beating the model counter model but with a 10 times higher number of samples. We believe that the example counterexample is the better approach. Our reasoning behind this is that even when the sample size is the same they can not be equated. The model counter model approach is more computationally complex and this is evident in Algorithm 6. For example, for the model counter model approach if we set the sample number to 100 ABoxes and each ABox contains 10 individuals we would have $1,000$ individuals and the computation time would still be greater than generating one ABox with a sample size of $1,000$ individuals for the example counterexample approach. In this case example counterexample trumps model counter model when the former's sample size is higher in both performance and time, and when the sample sizes are the same performance is comparable and efficiency is a magnitude better.

9.5.3 Performance Across Datasets

Across different datasets, the method demonstrated a consistent level of efficacy. The Atomic dataset, consisting solely of atomic subsumption axioms, showed consistently high performance, highlighting the method's capability in handling simpler logical constructs with high accuracy. Similarly, the more complex datasets, $Complex-1$ and $Complex-2$, were executed with satisfactory performance where F1 score reached as high as 0.84 with no application of any of our feature selection or active learning techniques. This shows that the similarity measure is robust and accurate especially when considering the nature and complexity of axioms processed. We also highlight hat this is a real-world scenario using an ontology with about 760 classes and a training set of about $1,000$ labeled axioms.

9.5.4 Best Performing Models

Notably, the KNN model frequently outperformed the RF and SVM models in terms of MCC and F1 scores in scenarios with higher sample sizes. We believe this is because of the nature of the problem, where at its core a distance/similarity measure is what models the relations. The high performance of the KNN model in the $Complex-1$ dataset at a sample size of $1,000$, achieving an F1 score of 0.84, is particularly noteworthy and suggests that KNN might be especially suited for datasets with mixed axiom types when optimized sample sizes are used.

9.6 Conclusion

This chapter has presented a novel method for computing semantic similarity between description logic axioms using a model-theoretic approach. Our experiments across various datasets demonstrate the method's robustness and effectiveness, particularly when coupled with machine learning techniques for truth label prediction. The empirical results underscore the significance of sample size and the choice of similarity measure approach, highlighting the superior performance of the example and counterexample method in most scenarios. Particularly, the KNN model showed exceptional performance, suggesting that it is well-suited for semantic similarity tasks within complex axiom sets due to its inherent reliance on distance metrics which align well with our semantic similarity calculations.

Further research could explore optimizing the computational efficiency of the model counter model approach or refining the example and counterexample method to reduce sample size requirements without compromising accuracy. Ultimately, the integration of model-theoretic semantics with machine learning opens new avenues for advancing automated reasoning and knowledge representation in ontology-driven sys-

tems, promising significant improvements in semantic web technologies and intelligent information retrieval.

Chapter 10

CONCLUSION

10.1 Contributions

Advancements in the evaluation phase of candidate axioms have long been limited by computational demands and the inefficiency of existing methods. In this thesis, we have developed a novel predictive model that utilizes heuristic-based candidates (constructed and scored) to revolutionize the evaluation phase of candidate axioms. By significantly reducing the computational burden, this model allows for more scalable and practical applications in real-world scenarios. This contribution is thoroughly discussed and experimentally validated in Chapters 5 and 6.

Machine learning and symbolic reasoning are two separate complex yet very compatible disciplines with distinct methods and applications. This thesis bridges this divide by integrating machine learning techniques with traditional symbolic reasoning. This integration aims to drastically increase the efficiency of logical formula evaluations, making it a cornerstone of modern automated reasoning systems. The detailed method and the impact of this integration are explored in Chapters 4 and 9.

The scalability of evaluation methods including traditional reasoners and machine learning models in the context of ontology evaluations has been a persistent challenge. This thesis introduces scalable machine learning models that predict the score and acceptability of candidate OWL class axioms using a vector-space dimension-reduced approach. These models are designed to handle large datasets with little to no compromise in performance, addressing key scalability issues prevalent in previous approaches. The development and effectiveness of these models are extensively

covered in Chapter 7.

Active learning techniques have been notable in refining machine learning models, yet their application in semantic web technologies remains under explored. This thesis implements active learning strategies using extended semantic similarity measures to complex class axioms. This approach significantly enhances model efficiency by selectively querying instances, which is a main concern for managing larger datasets with limited labeled data. The application and benefits of active learning in this context are detailed in Chapter 8.

In the domain of ontology representation and reasoning, accurately determining semantic similarity between axiomatic formulas and predicting their truth labels is essential. This thesis presents a model-theoretic approach that utilizes tree structures to represent complex axioms and employs methods reminiscent of the tableaux method to evaluate their co-satisfiability across various interpretations. With this approach, we contribute a semantic similarity between axioms that is a product of said approach. Leveraging this similarity, a machine learning model is introduced to predict the truth labels of unseen formulas. This innovative blending of traditional logic-based reasoning with advanced machine learning techniques provides a novel perspective on knowledge representation and automated reasoning. The method's effectiveness in both semantic similarity computation and truth label prediction is empirically demonstrated, offering a significant advancement in the field. These contributions are thoroughly discussed and validated in Chapter 9.

10.2 Perspectives

The methods developed in this thesis open several avenues for future research in the field of knowledge representation and automated reasoning. The integration of model-theoretic semantics with machine learning, as detailed in this thesis, lays a

robust foundation for the development of more sophisticated techniques that can further enhance the precision and scalability of semantic similarity assessments and truth label predictions. Future research could explore the application of these methods in more complex scenarios, including dynamic ontologies where axioms change over time, thus requiring adaptive models that can respond to these changes efficiently.

Additionally, the methods in this thesis were tested on class axioms, the similarity computation process in all the models developed guarantees their applicability and extension to property axioms making it the obvious next step to take in future research. Also, the model-theoretic semantics based similarity allows the extension of the method to include constructs other than quantifiers such as number restrictions.

Furthermore, the empirical success of the methods introduced invites the exploration of their application across different domains of knowledge, potentially transforming how knowledge is structured and accessed across various industries. From healthcare to finance, the principles outlined could be adapted to develop domain-specific models that leverage ontology-based reasoning for more accurate data analysis and decision-making processes.

Lastly, the integration of these methods with existing infrastructures presents an exciting challenge. Implementing these advanced reasoning techniques within frameworks that require logical formula evaluation could lead to significant improvements in their efficiency and effectiveness, opening up new possibilities for real-time, informed decision-making. These potential developments represent just the tip of the iceberg in terms of the practical applications of this thesis' findings and highlight the need for ongoing research and collaboration across academic and industry lines.

Bibliography

[1] Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. p. 207–216. SIGMOD '93, Association for Computing Machinery, New York, NY, USA (1993). doi: 10.1145/170035.170072, https://doi.org/10.1145/170035.170072

[2] Ahmadi, N., Lee, J., Papotti, P., Saeed*, M.: Explainable fact checking with probabilistic answer set programming. Truth and Trust Online **abs/1906.09198** (2019)

[3] Akbar, Z., Mustika, H.F., Rini, D.S., Manik, L.P., Indrawati, A., Fefirenta, A.D., Djarwaningsih, T.: An ontology-driven personalized faceted search for exploring knowledge bases of capsicum. Future Internet **13**(7), 172 (2021). doi: 10.3390/fi13070172

[4] Al-Mubaid, H., Nguyen, H.A.: A cluster-based approach for semantic similarity in the biomedical domain. IEEE Engineering in Medicine and Biology **10**(1109), 2713–2717 (2006). doi: 10.1109/IEMBS.2006.259235

[5] Alavi Milani, M.M.R., Hosseinpour, S., Pehlivan, H.: Rule-based production of mathematical expressions. Mathematics **6**(11) (2018). doi: 10.3390/math6110254, https://www.mdpi.com/2227-7390/6/11/254

[6] Aleksander, S.A., Balhoff, J., Carbon, S., Cherry, J.M., Drabkin, H.J., Ebert, D., Feuermann, M., Gaudet, P., Harris, N.L., et al.: The gene ontology knowledgebase in 2023. Genetics **224**(1), iyad031 (2023)

[7] Ali, J., Khan, R., Ahmad, N., Maqsood, I.: Random forests and decision trees. International Journal of Computer Science Issues (IJCSI) **9**(5), 272 (2012)

[8] Ali, W., Saleem, M., Yao, B., Hogan, A., Ngomo, A.N.: A survey of RDF stores & SPARQL engines for querying knowledge graphs. The VLDB Journal (2021). doi: 10.1007/s00778-021-00711-3, https://dx.doi.org/10.1007/s00778-021-00711-3

[9] Allemang, D., Hendler, J.: Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann (2011)

[10] Alola, A.A.: Analysis of possibility theory for reasoning under uncertainty. Ph.D. thesis, Eastern Mediterranean University (EMU) (2012)

[11] Amel, K.R.: From shallow to deep interactions between knowledge representation, reasoning and machine learning. In: Proceedings 13th International Conference Scala Uncertainity Mgmt (SUM 2019), Compiegne, LNCS. pp. 16–18 (2019)

[12] Amgoud, L., David, V.: Measuring similarity between logical arguments. In: Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (2018)

[13] Amgoud, L., David, V.: Measuring similarity between logical arguments. In: Thielscher, M., Toni, F., Wolter, F. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018. pp. 98–107. AAAI Press (2018)

[14] Amir, S., Aït-Kaci, H.: Cedar: efficient reasoning for the semantic web. In: 2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems. pp. 157–163. IEEE (2014)

[15] Andréka, H., Gergely, T., Németi, I.: Model theoretic semantics for many-purpose languages and language hierarchies. In: COLING 1980 Volume 1: The 8th International Conference on Computational Linguistics (1980). doi: 10.3115/990174.990211, https://dx.doi.org/10.3115/990174.990211

[16] Anthony, M., Hammer, P.L.: A Boolean measure of similarity. Discrete Applied Mathematics **154**(16), 2242–2246 (November 2006)

[17] Antoniou, G., Batsakis, S., Mutharaju, R., Pan, J.Z., Qi, G., Tachmazidis, I., Urbani, J., Zhou, Z.: A survey of large-scale reasoning on the web of data. The Knowledge Engineering Review **33**, e21 (2018). doi: 10.1017/S0269888918000255

[18] Antoniou, G., Ghose, A.: What is default reasoning good for? Applications revisited. In: 32nd Annual Hawaii International Conference on System Sciences (HICSS-32), January 5-8, 1999, Maui, Hawaii, USA. IEEE Computer Society (1999)

[19] Apache Jena Project: Apache jena fuseki. https://jena.apache.org/documentation/fuseki2/ (2024)

[20] Arai, N.H., Pitassi, T., Urquhart, A.: The complexity of analytic tableaux. Journal of Symbolic Logic **71**(3), 777–790 (2006). doi: 10.2178/jsl/1154698576

[21] Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., et al.: Gene ontology: tool for the unification of biology. Nature genetics **25**(1), 25–29 (2000)

[22] Ashmore, R., Gurfinkel, A., Trefler, R.: Local reasoning for parameterized first order protocols. In: Badger, J.M., Rozier, K.Y. (eds.) NASA Formal Methods. pp. 36–53. Springer International Publishing, Cham (2019)

[23] Asim, M.N., Wasim, M., Khan, M.U.G., Mahmood, W., Abbasi, H.M.: A survey of ontology learning techniques and applications. Database **2018** (2018). doi: 10.1093/database/bay101

[24] Baader, F., Küsters, R., Wolter, F.: Extensions to Description Logics, chap. 6, pp. 237–282. Cambridge University Press (2007). doi: 10.1017/CBO9780511711787.008

[25] Baader, F., Horrocks, I., Lutz, C., Sattler, U.: Complexity, chap. 5. Cambridge University Press (2017). doi: 10.1017/9781139025355.005

[26] Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An Introduction to Description Logic. Cambridge University Press (2017). doi: 10.1017/9781139025355

[27] Baader, F., Horrocks, I., Lutz, C., Sattler, U.: Reasoning in DLs with Tableau Algorithms, p. 69–105. Cambridge University Press (2017). doi: 10.1017/9781139025355.004

[28] Baader, F., Sattler, U.: An overview of tableau algorithms for description logics. Studia Logica **69**, 5–40 (2001)

[29] Badreddine, S., d'Avila Garcez, A., Serafini, L., Spranger, M.: Logic tensor networks. Artif. Intell. **303**, 103649 (2022). doi: 10.1016/j.artint.2021.103649, https://doi.org/10.1016/j.artint.2021.103649

[30] Bai, J., Wang, Y., Zheng, T., Guo, Y., Liu, X., Song, Y.: Abductive logical reasoning on knowledge graphs (2023)

[31] Balbiani, P., Tinchev, T.: Decidable and undecidable problems for first-order definability and modal definability. In: Özgün, A., Zinova, Y. (eds.) Language, Logic, and Computation. pp. 214–236. Springer International Publishing, Cham (2022)

[32] Ballatore, A., Bertolotto, M., Wilson, D.C.: An evaluative baseline for geo-semantic relatedness and similarity. GeoInformatica **18**(4), 747–767 (2014). doi: 10.1007/s10707-013-0

[33] Ballout, A., da Costa Pereira, C., Tettamanzi, A.G.B.: Learning to classify logical formulas based on their semantic similarity. In: Aydogan, R., Criado, N., Lang, J., Sánchez-Anguix, V., Serramia, M. (eds.) PRIMA 2022: Principles and Practice of Multi-Agent Systems - 24th International Conference, Valencia, Spain, November 16-18, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13753, pp. 364–380. Springer (2022). doi: 10.1007/978-3-031-21203-1_22

[34] Ballout, A., da Costa Pereira, C., Tettamanzi, A.: Scalable prediction of atomic candidate OWL class axioms using a vector-space dimension reduced approach. In: Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART. pp. 347–357. INSTICC, SciTePress (2024). doi: 10.5220/0012384200003636

[35] Ballout, A., da Costa Pereira, C., Tettamanzi, A.G.B.: Predicting the acceptability of atomic candidate OWL class axioms. In: 2023 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). pp. 339–345 (2023). doi: 10.1109/WI-IAT59888.2023.00055

[36] Ballout, A., Tettamanzi, A.G.B., Da Costa Pereira, C.: Predicting the score of atomic candidate OWL class axioms. In: 2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). pp. 72–79 (2022). doi: 10.1109/WI-IAT55865.2022.00020

[37] Banu, A., Fatima, S.S., Khan, K.U.R.: A survey and comparison of wordnet based semantic similarity measures. International Journal of Computer Science And Technology pp. 456–461 (2013)

[38] Batyrshin, I.: Towards a general theory of similarity and association measures: similarity, dissimilarity and correlation functions. Journal of Intelligent & Fuzzy Systems **36**(4), 2977–3004 (2019)

[39] Ben-Ari, M.: First-Order Logic: Undecidability and Model Theory, pp. 223–230. Springer London, London (2012). doi: 10.1007/978-1-4471-4129-7_12, https://doi.org/10.1007/978-1-4471-4129-7_12

[40] Benito-Monsalvo, C.: Local applications of logics via model-theoretic interpretations. Logic and Logical Philosophy **31**(4), 535–556 (Jul 2022). doi: 10.12775/LLP.2022.023, https://apcz.umk.pl/LLP/article/view/38400

[41] Beraha, M., Metelli, A.M., Papini, M., Tirinzoni, A., Restelli, M.: Feature selection via mutual information: New theoretical insights. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–9 (2019). doi: 10.1109/IJCNN.2019.8852410

[42] Berners-Lee, T.: Linked data (2006), https://www.w3.org/DesignIssues/LinkedData.html

[43] Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5), 34–43 (2001)

[44] Berry, S.: A Logical Foundation for Potentialist Set Theory. Cambridge University Press (2022). doi: 10.1017/9781108992756

[45] Bistarelli, S., Taticchi, C.: A labelling semantics and strong admissibility for weighted argumentation frameworks. Journal of Logic and Computation **32**(2), 281–306 (2022). doi: 10.1093/logcom/exab085

[46] Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far, p. 115–143. Association for Computing Machinery, New York, NY, USA, 1 edn. (2023), https://doi.org/10.1145/3591366.3591378

[47] Blee, J., Billington, D., Sattar, A.: Reasoning with levels of modalities in BDI logic. In: Ghose, A.K., Governatori, G., Sadananda, R. (eds.) Agent Computing and Multi-Agent Systems, 10th Pacific Rim International Conference on Multi-Agents, PRIMA 2007, Bangkok, Thailand, November 21-23, 2007. Revised Papers. Lecture Notes in Computer Science, vol. 5044, pp. 410–415. Springer (2007)

[48] Boisen, S.: NT Names Documentation. http://www.semanticbible.com/ntn/ntn-documentation.html (2006)

[49] Bowles, G.: Propositional relevance. Informal Logic **2**(12), 65–77 (Spring 1990)

[50] Brodt, A., Nicklas, D., Mitschang, B.: Deep integration of spatial query processing into native RDF triple stores. In: ACM Digital Library (2010). doi: 10.1145/1869790.1869799, https://dx.doi.org/10.1145/1869790.1869799

[51] Bühmann, L., Lehmann, J., Westphal, P.: DL-Learner -a framework for inductive learning on the semantic web. Journal of Web Semantics **39**, 15–24 (2016), https://doi.org/10.1016/j.websem.2016.06.001

[52] Bühmann, L., Lehmann, J.: Universal OWL axiom enrichment for large knowledge bases. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) Knowledge Engineering and Knowledge Management. pp. 57–71. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[53] Caferra, R.: Logic for Computer Science and Artificial Intelligence. John Wiley & Sons, Ltd (03 2013). doi: 10.1002/9781118604182

[54] Cai, J., Luo, J., Wang, S., Yang, S.: Feature selection in machine learning: A new perspective. Neurocomputing **300**, 70–79 (7 2018). doi: 10.1016/J.NEUCOM.2017.11.077

[55] Cai, W., Zhang, Y., Zhou, J.: Maximizing expected model change for active learning in regression. In: 2013 IEEE 13th International Conference on Data Mining. pp. 51–60 (2013). doi: 10.1109/ICDM.2013.104

[56] Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R.: Linking data to ontologies: The description logic dl-lite_a. In: Grau, B.C., Hitzler, P., Shankey, C., Wallace, E. (eds.) Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006. CEUR Workshop Proceedings, vol. 216. CEUR-WS.org (2006), https://ceur-ws.org/Vol-216/submission_23.pdf

[57] Cambridge Semantics: RDFS vs. OWL (2024), https://cambridgesemantics.com/blog/semantic-university/learn-owl-rdfs/rdfs-vs-owl/

[58] Cappé, O., Moulines, E., Rydén, T.: Inference in hidden markov models. In: Proceedings of EUSFLAT conference. pp. 14–16 (2009)

[59] Carral, D., Zalewski, J., Hitzler, P.: An efficient algorithm for reasoning over OWL EL ontologies with nominal schemas. Journal of Logic and Computation **33**(1), 136–162 (05 2022). doi: 10.1093/logcom/exac032, https://doi.org/10.1093/logcom/exac032

[60] Carriero, V.A., Gangemi, A., Mancinelli, M.L., Marinucci, L., Nuzzolese, A.G., Presutti, V., Veninata, C.: ArCo: the italian cultural heritage knowledge graph. In: ESWC 2019: The Semantic Web (2019). doi: 10.1007/978-3-030-30796-7_3, https://dx.doi.org/10.1007/978-3-030-30796-7_3

[61] Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: Implementing the semantic web recommendations. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters. p. 74–83. WWW Alt. '04, Association for Computing Machinery, New York, NY, USA (2004). doi: 10.1145/1013367.1013381, https://doi.org/10.1145/1013367.1013381

[62] Cerrito, S., Popescu, A.: Automated Reasoning with Analytic Tableaux and Related Methods: 28th International Conference, TABLEAUX 2019, London, UK, September 3-5, 2019, Proceedings, vol. 11714. Springer Nature (2019)

[63] Cerutti, F., Kaplan, L., Kimmig, A., Şensoy, M.: Probabilistic logic programming with beta-distributed random variables. In: Proceedings of the AAAI Conference on Artificial Intelligence. AAAI'19/IAAI'19/EAAI'19, vol. 33, pp. 7769–7776. AAAI Press (2019). doi: 10.1609/aaai.v33i01.33017769, https://doi.org/10.1609/aaai.v33i01.33017769

[64] Cetinkaya, A.: Regular expression generation through grammatical evolution. In: Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation. p. 2643–2646. GECCO '07, Association for Computing Machinery, New York, NY, USA (2007). doi: 10.1145/1274000.1274089, https://doi.org/10.1145/1274000.1274089

[65] Chaki, J.: A Fuzzy Logic-Based Approach to Handle Uncertainty in Artificial Intelligence, pp. 47–69. Springer Nature Singapore, Singapore (2023). doi: 10.1007/978-981-99-5333-2_5, https://doi.org/10.1007/978-981-99-5333-2_5

[66] Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & Electrical Engineering **40**(1), 16–28 (2014)

[67] Chang, C.: Symbolic Logic and Mechanical Theorem Proving. Academic Press, San Diego (1973)

[68] Chen, H., Chen, X., Gu, P., Wu, Z., Yu, T., et al.: OWL reasoning framework over big biological knowledge network. BioMed research international **2014** (2014)

[69] Chen, J., He, Y., Geng, Y., Jiménez-Ruiz, E., Dong, H., Horrocks, I.: Contextual semantic embeddings for ontology subsumption prediction. World Wide Web pp. 1–23 (2023)

[70] Chen, J., Hu, P., Jimenez-Ruiz, E., Holter, O.M., Antonyrajah, D., Horrocks, I.: Owl2vec*: embedding of OWL ontologies. Machine Learning **110** (2021). doi: 10.1007/s10994-021-05997-6

[71] Chen, Z., Wang, Y., Zhao, B., Cheng, J., Zhao, X., Duan, Z.: Knowledge graph completion: A review. Ieee Access **8**, 192435–192456 (2020)

[72] Cheng, K., Liu, J., Wang, W., Sun, Y.: Rlogic: Recursive logical rule learning from knowledge graphs. In: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2022). doi: 10.1145/3534678.3539421

[73] Cheruvu, A., Radhakrishna, V.: A survey of similarity measures for time stamped temporal datasets. In: DATA. pp. 193–197. ACM (2021)

[74] Cheung, K., Drennan, J., Hunter, J.: Towards an ontology for data-driven discovery of new materials. In: AAAI Spring Symposium: Semantic Scientific Knowledge Integration. pp. 9–14 (2008)

[75] Chicco, D., Jurman, G.: The advantages of the Matthews correlation coefficient (mcc) over F1 score and accuracy in binary classification evaluation. BMC genomics **21**(1), 1–13 (2020)

[76] Contributors, P.W.: Protege4pizzas10minutes. `https://protegewiki.stanford.edu/wiki/Protege4Pizzas10Minutes` (2021), accessed: 2024-01-30

[77] Cook, S.A.: The complexity of theorem-proving procedures. In: Conference on Computational Complexity. IEEE (1971)

[78] Corby, O., Dieng-Kuntz, R., Faron-Zucker, C.: Querying the semantic web with corese search engine. In: European conference on artificial intelligence. pp. 705–709 (2004)

[79] Corby, O., Dieng-Kuntz, R., Faron Zucker, C., Gandon, F.: Ontology-based Approximate Query Processing for Searching the Semantic Web with Corese. Ph.D. thesis, INRIA (2006)

[80] Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., Gandon, F.: Searching the semantic web: Approximate query processing based on ontologies. IEEE Intelligent Systems **21**(1), 20–27 (January 2006), `https://doi.org/10.1109/MIS.2006.16`

[81] Cordón, O., Herrera, F., Villar, P.: Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. IEEE Transactions on Fuzzy Systems **9**(4), 667–674 (2001). doi: 10.1109/91.940977, `https://dx.doi.org/10.1109/91.940977`

[82] Corradi, A., Foschini, L., Zanni, A., Casoni, M., Monti, S., Sprotetto, F.: A federation model to support semantic SPARQL queries for enterprise data governance. In: 2016 11th International Conference on Digital Information Management (ICDIM) (2016). doi: 10.1109/ICDIM.2016.7829778, `https://dx.doi.org/10.1109/ICDIM.2016.7829778`

[83] Costa, D., Martins, M., Marcos, J.: Two versions of Herbrand's theorem for hybrid logic. Fundamenta Informaticae (2019), https://dblp.org/rec/journals/flap/CostaMM19.html

[84] Costa, R., Figueiras, P., Paiva, L., Jardim-Gonçalves, R., Lima, C.: Capturing knowledge representations using semantic relationships. In: The Sixth International Conference on Advances in Semantic Processing. pp. 75–81 (01 2012)

[85] Craswell, N.: Mean Reciprocal Rank, pp. 1703–1703. Springer US, Boston, MA (2009). doi: 10.1007/978-0-387-39940-9_488, https://doi.org/10.1007/978-0-387-39940-9_488

[86] Crouse, M., Abdelaziz, I., Cornelio, C., Thost, V., Wu, L., Forbus, K., Fokoue, A.: Improving graph neural network representations of logical formulae with subgraph pooling. arXiv preprint arXiv:1911.06904 (2019)

[87] Cullen, J., Bryman, A.: The knowledge acquisition bottleneck: Time for reassessment? Expert Systems **5**, 216–225 (8 1988). doi: 10.1111/J.1468-0394.1988.TB00065.X

[88] Cuong, B.C., Anh, T.H., Hai, B.D.: Some operations on type-2 intuitionistic fuzzy sets. Journal of Computer Science and Cybernetics **28**(3), 274–283 (2012)

[89] Cyganiak, R., Wood, D., Lanthaler, M.: RDF 1.1 Concepts and Abstract Syntax (Feb 2014), http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[90] D'Agostino, M., Gabbay, D.M., Hähnle, R., Posegga, J.: Handbook of tableau methods. Springer Science & Business Media (2013)

[91] D'Aquin, M., Bunoiu, R., Cirstea, H., Lenczner, M., Lieber, J., Zamkotsian, F.: Combining representation formalisms for reasoning upon mathematical knowledge. In: Proceedings of the 12th Knowledge Capture Conference 2023. p. 180–187. K-CAP '23, Association for Computing Machinery, New York, NY, USA (2023). doi: 10.1145/3587259.3627549, https://doi.org/10.1145/3587259.3627549

[92] DBpedia Association: Dbpedia ontology documentation. https://www.dbpedia.org/resources/ontology/ (2024)

[93] De Giacomo, G., Lenzerini, M.: Tbox and ABox reasoning in expressive description logics. KR **96**, 316–327 (1996)

[94] Dean, M., Schreiber, A., Bechofer, S., Harmelen, F.V., Hendler, J., Horrocks, I., MacGuinness, D., Patel-Schneider, P., Stein, L.: OWL web ontology language - reference. World Wide Web Consortium (W3C)

[95] Demri, S., Goranko, V., Lange, M.: Tableaux-Based Decision Methods, p. 476–542. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press (2016). doi: 10.1017/CBO9781139236119.013

[96] Dentler, K., Cornet, R., Ten Teije, A., De Keizer, N.: Comparison of reasoners for large ontologies in the OWL 2 EL profile. Semantic Web **2**(2), 71–87 (2011)

[97] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: North American Chapter of the Association for Computational Linguistics (2019), https://api.semanticscholar.org/CorpusID:52967399

[98] Diehl, A.D., Meehan, T.F., Bradford, Y.M., Brush, M.H., Dahdul, W.M., Dougall, D.S., He, Y., Osumi-Sutherland, D., Ruttenberg, A., Sarntivijai, S., et al.: The cell ontology 2016: enhanced content, modularization, and ontology interoperability. Journal of biomedical semantics **7**, 1–10 (2016)

[99] Ding, H., Takigawa, I., Mamitsuka, H., Zhu, S.: Similarity-based machine learning methods for predicting drug–target interactions: a brief review. Briefings in Bioinformatics **15**(5), 734–747 (08 2013). doi: 10.1093/bib/bbt056, https://doi.org/10.1093/bib/bbt056

[100] Dodge, Y.: Coefficient of Determination, pp. 88–91. Springer New York, New York, NY (2008). doi: 10.1007/978-0-387-32833-1_62, https://doi.org/10.1007/978-0-387-32833-1_62

[101] Dodge, Y.: Mean Squared Error, pp. 337–339. Springer New York, New York, NY (2008). doi: 10.1007/978-0-387-32833-1_251, https://doi.org/10.1007/978-0-387-32833-1_251

[102] Dooley, D.M., Griffiths, E.J., Gosal, G.S., Buttigieg, P.L., Hoehndorf, R., Lange, M.C., Schriml, L.M., Brinkman, F.S., Hsiao, W.W.: Foodon: a harmonized food ontology to increase global food traceability, quality control and data integration. npj Science of Food **2**(1), 23 (2018)

[103] Duan, S., Jiang, S., Dai, H., Wang, L., He, Z.: The applications of hybrid approach combining exact method and evolutionary algorithm in combinatorial optimization. Journal of Computational Design and Engineering **10**(3), 934–946 (04 2023). doi: 10.1093/jcde/qwad029, https://doi.org/10.1093/jcde/qwad029

[104] Dubois, D., Prade, H.: Possibility Theory—An Approach to Computerized Processing of Uncertainty. Plenum Press, New York (1988)

[105] Dubois, D., Prade, H.: Fuzzy set and possibility theory-based methods in artificial intelligence. Artificial Intelligence **148**(1-2), 1–9 (2003)

[106] Dubois, D., Prade, H.: Possibility theory: an approach to computerized processing of uncertainty. Springer Science & Business Media (2012)

[107] Duff, I.S., Erisman, A.M., Reid, J.K.: Direct Methods for Sparse Matrices. Oxford University Press (01 2017). doi: 10.1093/acprof:oso/9780198508380.001.0001, https://doi.org/10.1093/acprof:oso/9780198508380.001.0001

[108] Dörpinghaus, J., Klante, S., Christian, M., Meigen, C., Düing, C.: From social networks to knowledge graphs: A plea for interdisciplinary approaches. Social Sciences & Humanities Open **6**(1) (2022). doi: https://doi.org/10.1016/j.ssaho.2022.100337, https://www.sciencedirect.com/science/article/pii/S2590291122000912

[109] D'Mello, R., Melcher, J., Torous, J.: Similarity matrix-based anomaly detection for clinical intervention. Scientific Reports **12**(1), 9162 (2022)

[110] Eberhart, A., Shimizu, C., Chowdhury, S., Sarker, M.K., Hitzler, P.: Expressibility of OWL axioms with patterns. In: ESWC 2021 (2021), https://daselab.cs.ksu.edu/publications/expressibility-owl-axioms-patterns

[111] Ebner, G.: Herbrand constructivization for automated intuitionistic theorem proving. In: Automated Reasoning (2019). doi: 10.1007/978-3-030-29026-9_20, https://dx.doi.org/10.1007/978-3-030-29026-9_20

[112] Elaine Rich, Alan Kaylor Cline: Reasoning: An Introduction to Logic, Sets, and Functions (2023), https://www.cs.utexas.edu

[113] Encyclopedia of Mathematics: Logical formula (2023), https://encyclopediaofmath.org/wiki/Logical_formula

[114] Erdogdu, U., Tan, M., Alhajj, R., Polat, F., Rokne, J., Demetrick, D.: Integrating machine learning techniques into robust data enrichment approach and its application to gene expression data. International Journal of Data Mining and Bioinformatics **8**(3), 247–281 (2013). doi: 10.1504/IJDMB.2013.056090, https://www.inderscienceonline.com/doi/abs/10.1504/IJDMB.2013.056090, pMID: 56090

[115] Erling, O., Mikhailov, I.: Virtuoso: RDF Support in a Native RDBMS, pp. 501–519. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). doi: 10.1007/978-3-642-04329-1_21, https://doi.org/10.1007/978-3-642-04329-1_21

[116] Esteva, F., Godo, L., Rodríguez, R.O., Vetterlein, T.: On Ruspini's models of similarity-based approximate reasoning. In: IPMU (1). Communications in Computer and Information Science, vol. 1237, pp. 3–13. Springer (2020)

[117] Etchemendy, J., Barker-Plummer, D.: Semantics of First-Order Logic (2024), https://www.edx.org/course/semantics-of-first-order-logic

[118] Ewald, W.: The Emergence of First-Order Logic. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, Spring 2019 edn. (2019)

[119] Fanizzi, N., d'Amato, C., Esposito, F.: Dl-foil concept learning in description logics. In: Inductive Logic Programming: 18th International Conference, ILP 2008 Prague, Czech Republic, September 10-12, 2008 Proceedings 18. pp. 107–121. Springer (2008)

[120] Faraj, G., Micsik, A.: Representing and validating cultural heritage knowledge graphs in cidoc-crm ontology. Future Internet **13**(11), 277 (2021). doi: 10.3390/fi13110277

[121] Farias, T.M., Dessimoz, C., Ayllón-Benítez, A., Yang, C.C., Long, J., Sima, A.: Federating and querying heterogeneous and distributed Web APIs and triple stores. Zenodo (2022). doi: 10.5281/zenodo.6952236, https://dx.doi.org/10.5281/zenodo.6952236

[122] Felin, R., Tettamanzi, A.G.B.: Using grammar-based genetic programming for mining subsumption axioms involving complex class expressions. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. p. 234–240. WI-IAT '21, Association for Computing Machinery, New York, NY, USA (2022). doi: 10.1145/3486622.3494025, https://doi.org/10.1145/3486622.3494025

[123] Felin, R., Corby, O., Faron, C., Tettamanzi, A.G.B.: Optimizing the computation of a possibilistic heuristic to test OWL subclassof axioms against RDF data. In: W-IAT 2022 - IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. Niagara Falls, Canada (2022). doi: 10.1109/WI-IAT55865.2022.00021

[124] Feller, W.: An Introduction to Probability Theory and Its Applications, Volume 2. No. v. 1-2 in An Introduction to Probability Theory and Its Applications, Wiley (1957), https://books.google.com.lb/books?id=G4JztQAACAAJ

[125] Fitting, M.: Tableau methods of proof for modal logics. Notre Dame Journal of Formal Logic **13**(2), 237–247 (1972)

[126] Fišer, P., Kubalík, P., Kubátová, H.: Output grouping method based on a similarity of Boolean functions. In: Proc. of 7th Int. Workshop on Boolean Problems (IWSBP), Freiberg (Germany), September 21–22. pp. 107–113 (2006)

[127] Fleischhacker, D., Völker, J.: Inductive learning of disjointness axioms. In: On the Move to Meaningful Internet Systems: OTM 2011: Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011, Hersonissos, Crete, Greece, October 17-21, 2011, Proceedings, Part II. pp. 680–697. Springer (2011)

[128] Fleischhacker, D., Völker, J., Stuckenschmidt, H.: Mining RDF data for property axioms. In: On the Move to Meaningful Internet Systems: OTM 2012: Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part II. pp. 718–735. Springer (2012)

[129] Foundry, O.: Cell ontology. https://obofoundry.org/ontology/cl.html (2022), accessed: 2023-09-22

[130] From, A.H., Lund, S.T., Villadsen, J.: A case study in computer-assisted metareasoning. In: González, S.R., Machado, J.M., González-Briones, A., Wikarek,

J., Loukanova, R., Katranas, G., Casado-Vara, R. (eds.) Distributed Computing and Artificial Intelligence, Volume 2: Special Sessions 18th International Conference. pp. 53–63. Springer International Publishing, Cham (2022)

[131] From, A.H., Villadsen, J.: A naive prover for first-order logic: A minimal example of analytic completeness. In: Ramanayake, R., Urban, J. (eds.) Automated Reasoning with Analytic Tableaux and Related Methods. pp. 468–480. Springer Nature Switzerland, Cham (2023)

[132] Fuchs, N.E.: Knowledge representation and reasoning in (controlled) natural language. In: Dau, F., Mugnier, M.L., Stumme, G. (eds.) Conceptual Structures: Common Semantics for Sharing Knowledge. pp. 51–51. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

[133] Gandon, F.: Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web. Ph.D. thesis, Université Nice Sophia Antipolis (2002)

[134] Garg, H., Deng, Y., Jeng, J.T. (eds.): International Journal of Fuzzy Systems - Volume 25, Issue 8, vol. 25. Springer (2023), https://link.springer.com/journal/40815/volumes-and-issues/25-8

[135] Gelfond, M., Kahl, Y.: Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach. Cambridge University Press (2014)

[136] Georgieva-Trifonova, T., Galabov, M.: Transforming 3d models to semantic web representation. Romanian Journal of Information Science and Technology (2023). doi: 10.59277/romjist.2023.1.03, https://dx.doi.org/10.59277/romjist.2023.1.03

[137] Giordano, L., Gliozzi, V., Theseider Dupré, D.: A conditional, a fuzzy and a probabilistic interpretation of self-organizing maps. Journal of Logic and Computation **32**(2), 178–205 (2022). doi: 10.1093/logcom/exab082

[138] Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: an OWL 2 reasoner. Journal of automated reasoning **53**, 245–269 (2014)

[139] Glimm, B., Kazakov, Y.: Classical algorithms for reasoning and explanation in description logics. Reasoning Web. Explainable Artificial Intelligence: 15th International Summer School 2019, Bolzano, Italy, September 20–24, 2019, Tutorial Lectures pp. 1–64 (2019)

[140] Goodman, S.N.: Toward evidence-based medical statistics. 2: The bayes factor. Annals of internal medicine **130**(12), 1005–1013 (1999)

[141] Grilletti, G.: Disjunction and existence properties in inquisitive first-order logic. Studia Logica **107**(2), 251–282 (2018). doi: 10.1007/S11225-018-9835-3, https://dx.doi.org/10.1007/S11225-018-9835-3

[142] Grindrod CBE, P.: 12 - Similarity, graphs and networks, random matrices, and SVD. In: Mathematical Underpinnings of Analytics: Theory and Applications. Oxford University Press (11 2014). doi: 10.1093/acprof:oso/9780198725091.003.0002, https://doi.org/10.1093/acprof:oso/9780198725091.003.0002

[143] Group, W.O.W.: Sroiq. https://www.w3.org/2007/OWL/wiki/SROIQ (2007)

[144] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016). doi: 10.1145/2939672.2939754

[145] Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? International journal of human-computer studies **43**(5-6), 907–928 (1995)

[146] Guo, P., Cheng, W., Wang, Y.: Hybrid evolutionary algorithm with extreme machine learning fitness function evaluation for two-stage capacitated facility location problems. Expert Systems with Applications **71**, 57–68 (2017). doi: https://doi.org/10.1016/j.eswa.2016.11.025, https://www.sciencedirect.com/science/article/pii/S0957417416306601

[147] Gupta, R., Malik, S.K.: RDF-ML: A proposed SPARQL tool for machine learning on semantic web data. In: ACM Digital Library (2022). doi: 10.1145/3590837.3590944, https://dx.doi.org/10.1145/3590837.3590944

[148] Géron, A.: Ensemble Learning and Random Forests, chap. 7. O'Reilly Media, Inc., 2 edn. (2019), https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ch07.html

[149] Géron, A.: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, chap. 10. O'Reilly Media, Inc., 2 edn. (2019), https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ch10.html

[150] Géron, A.: The Machine Learning Landscape, chap. 1. O'Reilly Media, Inc., 2 edn. (2019), https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ch01.html, subsection: 'Hyperparameter Tuning and Model Selection' under section: 'Testing and Validating'

[151] Géron, A.: Training Models, chap. 4. O'Reilly Media, Inc., 2 edn. (2019), https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ch04.html, focus on Linear Regression

[152] Géron, A.: Dimensionality Reduction, chap. 8. O'Reilly Media, Inc., 2 edn. (2020), https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ch08.html

[153] Haase, P., Völker, J.: Ontology learning and reasoning - Dealing with uncertainty and inconsistency. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 5327 LNAI, pp. 366–384 (2008). doi: 10.1007/978-3-540-89765-1-21

[154] Hadzic, M., Wongthongtham, P., Dillon, T., Chang, E., Hadzic, M., Wongthongtham, P., Dillon, T., Chang, E.: Introduction to ontology. Ontology-based multi-agent systems pp. 37–60 (2009)

[155] Hanea, A.M., Nane, G.F.: Multivariate probabilistic modelling for risk and decision analysis (2022)

[156] Harispe, S., Ranwez, S., Janaqi, S., Montmain, J.: Semantic similarity from natural language and ontology analysis. Springer (2015)

[157] Harris, S., Seaborne, A.: SPARQL 1.1 query language. W3c recommendation, World Wide Web Consortium (2013), https://www.w3.org/TR/sparql11-query/, published on 21 March 2013

[158] Harrison, J.: Handbook of Practical Logic and Automated Reasoning. Cambridge University Press (2009). doi: 10.1017/CBO9780511576430, https://www.cambridge.org/core/books/handbook-of-practical-logic-and-automated-reasoning/propositional-logic/7C1B557BE10D2F2C4B8E8C3F4724F7FC

[159] Hassanpour, S., O'Connor, M.J., Das, A.K.: Clustering rule bases using ontology-based similarity measures. Journal of Web Semantics **25**, 1–8 (2014). doi: 10.1016/j.websem.2014.03.001

[160] Havelund, K., Peled, D.: The syntax, semantics, and monitoring algorithms for both propositional and first-order temporal logics. Formal Aspects of Computing **30**(6), 617–647 (2018). doi: 10.1007/978-3-030-03769-7_7, https://dx.doi.org/10.1007/978-3-030-03769-7_7

[161] Hawthorne, F., Mills, S., Hatert, F., Rumsey, M.: Ontology, archetypes and the definition of 'mineral species'. Mineralogical Magazine **85**(2), 125–133 (2021). doi: 10.1180/mgm.2021.21, https://doi.org/10.1180/mgm.2021.21

[162] Heflin, J., Pan, Z.: A model theoretic semantics for ontology. In: The Semantic Web – ISWC 2004. pp. 62–76 (2004). doi: 10.1007/978-3-540-30475-3_6, https://dx.doi.org/10.1007/978-3-540-30475-3_6

[163] Hinrichs, T.: Herbrand logic – first-order syntax and herbrand semantics. https://www.cs.uic.edu/~hinrichs/herbrand/html/herbrandlogic.html

[164] Ho, V.T., Stepanova, D., Gad-Elrab, M.H., Kharlamov, E., Weikum, G.: Rule learning from knowledge graphs guided by embedding models. In: The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part I 17. pp. 72–90. Springer (2018)

[165] Hofer, M., Obraczka, D., Saeedi, A., Köpcke, H., Rahm, E.: Construction of knowledge graphs: State and challenges. arXiv preprint arXiv:2302.11509 (2023)

[166] Hofer, M., Obraczka, D., Saeedi, A., Köpcke, H., Rahm, E.: Construction of knowledge graphs: State and challenges. arXiv preprint arXiv:2302.11509 (2023)

[167] Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Gayo, J.E.L., Kirrane, S., Neumaier, S., Polleres, A., et al.: Knowledge graphs. preprint arXiv:2003.02320 (2020)

[168] Hohenecker, P., Lukasiewicz, T.: Ontology reasoning with deep neural networks. J. Artif. Intell. Res. **68**, 503–540 (2020). doi: 10.1613/jair.1.11661

[169] Holden, S.B., et al.: Machine learning for automated theorem proving: Learning to solve sat and qsat. Foundations and Trends® in Machine Learning **14**(6), 807–989 (2021)

[170] Holter, O.M., Myklebust, E.B., Chen, J., Jimenez-Ruiz, E.: Embedding OWL ontologies with owl2vec. CEUR Workshop Proceedings **2456**, 33–36 (January 2019), http://ceur-ws.org/Vol-2456/

[171] Horrocks, I.: Implementation and optimization techniques. In: The description logic handbook: theory, implementation, and applications, pp. 306–346. Cambridge university press (2003)

[172] Hur, A., Janjua, N., Ahmed, M.: A survey on state-of-the-art techniques for knowledge graphs construction and challenges ahead. In: 2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). pp. 99–103 (2021). doi: 10.1109/AIKE52691.2021.00021

[173] Hyvönen, E., Rantala, H.: Knowledge-based relation discovery in cultural heritage knowledge graphs. In: Navarretta, C., Agirrezabal, M., Maegaard, B. (eds.) Proceedings of the Digital Humanities in the Nordic Countries 4th Conference, Copenhagen, Denmark, March 5-8, 2019. CEUR Workshop Proceedings, vol. 2364, pp. 230–239. CEUR-WS.org (2019), https://ceur-ws.org/Vol-2364/21_paper.pdf

[174] Ji, S., Pan, S., Cambria, E., Marttinen, P., Yu, P.S.: A survey on knowledge graphs: Representation, acquisition and applications. CoRR **abs/2002.00388** (2020)

[175] Johnston, T., Scott, A.: Lipschitz bijections between Boolean functions. Combinatorics, Probability and Computing **30**, 513–525 (2021)

[176] Jörges, S.: The State of the Art in Code Generation, pp. 11–38. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). doi: 10.1007/978-3-642-36127-2_2, https://doi.org/10.1007/978-3-642-36127-2_2

[177] Kamp, H., Reyle, U.: From Discourse to Logic - Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory. Kluwer Academic Publishers (1993). doi: 10.1007/978-94-017-1616-1, https://dx.doi.org/10.1007/978-94-017-1616-1

[178] Kang, Y., Li, Y., Krishnaswamy, S.: A rigorous characterization of classification performance - A tale of four reasoners. In: Horrocks, I., Yatskevich, M., Jiménez-Ruiz, E. (eds.) Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012), Manchester, UK, July 1st, 2012. CEUR Workshop Proceedings, vol. 858. CEUR-WS.org (2012), https://ceur-ws.org/Vol-858/ore2012_paper8.pdf

[179] Keet, M.: An introduction to ontology engineering (2024), https://eng.libretexts.org

[180] Kersting, K., De Raedt, L., Raiko, T.: Logical hidden markov models. J. Artif. Intell. Res. (JAIR) **25**, 425–456 (04 2006). doi: 10.1613/jair.1675

[181] Kim, J.S., Choi, K.S.: Fact checking in knowledge graphs by logical consistency. Semantic Web Journal (2021), https://www.semantic-web-journal.net/content/fact-checking-knowledge-graphs-logical-consistency

[182] Kindermann, C., Lupp, D.P., Sattler, U., Thorstensen, E.: Generating ontologies from templates: A rule-based approach for capturing regularity. In: Ortiz, M., Schneider, T. (eds.) Proceedings of the 31st International Workshop on Description Logics co-located with 16th International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), Tempe, Arizona, US, October 27th - to - 29th, 2018. CEUR Workshop Proceedings, vol. 2211. CEUR-WS.org (2018), https://ceur-ws.org/Vol-2211/paper-22.pdf

[183] Kirst, D., Hermes, M.: Synthetic undecidability and incompleteness of first-order axiom systems in coq: Extended version. J. Autom. Reason. **67**(1) (mar 2023). doi: 10.1007/s10817-022-09647-x, https://doi.org/10.1007/s10817-022-09647-x

[184] Klimek, R., Grobler-Debska, K., Kucharska, E.: System for automatic generation of logical formulas. MATEC Web of Conferences **252** (2019). doi: 10.1051/matecconf/201925203005, https://www.matecconferences.org/articles/matecconf/abs/2019/01/matecconf_cmes2018_03005/matecconf_cmes2018_03005.html

[185] Klimek, Radosław, Grobler-Dębska, Katarzyna, Kucharska, Edyta: System for automatic generation of logical formulas. MATEC Web Conf. **252**, 03005 (2019). doi: 10.1051/matecconf/201925203005, https://doi.org/10.1051/matecconf/201925203005

[186] Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice-Hall, Inc., USA (1994)

[187] Knox, S.W.: Combining Classifiers, chap. 6, pp. 107–125. John Wiley & Sons, Ltd (2018). doi: https://doi.org/10.1002/9781119439868.ch6, https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119439868.ch6

[188] Knox, S.W.: Machine learning: a concise introduction, vol. 285. John Wiley & Sons (2018)

[189] Knox, S.W.: Regression, chap. 3, pp. 15–32. John Wiley & Sons, Ltd (2018). doi: https://doi.org/10.1002/9781119439868.ch3, https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119439868.ch3

[190] Knox, S.W.: Survey of Classification Techniques, chap. 4, pp. 33–95. John Wiley & Sons, Ltd (2018). doi: https://doi.org/10.1002/9781119439868.ch4, https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119439868.ch4

[191] Kontchakov, R., Zakharyaschev, M.: An Introduction to Description Logics and Query Rewriting, pp. 195–244. Springer International Publishing, Cham (2014). doi: 10.1007/978-3-319-10587-1_5, https://doi.org/10.1007/978-3-319-10587-1_5

[192] Koutsomitropoulos, D.A., Meidanis, D.P., Kandili, A.N., Papatheodorou, T.S.: OWL-based knowledge discovery using description logics reasoners. In: International Conference on Enterprise Information Systems. vol. 2, pp. 43–50. SCITEPRESS (2006)

[193] Krawczyk, K.A.: Three model-theoretic constructions for generalized epstein semantics. Review of Symbolic Logic **15**(4), 1023–1032 (2022). doi: 10.1017/s1755020321000368

[194] Kriegel, F.: Acquisition of Terminological Knowledge from Social Networks in Description Logic, pp. 97–142. Springer International Publishing, Cham (2017). doi: 10.1007/978-3-319-64167-6_5, https://doi.org/10.1007/978-3-319-64167-6_5

[195] Krötzsch, M.: OWL 2 profiles: An introduction to lightweight ontology languages. In: Reasoning Web International Summer School, pp. 112–183. Springer (2012). doi: 10.1007/978-3-642-33158-9_4, https://dx.doi.org/10.1007/978-3-642-33158-9_4

[196] Krötzsch, M.: OWL 2 Profiles: An Introduction to Lightweight Ontology Languages, pp. 112–183. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). doi: 10.1007/978-3-642-33158-9_4, https://doi.org/10.1007/978-3-642-33158-9_4

[197] Kuhn, M., Johnson, K.: Feature selection: An overview. In: Feature Engineering and Selection. CRC Press (2018). doi: 10.1201/9781315108230-10, https://www.taylorfrancis.com/chapters/mono/10.1201/9781315108230-10/feature-selection-overview-max-kuhn-kjell-johnson

[198] Kuhn, M., Johnson, K.: Feature engineering and selection: A practical approach for predictive models. Chapman and Hall/CRC (2019)

[199] Kulmanov, M., Smaili, F.Z., Gao, X., Hoehndorf, R.: Semantic similarity and machine learning with ontologies. Briefings in Bioinformatics **22**(4), bbaa199 (10 2020). doi: 10.1093/bib/bbaa199, https://doi.org/10.1093/bib/bbaa199

[200] Kung, S.Y.: Dimension-reduction: PCA/KPCA and feature selection, p. 77–78. Cambridge University Press (2014)

[201] Kung, S.Y.: Feature selection, p. 118–138. Cambridge University Press (2014). doi: 10.1017/CBO9781139176224.007

[202] Kung, S.Y.: Machine learning and kernel vector spaces, p. 1–2. Cambridge University Press (2014). doi: 10.1017/CBO9781139176224.002

[203] Kurokawa, H.: On the semantic concept of logical consequence. In: Sakamoto, M., Okazaki, N., Mineshima, K., Satoh, K. (eds.) New Frontiers in Artificial Intelligence. pp. 258–275. Springer International Publishing, Cham (2020)

[204] Köhler, S., Bauer, S., Mungall, C., Carletti, G., Smith, C., Schofield, P., Gkoutos, G., Robinson, P.: Improving ontologies by automatic reasoning and evaluation of logical definitions. BMC Bioinformatics **12**(Suppl 4), S5 (10 2011). doi: 10.17863/CAM.11412

[205] Lakemeyer, G., Levesque, H.J.: A First-Order Logic of Limited Belief Based on Possible Worlds. In: Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning. pp. 624–635 (9 2020). doi: 10.24963/kr.2020/62, https://doi.org/10.24963/kr.2020/62

[206] Lam, A.N., Elvesæter, B., Martin-Recuerda, F.: A performance evaluation of OWL 2 DL reasoners using ore 2015 and very large bio ontologies. Proceedings of the DMKG (2023)

[207] Lamy, J.B.: Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. Artificial intelligence in medicine **80**, 11–28 (2017)

[208] van der Lee, C., Krahmer, E., Wubben, S.: Automated learning of templates for data-to-text generation: comparing rule-based, statistical and neural methods. In: Krahmer, E., Gatt, A., Goudbeek, M. (eds.) Proceedings of the 11th International Conference on Natural Language Generation. pp. 35–45. Association for Computational Linguistics, Tilburg University, The Netherlands (Nov 2018). doi: 10.18653/v1/W18-6504, https://aclanthology.org/W18-6504

[209] Lee, J., Meng, Y.: First-order stable model semantics and first-order loop formulas. Journal of Artificial Intelligence Research (2011). doi: 10.1613/jair.3337, https://dx.doi.org/10.1613/jair.3337

[210] Lee, S.Y., Choi, H.J., Jeong, Y.J., Kim, T.H., Chae, H.S., Chang, C.K.: An improved technique of fitness evaluation for evolutionary testing. In: 2011 IEEE 35th Annual Computer Software and Applications Conference Workshops. pp. 190–193 (2011). doi: 10.1109/COMPSACW.2011.41

[211] Lee, W.N., Shah, N., Sundlass, K., Musen, M.: Comparison of ontology-based semantic-similarity measures. In: AMIA annual symposium proceedings. vol. 2008, p. 384. American Medical Informatics Association (2008)

[212] Leguy, J., Cauchy, T., Glavatskikh, M., Duval, B., Da Mota, B.: Evomol: a flexible and interpretable evolutionary algorithm for unbiased de novo molecular generation. Journal of Cheminformatics **12** (09 2020). doi: 10.1186/s13321-020-00458-z

[213] Lehmann, J.: DL-Learner: learning concepts in description logics. The Journal of Machine Learning Research **10**, 2639–2642 (2009)

[214] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., Bizer, C.: Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal **6** (01 2014). doi: 10.3233/SW-140134

[215] Lehmann, J., Völker, J.: Perspectives on ontology learning, vol. 18. IOS Press (2014)

[216] Lembo, D., Santarelli, V., Savo, D.F., et al.: A graph-based approach for classifying OWL 2 ql ontologies. In: Description Logics. pp. 747–759 (2013)

[217] Lenko, V., Pasichnyk, V., Kunanets, N., Shcherbyna, Y.: Knowledge representation and automated formal reasoning in description logic ALC. In: Emmerich, M., Lytvyn, V., Vysotska, V., Basto-Fernandes, V., Lytvynenko, V. (eds.) Modern Machine Learning Technologies and Data Science Workshop. Proc. 3rd International Workshop (MoMLeT&DS 2021). Volume I: Main Conference, Lviv-Shatsk, Ukraine, June 5-6, 2021. CEUR Workshop Proceedings, vol. 2917, pp. 26–39. CEUR-WS.org (2021), https://ceur-ws.org/Vol-2917/paper3.pdf

[218] Li, H.: Decision Tree, pp. 77–102. Springer Nature Singapore, Singapore (2024). doi: 10.1007/978-981-99-3917-6_5, https://doi.org/10.1007/978-981-99-3917-6_5

[219] Li, H.: Introduction to Machine Learning and Supervised Learning, pp. 1–37. Springer Nature Singapore, Singapore (2024). doi: 10.1007/978-981-99-3917-6_1, https://doi.org/10.1007/978-981-99-3917-6_1

[220] Li, H.: K-Nearest Neighbor, pp. 55–66. Springer Nature Singapore, Singapore (2024). doi: 10.1007/978-981-99-3917-6_3, https://doi.org/10.1007/978-981-99-3917-6_3

[221] Li, H.: Latent Semantic Analysis, pp. 365–385. Springer Nature Singapore, Singapore (2024). doi: 10.1007/978-981-99-3917-6_17, https://doi.org/10.1007/978-981-99-3917-6_17

[222] Li, H.: Logistic Regression and Maximum Entropy Model, pp. 103–125. Springer Nature Singapore, Singapore (2024). doi: 10.1007/978-981-99-3917-6_6, https://doi.org/10.1007/978-981-99-3917-6_6

[223] Li, H.: Support Vector Machine, pp. 127–177. Springer Nature Singapore, Singapore (2024). doi: 10.1007/978-981-99-3917-6_7, https://doi.org/10.1007/978-981-99-3917-6_7

[224] Li, Z., Huang, M., Zhong, Y., Qin, Y.: A description logic based ontology for knowledge representation in process planning for laser powder bed fusion. Applied Sciences **12**(9) (2022). doi: 10.3390/app12094612, https://www.mdpi.com/2076-3417/12/9/4612

[225] Liang, S., Stockinger, K., de Farias, T.M., Anisimova, M., Gil, M.: Querying knowledge graphs in natural language. J Big Data **8**(1), 3 (Jan 2021)

[226] Linked Open Data Cloud: The linked open data cloud (2024), https://lod-cloud.net/

[227] Lippman, D.: 2.1: Introduction to Formal Logic (2023), https://math.libretexts.org

[228] Lisi, F.A., Straccia, U.: A logic-based computational method for the automated induction of fuzzy ontology axioms. Fundamenta Informaticae **124**(4), 503–519 (2013)

[229] Liu, S., d'Aquin, M., Motta, E.: Measuring accuracy of triples in knowledge graphs. In: International Conference on Language, Data, and Knowledge (2017), https://api.semanticscholar.org/CorpusID:28647092

[230] Longo, C.F., Santoro, C.: A framework to build abductive-deductive chatbots, based on natural language processing and first-order logic. In: Calegari, R., Ciatto, G., Omicini, A. (eds.) Proceedings of the 37th Italian Conference on Computational Logic, Bologna, Italy, June 29 - July 1, 2022. CEUR Workshop Proceedings, vol. 3204, pp. 75–89. CEUR-WS.org (2022), https://ceur-ws.org/Vol-3204/paper_7.pdf

[231] Lu, C.: From bayesian inference to logical bayesian inference: A new mathematical frame for semantic communication and machine learning. In: Intelligence Science II: Third IFIP TC 12 International Conference, ICIS 2018, Beijing, China, November 2-5, 2018, Proceedings 2. pp. 11–23. Springer (2018)

[232] Lyon, T.S., Ostropolski-Nalewaja, P.: Connecting Proof Theory and Knowledge Representation: Sequent Calculi and the Chase with Existential Rules. In: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning. pp. 769–773 (8 2023). doi: 10.24963/kr.2023/79, https://doi.org/10.24963/kr.2023/79

[233] Ma., S., An., Y., Wang., J., Lawlor., A., Dong., R.: Adaptive adversarial samples based active learning for medical image classification. In: Proceedings of the 12th International Conference on Pattern Recognition Applications and Methods - ICPRAM. pp. 751–758. INSTICC, SciTePress (2023). doi: 10.5220/0011622100003411

[234] Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent systems **16**(2), 72–79 (2001)

[235] Mahmud, N., Seceleanu, C., Ljungkrantz, O.: Specification and semantic analysis of embedded systems requirements: From description logic to temporal logic. In: Cimatti, A., Sirjani, M. (eds.) Software Engineering and Formal Methods. pp. 332–348. Springer International Publishing, Cham (2017)

[236] Makinson, D.: Propositional relevance through letter-sharing. J. Appl. Log. **7**(4), 377–387 (2009)

[237] Makni, B., Hendler, J.: Deep learning for noise-tolerant rdfs reasoning. Semantic Web **10**(5), 823–862 (2019)

[238] Malchiodi, D., da Costa Pereira, C., Tettamanzi, A.: Predicting the possibilistic score of OWL axioms through support vector regression. In: Ciucci, D., Pasi, G., Vantaggi, B. (eds.) Scalable Uncertainty Management. SUM 2018. Lecture Notes in Artificial Intelligence 11142, Springer, Cham (2018)

[239] Malchiodi, D., da Costa Pereira, C., Tettamanzi, A.: Classifying candidate axioms via dimensionality reduction techniques. In: Torra, V., Narukawa, Y., Nin, J., Agell, N. (eds.) Modeling Decisions for Artificial Intelligence. 17th International Conference, MDAI 2020 Sant Cugat, Spain, September 2–4, 2020 Proceedings. p. 179—191. Lecture Notes in Computer Sciencce 12256, Springer, Cham, Switzerland (2020)

[240] Malchiodi, D., Tettamanzi, A.: Predicting the possibilistic score of OWL axioms through modified support vector clustering. In: Haddad, H., Wainwright, R., Chbeir, R. (eds.) SAC'18: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. p. 1984–1991. ACM (2018)

[241] Mandal, S., Karim, I.U., Raha, S.: A theory of approximate reasoning with type-2 fuzzy set. Journal of the Indonesian Mathematical Society **27**(1), 9–28 (2021)

[242] Mann, C.: The description logic handbook – theory, implementation and applications. Kybernetes **32** (2003). doi: 10.1108/k.2003.06732iae.006

[243] Manzoor, S., Rocha, Y.G., Joo, S.H., Bae, S.H., Kim, E.J., Joo, K.J., Kuc, T.Y.: Ontology-based knowledge representation in robotic systems: A survey oriented toward applications. Applied Sciences **11**(10), 4324 (2021)

[244] Mayburd, A., Baranova, A.: Knowledge-Based Compact Disease Models: A Rapid Path from High-Throughput Data to Understanding Causative Mechanisms for a Complex Disease, pp. 425–461. Springer New York, New York, NY (2017). doi: 10.1007/978-1-4939-7027-8_17, https://doi.org/10.1007/978-1-4939-7027-8_17

[245] McHugh, M.L.: The chi-square test of independence. Biochemia medica **23**(2), 143–149 (2013)

[246] Memariani, A., Glauer, M., Neuhaus, F., Mossakowski, T., Hastings, J.: Automated and explainable ontology extension based on deep learning: A case study in the chemical domain. In: Confalonieri, R., Kutz, O., Calvanese, D. (eds.) International Workshop on Data meets Applied Ontologies in Explainable AI (DAO-XAI 2021). CEUR Workshop Proceedings, vol. 2998. http://ceur-ws.org/Vol-2998/ (2021), http://ceur-ws.org/Vol-2998/paper1.pdf

[247] Mendonça, B.R.: A refinement of urn semantics for first-order logic. Logic and Logical Philosophy (2022). doi: 10.12775/llp.2022.021, https://dx.doi.org/10.12775/llp.2022.021

[248] Mežnar, S., Bevec, M., Lavrač, N., Škrlj, B.: Ontology completion with graph-based machine learning: A comprehensive evaluation. Machine Learning and Knowledge Extraction **4** (2022). doi: 10.3390/make4040056

[249] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: International Conference on Learning Representations (2013)

[250] Milner, R.: Edinburgh LCF: A mechanised logic of computation. LNCS **78** (1980)

[251] Mironov, V., Seethappan, N., Blondé, W., Antezana, E., Splendiani, A., Kuiper, M.: Gauging triple stores with actual biological data. BMC Bioinformatics **13**(Suppl 1), S3 (2012). doi: 10.1186/1471-2105-13-S1-S3, https://dx.doi.org/10.1186/1471-2105-13-S1-S3

[252] Moore, R.C.: The role of logic in knowledge representation and commonsense reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 2. AAAI (2023)

[253] Morishita, T., Morio, G., Yamaguchi, A., Sogawa, Y.: Learning deductive reasoning from synthetic corpus based on formal logic. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 25254–25274. PMLR (23–29 Jul 2023), https://proceedings.mlr.press/v202/morishita23a.html

[254] Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web ontology language structural specification and functional-style syntax (second edition). W3c recommendation, World Wide Web Consortium (W3C) (December 2012), https://www.w3.org/TR/owl2-syntax/, accessed: 2023-09-28

[255] Muggleton, S., De Raedt, L., Poole, D., Bratko, I., Flach, P., Inoue, K., Srinivasan, A.: Ilp turns 20: biography and future challenges. Machine learning **86**, 3–23 (2012)

[256] Mugzach, O., Peleg, M., Bagley, S.C., Altman, R.B.: Expanding the autism ontology to DSM-IV criteria. In: AMIA 2013, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November

16-20, 2013. AMIA (2013), https://knowledge.amia.org/amia-55142-a2013e-1.580047/t-006-1.582200/f-006-1.582201/a-365-1.582908/a-367-1.582902

[257] Murty, M.N., Raghava, R.: Kernel-based svm. In: Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks, pp. 57–67. Springer International Publishing, Cham (2016). doi: 10.1007/978-3-319-41063-0_5

[258] Mussmann, S.: Understanding and Analyzing the Effectiveness of Uncertainty Sampling. Stanford University (2021)

[259] Mussmann, S., Reisler, J., Tsai, D., Mousavi, E., O'Brien, S., Goldszmidt, M.: Active learning with expected error reduction (2022), https://arxiv.org/abs/2211.09283

[260] Ngonga Ngomo, A.C., Fundulaki, I., Krithara, A., Rashid, M., Torchiano, M., Rizzo, G., Mihindukulasooriya, N., Corcho, O., Ngonga Ngomo, A.C., Fundulaki, I., Krithara, A.: A quality assessment approach for evolving knowledge bases. Semant. Web **10**(2), 349–383 (jan 2019). doi: 10.3233/SW-180324, https://doi.org/10.3233/SW-180324

[261] Ngootip, T., Manorom, P., Chansanam, W., Buranarach, M.: Developing a linked open data platform for folktales in the Greater Mekong subregion. Engineering, Science and Technology, an International Journal (2023). doi: 10.28991/esj-2023-07-06-06, https://dx.doi.org/10.28991/esj-2023-07-06-06

[262] Nguyen, H., Patrick, J.: Text mining in clinical domain: Dealing with noise. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 549–558. KDD '16, Association for Computing Machinery, New York, NY, USA (2016). doi: 10.1145/2939672.2939720, https://doi.org/10.1145/2939672.2939720

[263] Nguyen, T.H.: Mining the semantic Web for OWL axioms. (Fouille du Web sémantique à la recherche d'axiomes OWL). Ph.D. thesis, University of Côte d'Azur, Nice, France (2021), https://tel.archives-ouvertes.fr/tel-03406784

[264] Nguyen, T.H., Tettamanzi, A.G.B.: An evolutionary approach to class disjointness axiom discovery. In: Barnaghi, P.M., Gottlob, G., Manolopoulos, Y., Tzouramanis, T., Vakali, A. (eds.) 2019 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2019, Thessaloniki, Greece, October 14-17, 2019. pp. 68–75. ACM (2019). doi: 10.1145/3350546.3352502, https://doi.org/10.1145/3350546.3352502

[265] Nguyen, T.H., Tettamanzi, A.G.B.: Learning class disjointness axioms using grammatical evolution. In: Sekanina, L., Hu, T., Lourenço, N., Richter, H., García-Sánchez, P. (eds.) Genetic Programming - 22nd European Conference, EuroGP 2019, Held as Part of EvoStar 2019, Leipzig, Germany, April 24-26,

2019, Proceedings. Lecture Notes in Computer Science, vol. 11451, pp. 278–294. Springer (2019). doi: 10.1007/978-3-030-16670-0_18, https://doi.org/10.1007/978-3-030-16670-0_18

[266] Nguyen, T.H., Tettamanzi, A.G.B.: Grammatical evolution to mine OWL disjointness axioms involving complex concept expressions. In: IEEE Congress on Evolutionary Computation, CEC 2020, Glasgow, United Kingdom, July 19-24, 2020. pp. 1–8. IEEE (2020). doi: 10.1109/CEC48606.2020.9185681, https://doi.org/10.1109/CEC48606.2020.9185681

[267] Nguyen, T.H., Tettamanzi, A.G.B.: A multi-objective evolutionary approach to class disjointness axiom discovery. In: He, J., Purohit, H., Huang, G., Gao, X., Deng, K. (eds.) IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, WI/IAT 2020, Melbourne, Australia, December 14-17, 2020. pp. 275–282. IEEE (2020). doi: 10.1109/WIIAT50758.2020.00040, https://doi.org/10.1109/WIIAT50758.2020.00040

[268] Nguyen, T.H., Tettamanzi, A.G.B.: Using grammar-based genetic programming for mining disjointness axioms involving complex class expressions. In: Alam, M., Braun, T., Yun, B. (eds.) Ontologies and Concepts in Mind and Machine - 25th International Conference on Conceptual Structures, ICCS 2020, Bolzano, Italy, September 18-20, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12277, pp. 18–32. Springer (2020). doi: 10.1007/978-3-030-57855-8_2, https://doi.org/10.1007/978-3-030-57855-8_2

[269] Nickel, M., Tresp, V., Kriegel, H.P.: Factorizing YAGO : Scalable machine learning for linked data. WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web pp. 271–280 (2012). doi: 10.1145/2187836.2187874

[270] Nikolaidis, E., Chen, S., Cudney, H., Haftka, R.T., Rosca, R.: Comparison of Probability and Possibility for Design Against Catastrophic Failure Under Uncertainty . Journal of Mechanical Design **126**(3), 386–394 (09 2003). doi: 10.1115/1.1701878, https://doi.org/10.1115/1.1701878

[271] Ontotext: What are ontologies? https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/ (2020)

[272] Ontotext: What is RDF and why to use it? https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf/ (2020)

[273] Ostertagova, E., Ostertag, O.: Methodology and application of one-way anova. American Journal of Mechanical Engineering **1**, 256–261 (11 2013). doi: 10.12691/ajme-1-7-21

[274] Ozaki, A.: Learning description logic ontologies: Five approaches. where do they stand? KI - Künstliche Intelligenz (2020). doi: 10.1007/s13218-020-00656-9, https://dx.doi.org/10.1007/s13218-020-00656-9

[275] Paliwal, A., Loos, S., Rabe, M., Bansal, K., Szegedy, C.: Graph representations for higher-order logic and theorem proving. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 2967–2974 (2020)

[276] Pareek, S., Gupta, H., Kaur, J., Kumar, R., Chohan, J.S.: Fuzzy logic in computer technology: Applications and advancements. In: 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN). pp. 1634–1637 (2023). doi: 10.1109/ICPCSN58827.2023.00273, https://dx.doi.org/10.1109/ICPCSN58827.2023.00273

[277] Parry, R., Jones, W., Stokes, T., Phan, J., Moffitt, R., Fang, H., Shi, L., Oberthuer, A., Fischer, M., Tong, W., et al.: k-nearest neighbor models for microarray gene expression analysis and clinical outcome prediction. The pharmacogenomics journal **10**, 292–309 (2010)

[278] Passalis, N., Tefas, A.: Dimensionality reduction using similarity-induced embeddings. IEEE transactions on neural networks and learning systems **29**(8), 3429–3441 (2017)

[279] Patton, E.W.: Energy-aware reasoning agents for the mobile semantic web. Rensselaer Polytechnic Institute (2016)

[280] Peng, C., Xia, F., Naseriparsa, M., Osborne, F.: Knowledge graphs: Opportunities and challenges. Artificial Intelligence Review **56**(11), 13071–13102 (2023)

[281] Peng, S., Yuan, K., Gao, L., Tang, Z.: Mathbert: A pre-trained model for mathematical formula understanding. arXiv **abs/2105.00377** (2021), https://api.semanticscholar.org/CorpusID:233481495

[282] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)

[283] Peters, S., Westerståhl, D.: The emergence of generalized quantifiers in modern logic. In: Quantifiers in Language and Logic. Oxford University Press (2008). doi: 10.1093/acprof:oso/9780199291267.003.0003, https://doi.org/10.1093/acprof:oso/9780199291267.003.0003

[284] Petrucci, G.: Learning to Learn Concept Descriptions. Ph.D. thesis, University of Trento, Italy (2018), http://eprints-phd.biblio.unitn.it/3088/

[285] Phillips, J.P.: A representation reducing approach to scientific discovery. In: Hendler, J., Subramanian, D. (eds.) Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, July 18-22, 1999, Orlando, Florida, USA. p. 978. AAAI Press / The MIT Press (1999), http://www.aaai.org/Library/AAAI/1999/aaai99-185.php

[286] Pilehvar, M.T., Camacho-Collados, J.: Embeddings in natural language processing: Theory and advances in vector representations of meaning. Morgan & Claypool Publishers (2020)

[287] Piotrowski, B., Urban, J.: Guiding inferences in connection tableau by recurrent neural networks. In: International Conference on Intelligent Computer Mathematics. pp. 309–314. Springer (2020)

[288] Putri, S.N., Saputro, D.: Construction fuzzy logic with curve shoulder in inference system mamdani. In: Journal of Physics: Conference Series. vol. 1776, p. 012060. IOP Publishing (2021)

[289] Ralph, B.: Herbrand Proofs and Expansion Proofs as Decomposed Proofs. Journal of Logic and Computation **30**(8), 1711–1742 (10 2020). doi: 10.1093/logcom/exaa052, https://doi.org/10.1093/logcom/exaa052

[290] Ramesh, C., Rao, C.K., Govardhan, A.: Web mining based framework for ontology learning. Computer Science & Information Technology **5**(13), 43–56 (2015). doi: 10.5121/CSIT.2015.51304

[291] Ramirez, A., Cheng, B., Mckinley, P., Beckmann, B.: Automatically generating adaptive logic to balance non-functional tradeoffs during reconfiguration. In: Proceeding of the 7th International Conference on Autonomic Computing, ICAC '10 and Co-located Workshops. pp. 225–234 (06 2010). doi: 10.1145/1809049.1809080

[292] Ramsay, A.M.: Formal Methods in Artificial Intelligence. Cambridge University Press (1991)

[293] Rathjen, M., Sieg, W.: Proof Theory. In: Zalta, E.N., Nodelman, U. (eds.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, Fall 2023 edn. (2023)

[294] Raza, Z., Mahmood, K., Warraich, N.: Application of linked data technologies in digital libraries: a review of literature. Library Hi Tech News **36**(3) (11 2019)

[295] Razouk, H., Liu, X.L., Kern, R.: Improving FMEA comprehensibility via common-sense knowledge graph completion techniques. IEEE Access **11**, 127974–127986 (2023). doi: 10.1109/ACCESS.2023.3331585

[296] RDF Working Group: Resource description framework (rdf). https://www.w3.org/RDF/ (2014)

[297] Restall, G.: Logic: An Introduction. Routledge, New York (2006), https://www.routledge.com/Logic-An-Introduction/Restall/p/book/9780415400688

[298] Riboni, D., Bettini, C.: OWL 2 modeling and reasoning with complex human activities. Pervasive and Mobile Computing **7**(3), 379–395 (2011). doi: 10.1016/j.pmcj.2011.02.001

[299] Riegel, R., Gray, A., Luus, F., Khan, N., Makondo, N., Akhalwaya, I.Y., Qian, H., Fagin, R., Barahona, F., Sharma, U., et al.: Logical neural networks. arXiv preprint arXiv:2006.13155 (2020)

[300] Ristoski, P., Rosati, J., Noia, T.D., Leone, R.D., Paulheim, H.: Rdf2vec: RDF graph embeddings and their applications. Semantic Web Journal (2017)

[301] Roffe, A.J., Calderon, J.S.T.: Random formula generators. CoRR **abs/2110.09228** (2021), https://arxiv.org/abs/2110.09228

[302] Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. Int. Conf. on Machine Learning (2001)

[303] Roy-Chowdhury, R., Dalal, M.: Model-theoretic semantics and tractable algorithm for CNF-BCP. In: Kuipers, B., Webber, B.L. (eds.) Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island, USA. pp. 227–232. AAAI Press / The MIT Press (1997), http://www.aaai.org/Library/AAAI/1997/aaai97-036.php

[304] Rudolph, S.: Foundations of Description Logics, pp. 76–136. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). doi: 10.1007/978-3-642-23032-5_2, https://doi.org/10.1007/978-3-642-23032-5_2

[305] Rushby, J.: Integrated formal verification: Using model checking with automated abstraction, invariant generation, and theorem proving. In: International SPIN Workshop on Model Checking of Software. pp. 1–11. Springer (1999)

[306] Ryan, C., O'Neill, M., Collins, J.J.: Handbook of Grammatical Evolution. Springer Publishing Company, Incorporated, 1st edn. (2018)

[307] Sagi, G.: Models and logical consequence. Journal of Philosophical Logic **43**(5), 943–964 (2014). doi: 10.1007/s10992-013-9303-5

[308] Sakhanenko, N.A., Galas, D.J.: Probabilistic logic methods and some applications to biology and medicine. Journal of Computational Biology **19**(3), 316–336 (2012)

[309] Sakharov, A.: First-order logic. Wolfram MathWorld (2002), http://mathworld.wolfram.com/First-OrderLogic.html

[310] Sathiya, B., Geetha, T.: A review on semantic similarity measures for ontology. Journal of Intelligent & Fuzzy Systems **36**(4), 3045–3059 (2019)

[311] Saylor Academy: PHIL102 (2018.A.01): Sentential Logic and Well-Formed Formulas (2023), https://learn.saylor.org

[312] Schefe, P.: Some fundamental issues in knowledge representation. In: Informatik-Fachberichte (1982). doi: 10.1007/978-3-642-68826-3_4, https://dx.doi.org/10.1007/978-3-642-68826-3_4

[313] Scheffler, I., Goodman, N.: Selective confirmation and the ravens: A reply to foster. The Journal of Philosophy **69**(3), 78–83 (1972)

[314] Schlichtkrull, A.: Formalization of the resolution calculus for first-order logic. Journal of Automated Reasoning (2018). doi: 10.1007/s10817-017-9447-z, https://dx.doi.org/10.1007/s10817-017-9447-z

[315] Schneider, P., Schopf, T., Vladika, J., Galkin, M., Simperl, E., Matthes, F.: A decade of knowledge graphs in natural language processing: A survey. In: He, Y., Ji, H., Li, S., Liu, Y., Chang, C.H. (eds.) Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 601–614. Association for Computational Linguistics, Online only (Nov 2022), https://aclanthology.org/2022.aacl-main.46

[316] Schwarz, D.F., König, I.R., Ziegler, A.: On safari to random jungle: a fast implementation of random forests for high-dimensional data. Bioinformatics **26**(14), 1752–1758 (2010)

[317] Seaborne, A., Prud'hommeaux, E.: SPARQL query language for RDF. W3c recommendation, World Wide Web Consortium (2008), https://www.w3.org/TR/rdf-sparql-query/, published on 15 January 2008

[318] Settles, B.: Active learning literature survey. Machine Learning **15** (2010). doi: 10.1.1.167.4245

[319] Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. p. 287–294. COLT '92, Association for Computing Machinery, New York, NY, USA (1992). doi: 10.1145/130385.130417, https://doi.org/10.1145/130385.130417

[320] Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Proceedings of the fifth annual workshop on Computational learning theory. pp. 287–294 (1992)

[321] Shadbolt, N., Hall, W., Berners-Lee, T.: The semantic web revisited. IEEE Intelligent Systems **21**(3), 96–101 (2006)

[322] Shapiro, S.: Connectives, Quantifiers, Logic. In: Vagueness in Context. Oxford University Press (01 2006). doi: 10.1093/acprof:oso/9780199280391.003.0004, https://doi.org/10.1093/acprof:oso/9780199280391.003.0004

[323] Shapiro, S., Kouri Kissel, T.: Classical Logic. In: Zalta, E.N., Nodelman, U. (eds.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, Spring 2024 edn. (2024)

[324] Shearer, R., Horrocks, I.: Exploiting partial information in taxonomy construction. In: International Semantic Web Conference. pp. 569–584. Springer (2009)

[325] Shi, F., Li, J., Tang, J., Xie, G., Li, H.: Actively learning ontology matching via user interaction. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2009. pp. 585–600. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[326] Shi, S., Chen, H., Ma, W., Mao, J., Zhang, M., Zhang, Y.: Neural logic reasoning. In: d'Aquin, M., Dietze, S., Hauff, C., Curry, E., Cudré-Mauroux, P. (eds.) CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020. pp. 1365–1374. ACM (2020). doi: 10.1145/3340531.3411949, https://doi.org/10.1145/3340531.3411949

[327] Shminke, B.: Applications de l'IA à l'étude des structures algébriques finies et à la démonstration automatique de théorèmes. Ph.D. thesis, Université Côte d'Azur, Nice, France (9 2023), https://www.theses.fr/2023COAZ4058, thèse de doctorat en Mathématiques

[328] Shofi, I.M., Budiardjo, E.K.: Addressing OWL ontology for goal consistency checking. In: Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services. pp. 336–341 (2012)

[329] Siddiqui, N., Verma, A., Shrivastava, D.: Fuzzy logic-based MPPT control for bifacial photovoltaic module. In: 2023 Second International Conference on Electronics and Renewable Systems (ICEARS). pp. 1–6 (2023). doi: 10.1109/ICEARS56392.2023.10085389, https://dx.doi.org/10.1109/ICEARS56392.2023.10085389

[330] Simon, S.M., Glaum, P., Valdovinos, F.S.: Interpreting random forest analysis of ecological models to move from prediction to explanation. Scientific Reports **13**(1), 3881 (2023)

[331] Simperl, E., Bürger, T., Hangl, S., Wörgl, S., Popov, I.: Ontocom: A reliable cost estimation method for ontology development projects. Journal of Web Semantics **16**, 1–16 (2012)

[332] Singh, G., Bhatia, S., Mutharaju, R.: Owl2bench: A benchmark for OWL 2 reasoners. In: Pan, J.Z., Tamma, V., d'Amato, C., Janowicz, K., Fu, B., Polleres, A., Seneviratne, O., Kagal, L. (eds.) The Semantic Web – ISWC 2020. pp. 81–96. Springer International Publishing, Cham (2020)

[333] Singh, G., Bhatia, S., Mutharaju, R.: Neuro-symbolic RDF and description logic reasoners: The state-of-the-art and challenges. In: Compendium of Neurosymbolic Artificial Intelligence, pp. 29–63. IOS Press (2023)

[334] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics **5**(2), 51–53 (2007)

[335] Skjæveland, M.G., Forssell, H., Klüwer, J.W., Lupp, D.P., Thorstensen, E., Waaler, A.: Pattern-based ontology design and instantiation with reasonable ontology templates. In: Blomqvist, E., Corcho, Ó., Horridge, M., Carral, D., Hoekstra, R. (eds.) Proceedings of the 8th Workshop on Ontology Design and Patterns (WOP 2017) co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017. CEUR Workshop Proceedings, vol. 2043. CEUR-WS.org (2017), https://ceur-ws.org/Vol-2043/paper-04.pdf

[336] Smaili, F.Z., Gao, X., Hoehndorf, R.: Onto2Vec: joint vector-based representation of biological entities and their ontology-based annotations. Bioinformatics **34**(13), i52–i60 (06 2018). doi: 10.1093/bioinformatics/bty259, https://doi.org/10.1093/bioinformatics/bty259

[337] Smaili, F.Z., Gao, X., Hoehndorf, R.: Opa2vec: Combining formal and informal content of biomedical ontologies to improve similarity-based prediction. Bioinformatics **35** (2019). doi: 10.1093/bioinformatics/bty933

[338] Smith, P.: An introduction to formal logic. Cambridge University Press (2003)

[339] Smith, P.: An Introduction to Formal Logic. Cambridge University Press, 2nd edn. (2020), https://www.logicmatters.net/ifl/

[340] Sowa, J.F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co (2000), http://www.jfsowa.com/krbook/

[341] Spangler, S., Wilkins, A.D., Bachman, B.J., Nagarajan, M., Dayaram, T., Haas, P., Regenbogen, S., Pickering, C.R., Comer, A., Myers, J.N., Stanoi, I., Kato, L., Lelescu, A., Labrie, J.J., Parikh, N., Lisewski, A.M., Donehower, L., Chen, Y., Lichtarge, O.: Automated hypothesis generation based on mining scientific literature. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1877–1886. KDD '14, Association for Computing Machinery, New York, NY, USA (2014). doi: 10.1145/2623330.2623667, https://doi.org/10.1145/2623330.2623667

[342] Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10**(2), 99–127 (2002). doi: 10.1162/106365602320169811

[343] Sujatha, B., Viswanadha, S.: Natural language parser for database querying. International Journal of Computer Applications (2016). doi: 10.5120/IJCA2016908176, https://dx.doi.org/10.5120/IJCA2016908176

[344] Suryanarayana, D., et al.: Knowledge representation of natural language in high performance linguistics scheme. In: International Conference on Computational Intelligence and Computing Research (2016). doi: 10.1109/ICCIC.2016.7919548, https://dx.doi.org/10.1109/ICCIC.2016.7919548

[345] Suvorov, R., Shelmanov, A., Smirnov, I.: Active learning with adaptive density weighted sampling for information extraction from scientific papers. In: Artificial Intelligence and Natural Language: 6th Conference, AINL 2017, St. Petersburg, Russia, September 20–23, 2017, Revised Selected Papers 6. pp. 77–90. Springer (2018)

[346] Syafiandini, A., Song, G., Ahn, Y., Kim, H., Song, M.: An automatic hypothesis generation for plausible linkage between xanthium and diabetes. Scientific Reports **12** (10 2022). doi: 10.1038/s41598-022-20752-0

[347] SYSTAP, LLC: Blazegraph database. https://www.blazegraph.com/ (2024)

[348] Tettamanzi, A., Zucker, C.F., Gandon, F.: Possibilistic testing of OWL axioms against RDF data. International Journal of Approximate Reasoning (2017). doi: 10.1016/j.ijar.2017.08.012

[349] Tettamanzi, A.G.B., Zucker, C.F., Gandon, F.: Testing OWL axioms against RDF facts: A possibilistic approach. In: Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW 2014. pp. 519–530. Linköping, Sweden (2014). doi: 10.1007/978-3-319-13704-9_39

[350] Tettamanzi, A.G.B., Zucker, C.F., Gandon, F.: Dynamically time-capped possibilistic testing of subclassof axioms against RDF data to enrich schemas. In: Proceedings of the 8th International Conference on Knowledge Capture. K-CAP 2015, Association for Computing Machinery, New York, NY, USA (2015). doi: 10.1145/2815833.2815835, https://doi.org/10.1145/2815833.2815835

[351] Thompson, B., Personick, M., Cutcher, M.: The bigdata® RDF graph database. In: Harth, A., Hose, K., Schenkel, R. (eds.) Linked Data Management. pp. 193–237. CRC Press (2014)

[352] Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of machine learning research **2**(Nov), 45–66 (2001)

[353] Tran, T., Le, V., Le, H., Le, T.M.: From deep learning to deep reasoning. In: KDD '21: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 4076–4077 (2021). doi: 10.1145/3447548.3470803

[354] Traversa, F.L., Di Ventra, M.: Efficient solution of Boolean satisfiability problems with digital memcomputing. Scientific Reports **7**(1), 1–13 (2017). doi: 10.1038/s41598-017-07236-6, https://www.nature.com/articles/s41598-017-07236-6

[355] Tsarkov, D., Horrocks, I.: Fact++ description logic reasoner: System description. In: International joint conference on automated reasoning. pp. 292–297. Springer (2006)

[356] Tu, E., Zhang, Y., Zhu, L., Yang, J., Kasabov, N.: A graph-based semi-supervised k nearest-neighbor method for nonlinear manifold distributed data classification. Information Sciences—Informatics and Computer Science, Intelligent Systems, Applications: An International Journal **367**(C), 673–688 (2016)

[357] Turhan, A.Y.: Description logic reasoning for semantic web ontologies. In: Proceedings of the International Conference on Web Intelligence, Mining and Semantics. WIMS '11, Association for Computing Machinery, New York, NY, USA (2011). doi: 10.1145/1988688.1988696, https://doi.org/10.1145/1988688.1988696

[358] Tyagi, L., Singal, S.: Application of fuzzy logic control systems in military platforms. In: 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence). pp. 397–402 (2019). doi: 10.1109/CONFLUENCE.2019.8776955, https://dx.doi.org/10.1109/CONFLUENCE.2019.8776955

[359] Uddin, S., Haque, I., Lu, H., Moni, M.A., Gide, E.: Comparative performance analysis of k-nearest neighbour (knn) algorithm and its different variants for disease prediction. Scientific Reports **12**, 6256 (2022)

[360] Urban, J., Sutcliffe, G., Pudlák, P., Vyskočil, J.: Malarea sg1-machine learner for automated reasoning with semantic guidance. In: Automated Reasoning: 4th International Joint Conference, IJCAR 2008 Sydney, Australia, August 12-15, 2008 Proceedings 4. pp. 441–456. Springer (2008)

[361] Uschold, M., Gruninger, M.: Ontologies: principles, methods and applications. The Knowledge Engineering Review **11**(2), 93–136 (1996). doi: 10.1017/S0269888900007797

[362] Vakili, A., Day, N.A.: Reducing CTL-live model checking to first-order logic validity checking. In: 2014 Formal Methods in Computer-Aided Design (FMCAD). pp. 215–218 (2014). doi: 10.1109/FMCAD.2014.6987616

[363] Verhodubs, O.: Comparison of ontology reasoning systems for semantic web expert system. Tech. rep., EasyChair (2023)

[364] Vrandečić, D., Krötzsch, M.: Wikidata: A free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (sep 2014). doi: 10.1145/2629489, https://doi.org/10.1145/2629489

[365] W3C OWL Working Group: OWL 2 Web Ontology Language: Document Overview. http://www.w3.org/TR/owl2-overview/ (2009), w3C Recommendation, October 27

[366] W3C OWL Working Group: OWL 2 web ontology language document overview (second edition). W3c recommendation, World Wide Web Consortium (W3C) (December 2012), https://www.w3.org/TR/owl2-overview/, accessed: 2023-09-28

[367] W3C OWL Working Group: Web ontology language (OWL). https://www.w3.org/OWL/ (2012)

[368] W3C Semantic Web Standards: Semantic web standards. https://www.w3.org/2001/sw/wiki/Main_Page (2019)

[369] Wang, Z., Wang, K., Zhuang, Z., Qi, G.: Instance-driven ontology evolution in dl-lite. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. p. 1656–1662. AAAI'15, AAAI Press (2015)

[370] Wendler, J., Bach, J.: Recognizing and predicting agent behavior with case based reasoning. In: RoboCup. Lecture Notes in Computer Science, vol. 3020, pp. 729–738. Springer (2003)

[371] Winham, S.J., Colby, C.L., Freimuth, R.R., Wang, X., de Andrade, M., Huebner, M., Biernacka, J.M.: SNP interaction detection with random forests in high-dimensional genetic data. BMC bioinformatics **13**, 1–13 (2012)

[372] World Wide Web Consortium (W3C): Sparql 1.1 overview (2013), https://www.w3.org/TR/sparql11-overview/

[373] Wu, K., Haarslev, V.: Parallel OWL reasoning: Merge classification. In: Semantic Technology: Third Joint International Conference, JIST 2013, Seoul, South Korea, November 28–30, 2013, Revised Selected Papers. p. 211–227. Springer-Verlag, Berlin, Heidelberg (2013). doi: 10.1007/978-3-319-06826-8_17, https://doi.org/10.1007/978-3-319-06826-8_17

[374] Wu, S.H., Zhan, Z.H., Zhang, J.: SAFE: scale-adaptive fitness evaluation method for expensive optimization problems. IEEE Transactions on Evolutionary Computation **25**(3), 478–491 (2021). doi: 10.1109/TEVC.2021.3051608

[375] Wyner, A., Engers, T., Bahreini, K.: From policy-making statements to first-order logic. In: Conference on Electronic Government (2010). doi: 10.1007/978-3-642-15172-9_5, https://dx.doi.org/10.1007/978-3-642-15172-9_5

[376] XUE, J.z., YAN, X.g., ZHENG, C.x.: Classifications of eeg during mental tasks by kernel learning algorithms. ACTA ELECTONICA SINICA **32**(10), 1749 (2004)

[377] Xue, W., Bao, H., Xue, W., Huang, W., Lu, Y.: Web page classification based on svm. In: 2006 6th World Congress on Intelligent Control and Automation. vol. 2, pp. 6111–6114 (2006). doi: 10.1109/WCICA.2006.1714255

[378] Yang, L.A., Liu, J.P., Chen, C.H., Chen, Y.p.: Automatically proving mathematical theorems with evolutionary algorithms and proof assistants. In: 2016 IEEE Congress on Evolutionary Computation (CEC). p. 4421–4428. IEEE Press (2016). doi: 10.1109/CEC.2016.7744352, https://doi.org/10.1109/CEC.2016.7744352

[379] Yang, S., Zhu, W., Zhu, Y.: Sparse-dense subspace clustering. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 247–254 (2021). doi: 10.1109/ICPR48806.2021.9412260

[380] Yu, F., Zeng, Y., Mao, J., Li, W.: Online estimation of similarity matrices with incomplete data. In: Uncertainty in Artificial Intelligence. pp. 2454–2464. PMLR (2023)

[381] Yuxin, Y., Dantong, O., Yao, L., Jigui, S.: Consistency checking of the SHOIQ (D) based ontology. Computer Engineering & Science **1**(31), 08 (2009)

[382] Zadeh, L.: Fuzzy sets. Information and Control **8**(3), 338–353 (1965). doi: https://doi.org/10.1016/S0019-9958(65)90241-X, https://www.sciencedirect.com/science/article/pii/S001999586590241X

[383] Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy sets and systems **1**(1), 3–28 (1978)

[384] Zalta, E.N.: Propositional logic. Stanford Encyclopedia of Philosophy (2003), https://plato.stanford.edu/entries/logic-propositional/, retrieved January 12, 2024

[385] Zhang, C., Beetz, J., Vries, B.: BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data. Semantic Web (2018). doi: 10.3233/SW-180297, https://dx.doi.org/10.3233/SW-180297

[386] Zhang, J.: Limits of propositional logic. University of Rochester, https://www.cs.rochester.edu/u/nelson/courses/csc_173/proplogic/limits.html

[387] Zheng, A.: Evaluating Machine Learning Models: A Beginner's Guide to Key Concepts and Pitfalls. O'Reilly Media (2015), https://books.google.com.lb/books?id=OFhauwEACAAJ

[388] Zheng, A.: Evaluation Metrics, chap. 2, p. Refer to the book for specific page numbers. O'Reilly Media (2015), https://www.oreilly.com/library/view/evaluating-machine-learning/9781492048756/ch02.html

[389] Zhihong, Z., Mingtian, Z.: Web ontology language OWL and its description logic foundation. In: Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies (2003). doi: 10.1109/PDCAT.2003.1236278

[390] Zhou, D., Jin, X., Lian, X., Yang, L., Xue, Y., Hou, Q., Feng, J.: Autospace: Neural architecture search with less human interference. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 327–336 (2021). doi: 10.1109/ICCV48922.2021.00039

[391] Zhu, J., Wang, H., Tsou, B.K.: A density-based re-ranking technique for active learning for data annotations. In: Computer Processing of Oriental Languages. Language Technology for the Knowledge-based Economy: 22nd International Conference, ICCPOL 2009, Hong Kong, March 26-27, 2009. Proceedings 22. pp. 1–10. Springer (2009)

[392] Zhu, J., Wang, H., Tsou, B.K., Ma, M.: Active learning with sampling by uncertainty and density for data annotations. IEEE Transactions on audio, speech, and language processing **18**(6), 1323–1331 (2009)

[393] Zin, M.M., Nguyen, H.T., Satoh, K., Sugawara, S., Nishino, F.: Improving translation of case descriptions into logical fact formulas using legalcasener. In: Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law. p. 462–466. ICAIL '23, Association for Computing Machinery, New

York, NY, USA (2023). doi: 10.1145/3594536.3595141, https://doi.org/10.1145/3594536.3595141