DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND RESEARCH
University of Blida 1
Faculty of Sciences
Computer Science Department

DOCTORAL THESIS

Option : Software Engineering

Author : Oussama Hamel

Theme

Uncertainty in Linked Data

The Examining Committee for this dissertation consists of:

Pr. BOUSTIA Narhimene	University of Blida 1	President
Pr. BOUKHALFA Kamel	ENSIA	Examiner
Dr. LOUKAM Mourad	University of UHBC, Chlef	Examiner
Dr. MADANI Amina	University of Blida 1	Examiner
Dr. FAREH Messaouda	University of Blida 1	Supervisor

Date: **25/02/2025**

I would like to dedicate this thesis to my beloved brother, Nourredine. His memory has been a source of strength and inspiration throughout this journey.

Acknowledgement

I would like to begin by expressing my deepest gratitude to Allah for His guidance and blessings throughout this journey.

My heartfelt thanks go to my supervisor, Dr. Fareh, for her unwavering support, insightful feedback, and encouragement. Her guidance has been instrumental in the completion of this thesis.

I am also grateful to the jury members for their time, valuable input, and constructive critiques, which have significantly contributed to the refinement of my work.

My appreciation extends to the professors and administration of the Computer Science Department and the Faculty of Science for their support and for providing an excellent academic environment.

I would like to thank my parents, whose love and encouragement have been a constant source of strength. To my brothers, Redouane and Mohamed, my sister Asma, my nephews Youcef, Anes, Akram, and Abderrahmane, and my nieces Nourelhouda and Lylia, your support and belief in me have been invaluable.

I am deeply grateful to a special person, Imene Senhadji, for her presence and encouragement throughout this journey. Her presence has been a source of great comfort and motivation.

I am grateful to my entire family for their patience, understanding, and continuous support throughout this endeavor.

To my friends, your camaraderie and encouragement have been a source of motivation and joy.

I would like to express my gratitude to my colleagues at the Bank of Algeria. Your support has been invaluable throughout this journey.

Finally, I want to extend my gratitude to everyone who has contributed to the realization of this thesis. Your help and support have made this achievement possible.

Oussama Hamel

Abstract

The semantic web and linked data have become essential in modern data management, providing valuable insights by interconnecting diverse datasets. However, the quality and completeness of linked data remain significant challenges due to uncertainties such as integration errors, ambiguities in semantic relationships, and incomplete data. This thesis addresses these issues through advanced deep learning techniques. We propose four main contributions: (1) LinkED-S2S, a sequence-to-sequence model incorporating an embedding layer and an attention mechanism for detecting missing semantic links in RDF (Resource Description Framework) datasets; (2) An encoder-decoder model based on an embedding layer and an attention mechanism for predicting missing types of RDF entities; (3) SASNN, a Siamese neural network for detecting sameAs links, aimed at improving entity alignment within the dataset and across multiple datasets; and (4) A model relying on three LSTM (Long Short Term Memory) neural networks and embedding layers for detecting erroneous triples in linked data. Each contribution has been tested individually, achieving promising results on several benchmark datasets, demonstrating the effectiveness of our methods. Additionally, our contributions address scalability issues and consider the semantic relationships between entities. A case study on the UniProt dataset further validates the combined application of these contributions in generating and verifying triples, enabling the resolution of incompleteness and errors in linked data.

Keywords: Semantic Web, Linked Data, Uncertainty, Incompleteness, Resource Description Framework, Deep Learning

Résumé

Le Web sémantique et les données liées sont devenus essentiels dans la gestion moderne des données, offrant des informations précieuses en interconnectant des ensembles de données variés. Cependant, la qualité et la complétude des données liées demeurent des défis majeurs en raison des incertitudes, telles que les erreurs d'intégration, les ambiguïtés dans les relations sémantiques et les données incomplètes. Cette thèse aborde ces problématiques à travers des techniques avancées d'apprentissage profond. Nous proposons quatre contributions principales : (1) LinkED-S2S, un modèle séquence-à-séquence intégrant une couche d'incorporation et un mécanisme d'attention pour la détection de liens sémantiques manquants dans les ensembles de données RDF ; (2) Un modèle encodeur-décodeur basé sur une couche d'incorporation et un mécanisme d'attention pour prédire les types manquants des entités RDF ; (3) SASNN, un réseau de neurones siamois pour la détection de liens sameAs, visant à améliorer l'alignement des entités dans le jeu de données et entre plusieurs ensembles de données ; et (4) Un modèle reposant sur trois réseaux de neurones LSTM et des couches d'incorporation pour la détection des triples erronés dans les données liées. Chaque contribution a été testée individuellement, obtenant des résultats prometteurs sur plusieurs ensembles de données de référence, ce qui démontre l'efficacité de nos méthodes. De plus, nos contributions traitent des problèmes de scalabilité et prennent en considération les relations sémantiques entre les entités. Une étude de cas sur le jeu de données UniProt valide l'application combinée de ces contributions dans la génération et la vérification des triples, permettant de résoudre l'incomplétude et les erreurs dans les données liées.

Mots Clée : Web sémantique, Données liées, Incertitude, Incomplétude, Cadre de Description des Ressources, Apprentissage profond

ملخص

لقد أصبح الويب الدلالي والبيانات المرتبطة أساسياً في إدارة البيانات الحديثة، حيث يوفر رؤى قيمة من خلال ربط مجموعات بيانات متنوعة. ومع ذلك، تظل جودة واكتمال البيانات المرتبطة تحديات كبيرة بسبب عدم اليقين، مثل أخطاء التكامل، والغموض في العلاقات الدلالية، والبيانات غير المكتملة. تتناول هذه الأطروحة هذه القضايا من خلال تقنيات التعلم العميق المتقدمة. نقترح أربع مساهمات رئيسية: (١)، نموذج تسلسل إلى تسلسل يدمج طبقة تضمين وآلية انتباه لاكتشاف الروابط الدلالية المفقودة في مجموعات بيانات ؛ (٢) نموذج مشفر-مفكك يعتمد على طبقة تضمين وآلية انتباه للتنبؤ بالأنواع المفقودة من الكيانات ؛ (٣) ، شبكة عصبية سيامية لاكتشاف روابط ، تهدف إلى تحسين توافق الكيانات ضمن مجموعة البيانات وعبر مجموعات بيانات متعددة؛ و(٤) نموذج يعتمد على ثلاثة شبكات عصبية وطبقات تضمين لاكتشاف الثلاثيات الخاطئة في البيانات المرتبطة. تم اختبار كل مساهمة بشكل فردي، محققة نتائج واعدة على عدة مجموعات بيانات مرجعية، مما يثبت فعالية أساليبنا. بالإضافة إلى ذلك، تعالج مساهماتنا مشكلات قابلية التوسع وتأخذ في الاعتبار العلاقات الدلالية بين الكيانات. تؤكد دراسة حالة على مجموعة بيانات التطبيق المشترك لهذه المساهمات في توليد والتحقق من الثلاثيات، مما يمكن من حل مشاكل عدم الاكتمال والأخطاء في البيانات المرتبطة.

الكلمات المفتاحية: الويب الدلالي، البيانات المرتبطة، عدم اليقين، النقص، إطار توصيف الموارد، التعلم العميق

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Acronyms / Abbreviations

ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
AM	Attention Mechanism
BI-LSTM	Bidirectional Long Short-Term Memory
BRNN	Bidirectional Recurrent Neural Network
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IRI	Internationalized Resource Identifier
KG	Knowledge Graph
LOD	Linked Open Data
LSTM	Long Short-Term Memory
ML	Machine Learning
MRR	Mean Reciprocal Rank
OWL DL	Web Ontology Language Description Logic
OWL	Web Ontology Language
RAM	Random Access Memory

RDF	Resource Description Framework
RDFS	RDF Schema (Resource Description Framework Schema)
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SPARQL	SPARQL Protocol and RDF Query Language
SVM	Support Vector Machine
Tanh	Hyperbolic Tangent
URW3-XG	Uncertainty Reasoning for the World Wide Web Incubator Group
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language

General Introduction

Research Context

In recent years, a significant evolution has occurred in the web, particularly with the advent of the Semantic Web and Linked Data [1]. The Semantic Web aims to enhance the way data is shared and reused by promoting the use of standardized data formats that facilitate machine processing and enable the generation of new insights through reasoning. By embedding meaning into data, the Semantic Web transforms the conventional web into a more interconnected and intelligent space where data can be easily interpreted and utilized by both humans and machines.

Linked Data, as a method of publishing data, plays a crucial role in this transformation. It promotes the automated processing of information and the establishment of connections between various data sources [2]. Through the principles of Linked Data, datasets are not merely isolated entities but are interlinked to form a vast, interconnected web of information. This interconnectedness allows for more efficient and comprehensive data retrieval, as information from disparate sources can be seamlessly combined to address complex queries.

The significance of Linked Data lies in its ability to facilitate the integration and interoperability of diverse datasets. For instance, when data is made open, it means that access to and reuse of this data are not restricted by proprietary barriers. Open data supports transparency, innovation, and the creation of new knowledge by enabling a broader audience to engage with and build upon existing datasets.

The Semantic Web and Linked Data are not focused on linking documents in the traditional sense of the web but rather on connecting the objects or entities contained within these documents. This shift from document-centric linking to entity-centric linking represents a fundamental change in how information is organized and accessed. It opens up new possibilities for data integration, contextual understanding, and knowledge discovery across various domains.

Real-world applications of these concepts are numerous and varied. In healthcare,

for example, Linked Data can integrate patient records from different institutions to provide a comprehensive view of a patient's health history, leading to better-informed medical decisions. In the financial sector, Linked Data can enhance the transparency and traceability of transactions, improving fraud detection and regulatory compliance. Similarly, in the research domain, Semantic Web technologies facilitate the aggregation of scientific data from multiple sources, enabling more robust and interdisciplinary analyses.

Overall, the development of the Semantic Web and Linked Data represents a paradigm shift towards a more interconnected, intelligent, and accessible web, paving the way for innovative applications and enhanced data-driven insights across numerous fields.

Types of Uncertainty

Uncertainty in Linked Data manifests in several forms [3]:

- Incompleteness: Occurs when expected data values are missing, resulting in gaps. For instance, an entity lacks essential property values, affecting the dataset's comprehensiveness. This can stem from missing types of entities, absent sameAs links between entities, or missing semantic links, which undermine the dataset's completeness and connectivity.
- Imprecision: Refers to data values that are not precisely defined or contain inherent vagueness, such as dates expressed as ranges or measurements with error margins.
- Ambiguity: Arises when data values can be interpreted in multiple ways, such as homonyms or synonyms, complicating data interpretation.
- Errors: These involve incorrect or inaccurate data values that do not reflect the true information, leading to misinformation and faulty conclusions in data analysis.

Consequences of Uncertainty

The impact of these uncertainties includes:

- Reduced Accuracy and Consistency: Applications relying on Linked Data experience diminished accuracy and consistency due to incomplete or imprecise information.
- Integration Challenges: Integrating data from various sources becomes problematic due to inconsistencies caused by different types of uncertainty. For instance, incomplete healthcare data can lead to flawed medical decisions or skewed research outcomes.

- Limited Exploitation Potential: The full potential of Linked Data for analysis and knowledge discovery is constrained by these uncertainties. In clinical research, missing or inaccurate data can hinder research outcomes and practical applications.

Sources of Errors and Incompleteness

Several factors contribute to errors and incompleteness in Linked Data:

- Erroneous Data Integration: Integrating datasets with flawed information can introduce errors.
- Concept Alignment Issues: Problems during ontology alignment can lead to errors in data integration.
- Ambiguity from Imperfect Interlinking: Imperfect linking between datasets can cause ambiguity.
- Imprecise Data Representation: Data with implicit measurements or heterogeneous formats contributes to imprecision [3, 4].
- Outdated Data: Datasets that are not updated automatically can lead to inaccuracies.
- Heterogeneity: Differences in data formats, models, or structures across datasets create integration challenges.
- Use of Ambiguous Relations: Utilizing vague or unclear relations between datasets can introduce confusion.
- Automatically Generated Data: Data automatically derived from other dataset formats contains errors.
- Human Errors: Mistakes made during data input or processing contribute to inaccuracies.

Existing approaches for detecting and managing errors and incompleteness in Linked Data have limitations:

- Several methods are limited to handling specific link types or certain aspects of dataset alignment, like sameAs links or the prediction of missing types, which results in gaps and incompleteness in the linking process [5, 6].

- Some approaches generate incorrect links, compromising data quality.
- Some approaches overlook semantic relations between triples, focusing instead on textual descriptions [7].
- Certain neural network-based methods fail to prioritize more meaningful data effectively [8, 9].
- Scalability is another challenge, as solutions based on probability theory and possibility theory do not adequately address this issue [10].
- Many methods rely on manual correction of inconsistencies, which can be inefficient [11].

Problem Statement

With the increasing adoption of Linked Data and open data principles, a wealth of datasets has become available on the web, significantly advancing the Semantic Web and its applications, such as recommendation systems and knowledge graphs [12]. Linked Data aims to share and interconnect structured data on the web, creating a unified data space where information can be linked and enriched through connections with other datasets. Linked Open Data (LOD) exemplifies this approach, allowing for the publication and interconnection of structured data in line with Linked Data principles. Despite its potential, the publication and integration of Linked Data present substantial challenges related to data quality, primarily due to inherent uncertainties.

Our research is focused on addressing critical challenges in Linked Data to improve its quality and completeness. We aim to:

1. Address Incompleteness: Enhance the comprehensiveness of RDF datasets by generating additional triples to fill gaps in the data. This will help provide more robust and complete results for users relying on RDF datasets.
2. Detect and Correct Errors: Identify and remove erroneous triples to ensure the consistency and accuracy of Linked Data. This includes detecting inaccuracies and applying corrective measures to improve dataset quality.

Research Methodology

Deep learning offers a promising solution to the challenges of uncertainty in Linked Data due to its capacity to handle complex and large-scale datasets effectively. One of the

key advantages of deep learning is its ability to automatically extract features and learn representations from data, enabling it to identify and model intricate patterns and relationships. This capability is particularly beneficial for addressing issues such as data incompleteness and errors. Advanced deep learning models can process vast amounts of data, improving their performance and accuracy over time. They are also well-suited for scalable applications, allowing efficient processing of extensive datasets and handling high-dimensional information. Furthermore, deep learning techniques demonstrate robustness in dealing with ambiguous and imprecise data, making them effective in refining data quality and enhancing the accuracy of link discovery and integration tasks.
Our research introduces several innovative contributions to achieve our objectives, as shown in Figure 0.1:

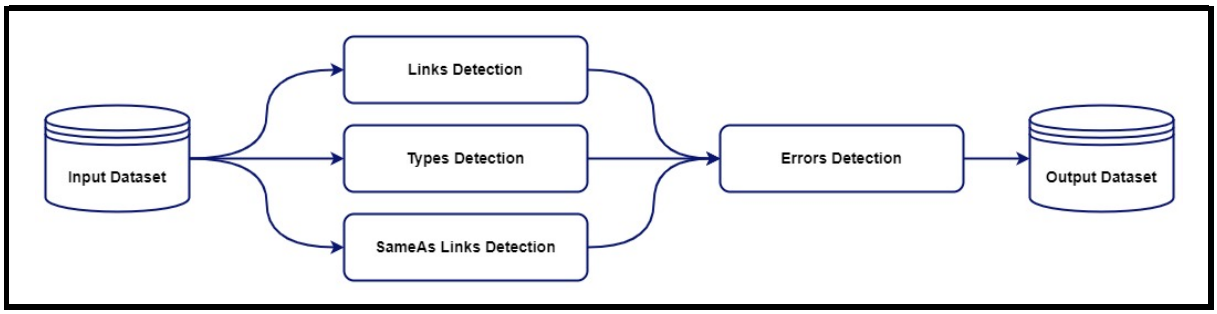


Figure 0.1: Research Methodology

1. LinkED-S2S: A Deep Learning Approach for Missing Links Detection: We propose a method for detecting missing links between resources in RDF datasets, which aims to reduce dataset incompleteness and provide more comprehensive results. Our approach employs an encoder-decoder model enhanced by an attention mechanism. This model can uncover various hidden relationships between elements in RDF datasets, processing long data sequences effectively. The attention mechanism prioritizes relevant data points, improving link prediction accuracy.
   The key contribution is the introduction of LinkED-S2S, an innovative Encoder-Decoder Model with an Attention Mechanism designed for link discovery in Linked Data. Unlike existing methods that focus on specific link types, LinkED-S2S aims to discover all types of missing links, addressing a broader range of incompleteness issues. The model facilitates the discovery of missing links not only between different datasets but also within a single dataset, providing a more thorough identification of connections. By leveraging deep learning techniques, our approach goes beyond traditional methods, offering a more accurate discovery of hidden relationships. The attention mechanism focuses the model on the most relevant data points, enhancing

prediction accuracy and efficiency. We evaluated LinkED-S2S against state-of-the-art models and achieved encouraging results, demonstrating its effectiveness in link discovery tasks.

2. Type Prediction for RDF Entities: Our approach addresses type incompleteness in RDF datasets by predicting missing types for entities based on their predicates and objects. We use an encoder-decoder model with an attention mechanism to enhance type prediction accuracy. This method leverages deep learning techniques suitable for handling large datasets, making it effective for Linked Data analysis.
   In this contribution, we propose an encoder-decoder network for multi-labeling, incorporating an attention mechanism to model links between data and predict missing types. This approach utilizes advanced deep learning methods to handle large-scale datasets and enhance type prediction accuracy. We applied this approach to the DBpedia[1] dataset, showcasing its practical applicability. The evaluation of this method against existing models yielded promising results, confirming its potential in improving RDF data quality.

3. Siamese Neural Networks for SameAs Links Detection: We develop a Siamese neural network model for detecting sameAs links, which improves accuracy and efficiency in aligning similar entities. Our approach addresses the challenges posed by Linked Data's incompleteness and heterogeneity.
   The principal contributions include enhancing link discovery accuracy through Siamese neural networks, which capture intricate patterns and dependencies in linked data, leading to more reliable link predictions. The model benefits from automatic feature learning, eliminating the need for manual feature engineering, and adapts to various types of Linked Data. By incorporating contextual information such as entity attributes and relationships, the model improves its ability to distinguish between valid and erroneous links. The approach is also scalable and efficient, making it suitable for real-world applications with extensive datasets. Additionally, Siamese networks support end-to-end learning, learning the optimal link detection function directly from the data. The versatility of Siamese networks allows them to be trained on diverse types of Linked Data, including text, images, and structured data. The Siamese network approach was evaluated against state-of-the-art models, demonstrating enhanced accuracy and contextual awareness in sameAs link detection.

4. Error Detection Using LSTM Networks: To address challenges in Linked Data, we introduce a novel deep learning approach for error detection using Long Short-Term

[1] https://downloads.dbpedia.org/

Memory (LSTM) networks. This model captures long-term dependencies and patterns within RDF triples, offering a scalable solution for various datasets.
The main contributions of this work include the development of an LSTM-based architecture to model complex RDF triple sequences. This approach uses embedding layers to learn dense representations of entities and relations, providing a generalized solution applicable to different Linked Data sets. Our model is designed to handle large datasets effectively and considers semantic links between entities to enhance error detection capabilities. The LSTM-based error detection method was tested against state-of-the-art approaches, achieving notable improvements in dataset quality.

Our research introduces a unified framework (Figure 0.1) designed to resolve uncertainty in Linked Data by integrating several innovative contributions. This comprehensive framework aims to address various dimensions of uncertainty within Linked Data environments. In order to test all these contributions together, we conducted a case study on the UniProt dataset[2]. We utilized the first three contributions—LinkED-S2S for generating additional triples, the type prediction model for enhancing type completeness, and the Siamese neural network for detecting and aligning similar entities. The fourth contribution, the LSTM-based error detection model, was applied to identify correct triples and remove false ones, ensuring the overall quality of the enriched dataset. This integrated approach demonstrated the effectiveness of our methods in improving Linked Data completeness and accuracy.

Manuscript Outline

The structure of the manuscript is designed to provide a comprehensive overview of the research and its contributions. The outline, as illustrated in Figure 0.2, is divided into two main parts: a literature review and a detailed presentation of contributions. Each part is carefully organized to build the foundation for the research and to highlight the advancements made in addressing the challenges of Linked Data.

Part1: Literature Review

1. Fundamentals of the Semantic Web and Linked Data: This chapter introduces the foundational concepts of the Semantic Web and Linked Data, providing an overview of their principles, technologies, and standards. It covers the architecture of the

[2] https://www.uniprot.org/help/downloads

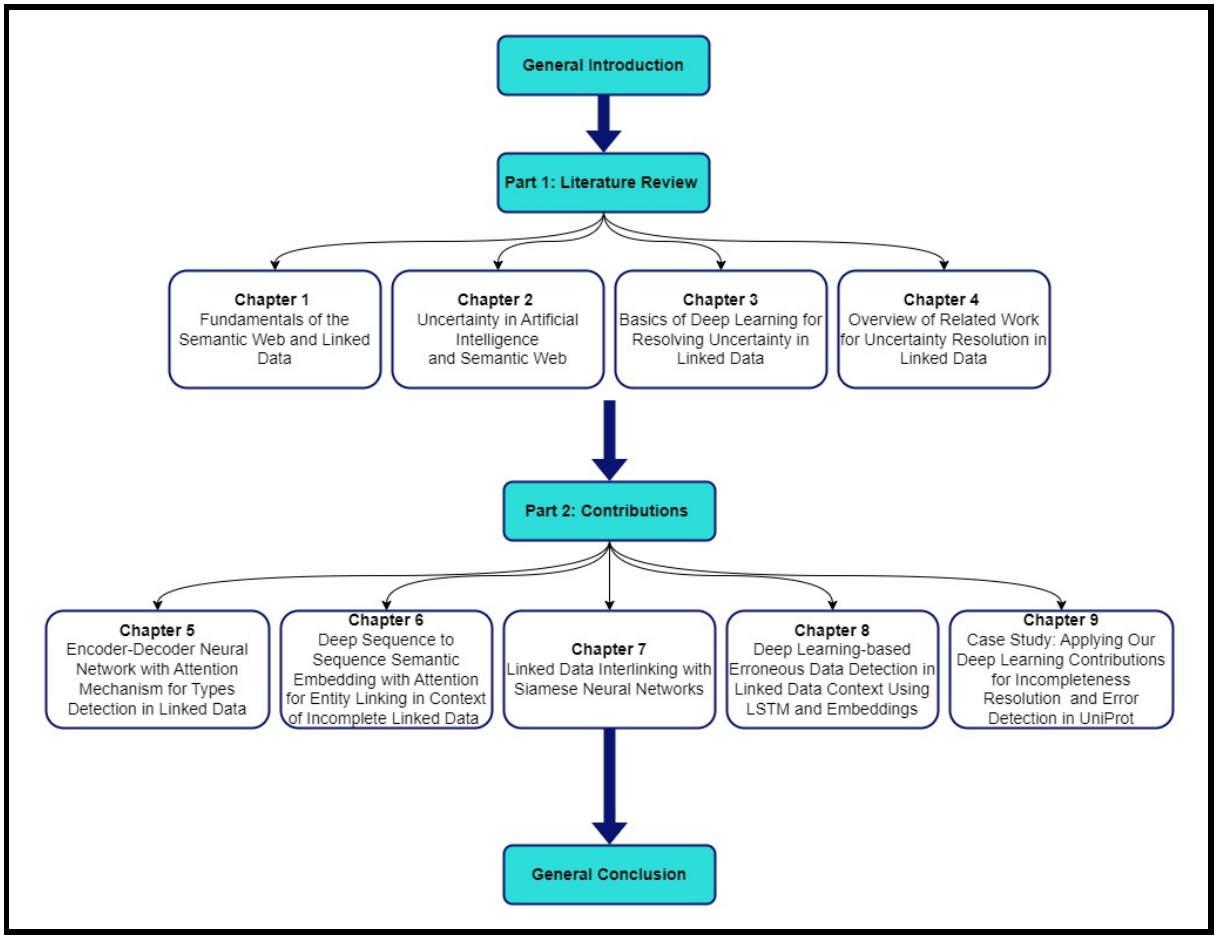


Figure 0.2: Manuscript Outline

Semantic Web, including key elements such as RDF, OWL, and SPARQL, and explains how Linked Data facilitates the interlinking of data across different sources.

2. Uncertainty in Artificial Intelligence and Semantic Web: Here, the focus shifts to the concept of uncertainty within the context of Artificial Intelligence. This chapter explores various types of uncertainty, including epistemic and aleatory uncertainty, and examines how they impact AI models and decision-making processes. It also discusses methods for managing and mitigating uncertainty in AI systems.

3. Basics of Deep Learning for Resolving Uncertainty in Linked Data: This chapter provides an overview of deep learning, including its basic principles, architectures, and techniques. It covers key concepts such as neural networks, backpropagation, activation functions, and optimization methods. The chapter also introduces various deep learning models and their applications in different domains.

4. Overview of Related Work for Uncertainty Resolution in Linked Data: This chapter

reviews existing research related to uncertainty in Linked Data, specifically focusing on issues of incompleteness and errors. It assesses current methods and approaches for addressing these challenges and identifies gaps in the research that the current study aims to fill. This review sets the stage for the contributions of the thesis by highlighting the need for more effective solutions.

Part2: Contributions

1. Encoder-Decoder Neural Network with Attention Mechanism for Types Detection in Linked Data: In this chapter, a method for predicting missing types for RDF entities is discussed. The approach leverages an encoder-decoder model and attention mechanism to enhance the accuracy of type prediction based on the predicates and objects associated with entities.

2. Deep Sequence to Sequence Semantic Embedding with Attention for Entity Linking in Context of Incomplete Linked Data: This chapter presents a novel approach for detecting missing links between resources in RDF datasets. It describes the proposed method, which utilizes a deep learning encoder-decoder model with an attention mechanism to detect hidden relationships and improve the completeness of Linked Data.

3. Linked Data Interlinking with Siamese Neural Networks: This chapter introduces a Siamese neural network model for detecting sameAs links in Linked Data. It highlights how this model improves accuracy and efficiency in aligning similar entities and addresses the challenges posed by the heterogeneity and incompleteness of Linked Data.

4. Deep Learning-based Erroneous Data Detection in Linked Data Context Using LSTM and Embeddings: This chapter focuses on a deep learning approach for detecting errors in RDF datasets using Long Short-Term Memory (LSTM) networks. It explains how the model captures long-term dependencies and patterns to identify and remove erroneous triples, improving the overall quality of the data.

5. Case Study: Applying our Deep Learning Contributions for Incompleteness Resolution and Error Detection in UniPro: The final chapter presents a case study applying the proposed methods to the UniProt dataset. It demonstrates how the first three contributions—link detection, type prediction, and sameAs link detection—are used to generate additional triples, while the fourth contribution is employed to identify

and correct errors. This comprehensive case study validates the effectiveness of the contributions in improving Linked Data quality and completeness.

General Conclusion: The concluding chapter synthesizes the overall findings of the research and assesses their implications for managing uncertainty in Linked Data. It summarizes our contributions and their impact on enhancing data quality and completeness. The chapter also outlines future research directions, including potential improvements to the framework and applications to other datasets.

Publications Related to This Thesis

In the context of this thesis, several works have been published that contribute to the research objectives and advancements discussed. Each publication reflects different elements of the research and provides insights into the progress made in addressing the challenges of Linked Data.

1. Hamel, O., & Fareh, M. (2024). Deep sequence to sequence semantic embedding with attention for entity linking in context of incomplete linked data. Engineering Applications of Artificial Intelligence, 134, 108689.

2. Hamel, O., & Fareh, M. (2022, September). Missing types prediction in linked data using deep neural network with attention mechanism: Case study on dbpedia and uniprot datasets. In Special Sessions in the Advances in Information Systems and Technologies Track of the Conference on Computer Science and Intelligence Systems (pp. 212-231). Cham: Springer Nature Switzerland.

3. Hamel, O., & Fareh, M. (2022, September). Encoder-decoder neural network with attention mechanism for types detection in linked data. In 2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS) (pp. 733-739). IEEE.

4. Hamel, O., & Fareh, M. (2024). Deep Learning-based Erroneous Data Detection in Linked Data Context Using LSTM and Embeddings. In 2024 International Conference on Advances in Electrical and Communication Technologies (ICAECOT24).

5. Hamel, O., & Fareh, M. (2024). A Deep Learning-Based Framework for Handling Incompleteness and Detecting Errors in Linked Data Applied to the UniProt Dataset. In 8th International Artificial Intelligence and Data Processing Symposium (IDAP'24), Sept 21-22, 2024, Malatya, Turkiye.

Part I

Literature Review

Chapter 1

Fundamentals of the Semantic Web and Linked Data

1.1 Introduction

The Semantic Web, proposed by the W3C, is an extension of the current web that adds a layer that enables reasoning about data and consideration of semantics. Unlike the current version, known as the "web of documents," where search engines rely on syntactic measures during search operations, the Semantic Web focuses on meaningful connections.

The implementation of the Semantic Web necessarily involves linking data (the implementation of the web of data) and using various W3C standards for data representation, schema representation, inference, and more. This version of the web is expected to enhance the quality of available services, particularly in applications such as e-commerce and e-government.

Today, numerous researchers are involved in this project, working to propose recommendations for resolving various issues related to the Semantic Web. In this chapter, we explain the characteristics of different web versions, especially the Semantic Web and Linked Data, by citing the various standards proposed by the W3C. We also provide a comparative synthesis of these different versions. Finally, we discuss the challenges and difficulties encountered in realizing this project, with a particular focus on the notion of uncertainty in linked data.

1.2 World Wide Web

The web, since its inception, has transformed the way we access and share information, fundamentally altering communication, commerce, and daily life. From static pages to

dynamic interactions and intelligent systems, the web's evolution reflects our growing technological capabilities and the increasing complexity of our digital needs. Understanding the different versions of the web, from its early stages to the emerging Semantic Web, provides insight into how far we've come and where we're headed. This section explores the definition, evolution, and characteristics of the web's various generations, shedding light on its continuous reinvention to meet ever-changing demands.

1.2.1 Definition of the Web

The World Wide Web (WWW), or simply the Web, was invented by Tim Berners-Lee. It is a hypertext system on the internet that presents hyperlinks between pages, allowing access to various interconnected pages within this web [13].

1.2.2 Versions of the Web

Since its creation, the web has continuously evolved to meet new needs, leading to the emergence of different versions or generations of the web: Web 1.0 (static web), Web 2.0 (current web), and Web 3.0 (still in development). Below, we outline the main characteristics of each of these versions [14].

1.2.2.1 Web 1.0

Web 1.0, or the traditional web, enabled companies to share information and users to find it without the ability to interact. The loading times were quite slow in this version. Later, with the advent of scripting languages and the development of browsers, this version evolved into a dynamic web.

1.2.2.2 Web 2.0

This current version of the web connects documents. It allows interaction between users (social web, e-commerce, etc.) thanks to technological advancements that have enabled the use of various types of applications through a browser.

1.2.2.3 Web 3.0

Web 3.0, also known as the Semantic Web, is still under development. It aims to add a layer of semantics (metadata) to the current web by using various standards proposed by the W3C to ensure interoperability. This version will connect resources rather than documents. Software and agents were proposed to utilize the diverse data published on the web and reason over it to produce new information.

1.3 Current Web

Today's web is considered syntactic due to the absence of a semantic layer. In this version, only users can interpret (provide semantics to) the various web pages.

The interaction between users and web pages occurs via search engines, which rely on syntactic aspects during searches. Several issues persist in this version, summarized as follows:

- Absence of data/page descriptions in HTML pages.
- Data format does not allow for reasoning.
- Page content is inaccessible to machines.
- Many information and services are hidden in HTML pages because search engines do not perform reasoning on textual content.
- Irrelevant search results, as searches are based on keywords (syntactic search).

To address the issues above, the Semantic Web has been proposed. It involves using metadata to annotate data in a standard language and vocabulary (proposed by the W3C). This solution is detailed in the following section.

1.4 Semantic Web

The evolution of the web has brought us to an era where data interconnectivity and understanding are paramount. The Semantic Web and Linked Data are revolutionary concepts that aim to transform how we interact with and derive meaning from web data. By adding semantic context to data, these technologies enable machines and humans to process information more intelligently and intuitively. This section delves into the definitions, distinctions, and implications of the Semantic Web and Linked Data, highlighting their potential to revolutionize data comprehension and utilization.

1.4.1 Definition of the Semantic Web

The Semantic Web, also known as the Web of Data, is an advanced version of the current web. It incorporates a semantic layer that provides meanings to various data, enabling both machines and people to reason about this data [15].

Table 1.1: Multi-criteria Comparison Between the Current Web and the Semantic Web

Criteria	**Current Web**	**Semantic Web**
Representation	Set of documents	Set of knowledge
Search Method	Keyword search	Concept search
Usage	Human use	Machine and human use
Data Processings	No reasoning on data	Data exploitation (reasoning)
Standards	HTML standard	XML and RDF standards
Formalization	Lack of formalization	Knowledge formalization
Annotation Richness	Less rich annotation	Rich annotation

1.4.2 Current Web vs. Semantic Web

In this section, we will highlight the differences between these two versions of the web. Table 1.1 provides a summary of this comparison across different criteria.

The Semantic Web offers more possibilities and greater freedom for reasoning about data, allowing for the extraction of new knowledge.

Figure 1.1 demonstrates the difference between the representations of the current web and the Semantic Web. It clearly shows that the Semantic Web representation offers greater richness in the links between resources.

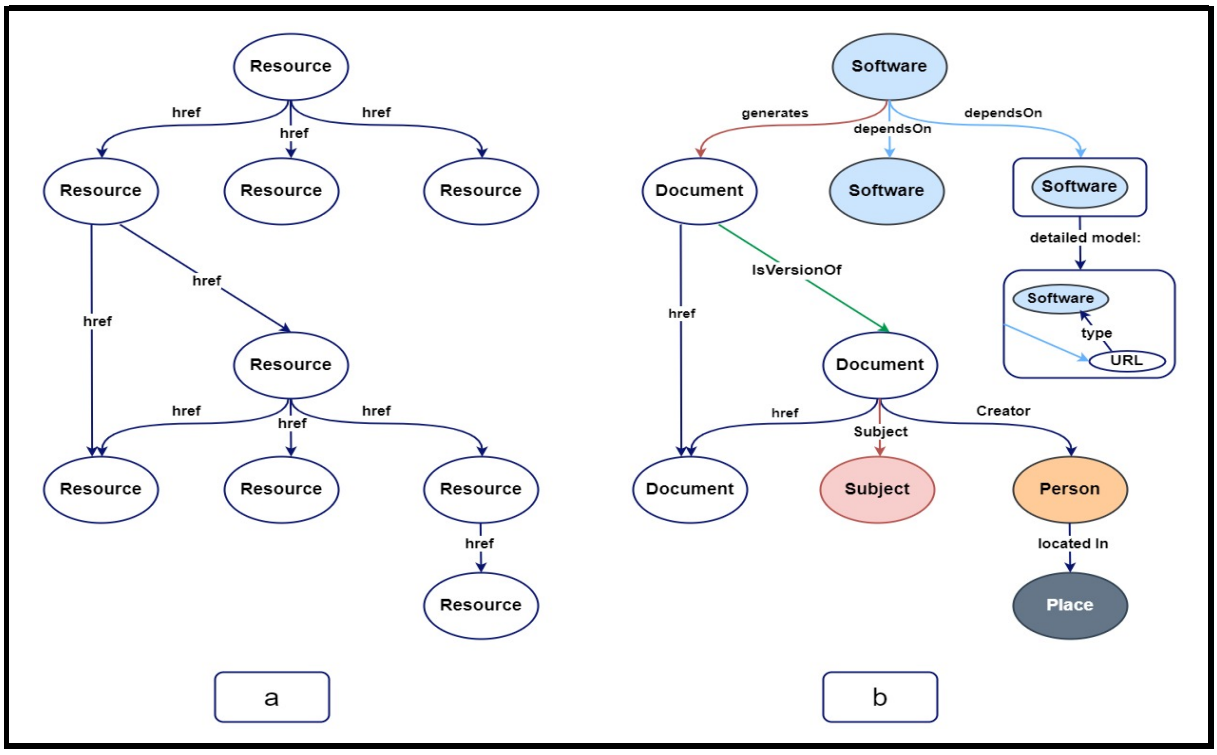


Figure 1.1: a. Current Web, b. Semantic Web [16]

1.4.3 Web of Data

The Web of Data enables the structuring and linking of information published on the web, allowing machines to process this information to extract and generate new knowledge [17]. Several objectives of the Web of Data can be outlined:

- Enhanced Interoperability: Through standards provided by the W3C, the Web of Data facilitates the representation of data, making it more interoperable across different systems.
- Machine Processing and Interpretation: It allows for more effective data processing and interpretation by machines.
- Improved Reasoning Quality: By linking data, it enhances the quality of reasoning about the data.
- Data Sharing and Reuse: It promotes the sharing and reuse of data.

This new version of the web, known as the Semantic Web, shifts from a document-centric web to a web of linked data.

Figure 1.2 illustrates the level of richness in the relationships between the different objects in each type of web mentioned above: the web of documents and the web of linked data.

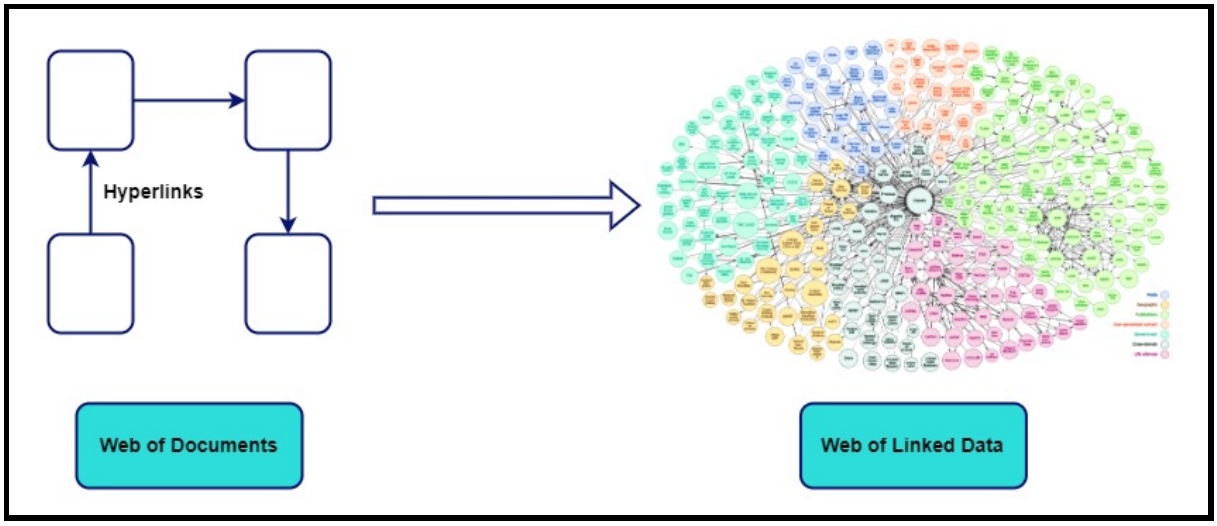


Figure 1.2: Web of documents vs. web of linked data

1.4.4 Open Data

Open Data refers to public datasets that are not linked to other sources. In contrast, Linked Data represents a design principle that creates connections between datasets published on the web in RDF format (rather than linking documents). This allows machines to navigate the web and discover additional data through these links [1]. In this version of the web, various objects are identified using URIs (Uniform Resource Identifiers).

Tim Berners-Lee has proposed four rules for designing Linked Data, which are detailed below:

- Use of URIs: To uniquely identify objects and concepts.
- HTTP Protocol: To allow human access to websites and data.
- Semantic Web Standards: To provide relevant information about URIs.
- Links to Other Data: To offer more options during exploration and navigation.

These principles help ensure that web data is interconnected, discoverable, and usable, thereby enriching the overall web experience.

1.5 Semantic Web Architecture

The Semantic Web stack represents the architecture of the Semantic Web. This stack illustrates the various functional layers that compose the Semantic Web, where each layer builds upon the layers below it. Figure 1.3 displays the different components of this stack.

We can identify four major functionalities within this stack, encompassing various standards: representation, querying, reasoning, and trust. Figure 1.4 clearly delineates these functionalities.

In the following sections, we will define and explain the key standards proposed by the W3C for the Semantic Web.

1.5.1 Representation Components

1.5.1.1 URI/IRI

URIs (Uniform Resource Identifiers) are used to identify web resources. They follow the format: `scheme:[//authority]path[?query][#fragment]`.

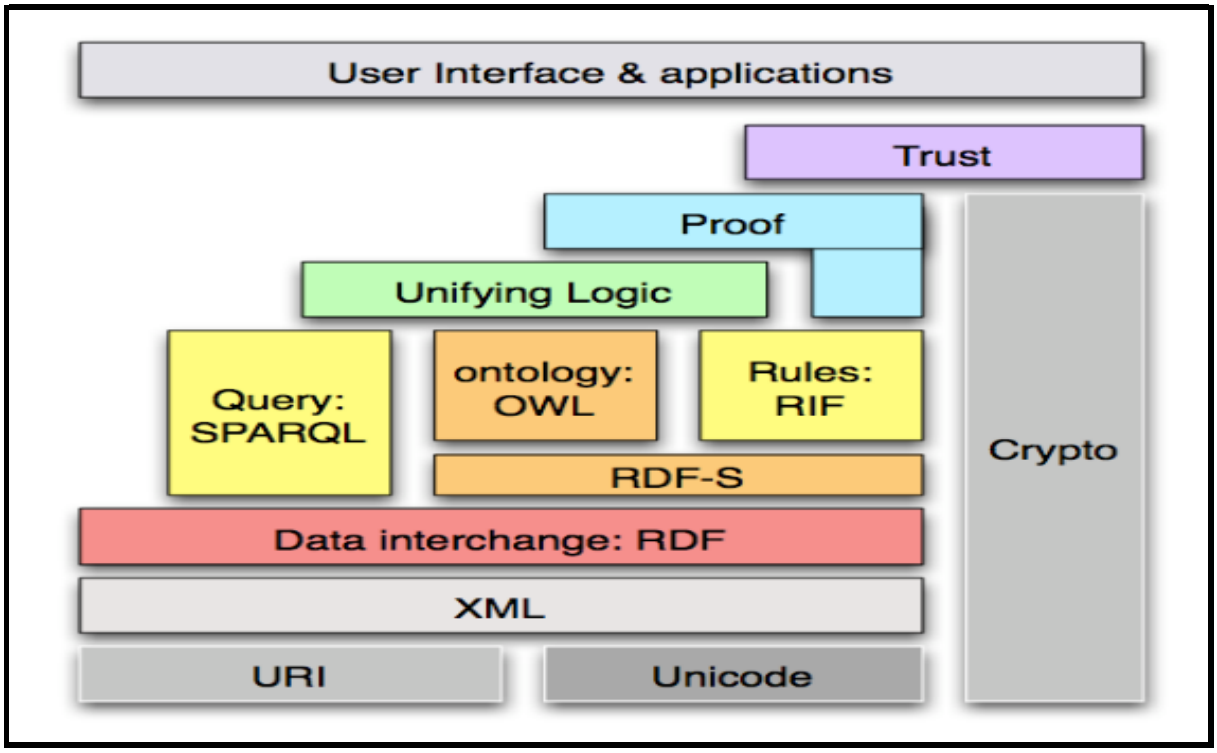


Figure 1.3: Architecture of the Semantic Web [18]

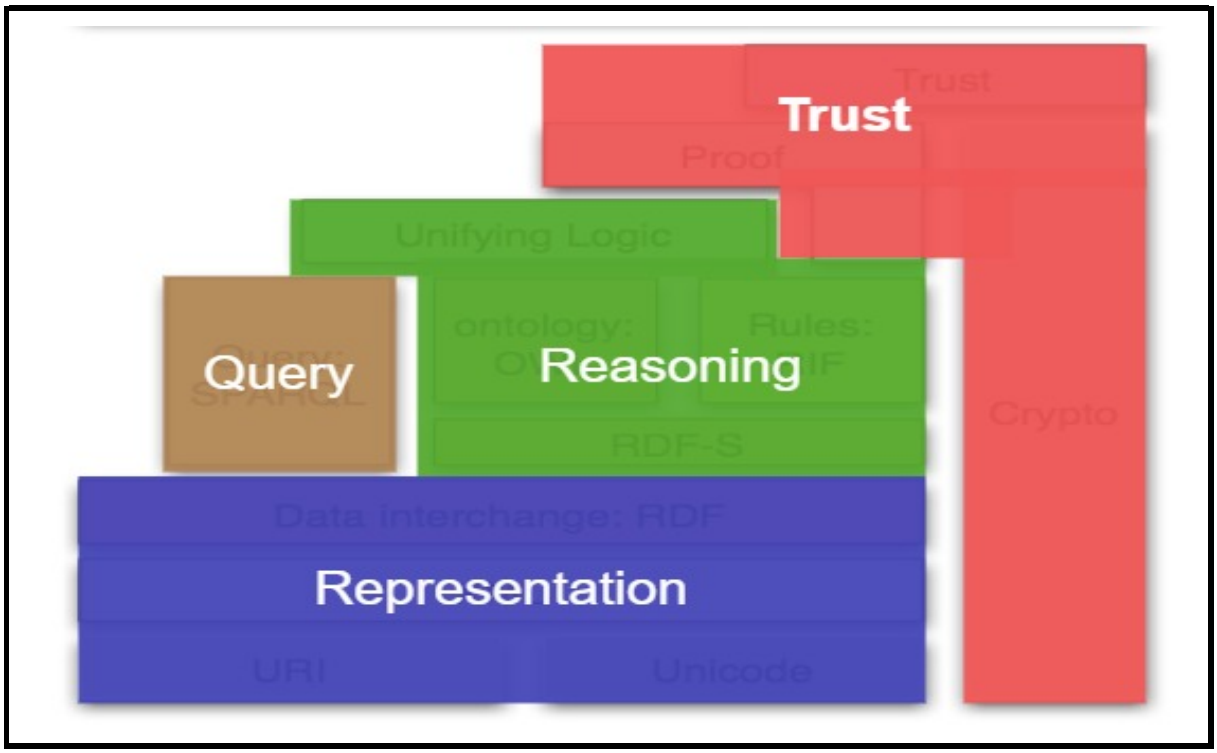


Figure 1.4: Functions of Semantic Web Standards [19]

- Scheme: The most common schemes are http, ftp, data, etc.
- Authority: Consists of three parts (user information, host, and port number).
- Path: Specifies the location of the resource.
- Query: An optional string of characters representing a query.
- Fragment: An optional argument to access a secondary resource.

Example: `<http://www.example.org/ns#>`
IRIs (International Resource Identifiers) extend URIs to include all existing alphabets, allowing for a broader range of resource identification.

1.5.1.2 XML

XML (eXtensible Markup Language) is a W3C recommendation for representing resources. It is a meta-language used to create new document languages. An XML document must adhere to a schema to ensure its structure.
Example: An XML document representing objects of type Person.

```
<personnes>
    <personne>
        <nom>Oussama</nom>
        <service>Informatique</service>
    </personne>
    <personne>
        <nom>Adam</nom>
        <service>RH</service>
    </personne>
    <personne>
        <nom>Ines</nom>
        <service>Commercial</service>
    </personne>
</personnes>
```

Namespaces can be used in XML to enrich the vocabulary. Example: XML document with the namespace “autre”.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<message xmlns='http://example.dz/xml/message'
   xmlns:autre="http://www.autre.com/namespace">
    <expediteur>
        <nom>Oussama</nom>
        <email>abc@example.com</email>
    </expediteur>
    <destinataire>
        <nom>Adam</nom>
        <email>xyz@example.com</email>
    </destinataire>
    <texte>texte du message</texte>
    <autre:titre>titre du message</autre:titre>
    <autre:texte>ceci est un texte</autre:texte>
</message>
```

1.5.1.3 RDF

RDF (Resource Description Framework) [20] is a W3C standard for representing and linking web data and resources. RDF provides a richer representation of data compared to XML.

RDF represents resources using triples (Subject, Predicate, Object), where:

- Subject: A resource identified by a URI.
- Predicate: Expresses the relationship between the subject and the object. It is identified by a URI.
- Object: A resource or literal value related to the subject through the predicate.

Triples can be visualized as graphs, as shown in the following example (Figure 1.5). Example: Representation of (Oussama Hamel is the author of Article.pdf).

- Triple: (Article.pdf, authorOf, Oussama Hamel).
- Graph: Uses URIs to represent the subject and predicate and a literal node for the object.

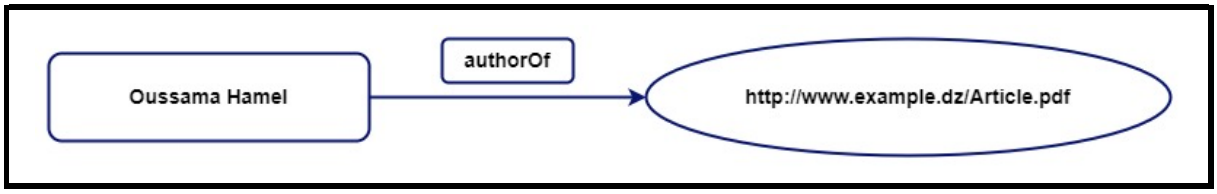


Figure 1.5: Example of an RDF Graph

Several serializations or presentation formats are available to represent RDF triples, including XML, TURTLE, and N-TRIPLES.

The XML serialization for the previous example is shown below.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:exemple="http://exemple.dz/schema#">
     <rdf:Description rdf:about="http://exemple.dz/Rapport.pdf">
         <exemple:auteur>Web</exemple:auteur>
     </rdf:Description>
</rdf:RDF>
```

1.5.2 Query Components

The standard for querying is SPARQL [21], which stands for Simple Protocol and RDF Query Language. Inspired by SQL (using SELECT, FROM, WHERE syntax), SPARQL is designed to query RDF data and RDF schemas (represented in RDFS) to answer specific questions or queries. Results are returned in XML format.

SPARQL also supports operations such as additions, modifications, and deletions of RDF data.

Example: A SPARQL query to retrieve a list of all authors.

```
PREFIX ex: <http://www.example.dz/schema#>
SELECT ?author
WHERE {
    ?Document <http://www.example.dz#author> ?author .
}
```

1.5.3 Reasoning Components

Reasoning is a fundamental aspect of the Semantic Web, enabling sophisticated analysis and interpretation of data based on its semantic structure. This capability allows machines

to infer new information and derive logical conclusions from existing data. The reasoning components of the Semantic Web are built upon ontologies and related standards that formalize knowledge and relationships within specific domains. These components include ontology representation, schema definitions, and advanced reasoning capabilities.

1.5.3.1 Ontology

An ontology represents a set of concepts, terms, and relationships that define the meaning within a specific domain. It is also described as "a formal and explicit specification of a shared conceptualization of a domain of interest" [22].

The primary goal of using ontologies is to formalize knowledge in a particular domain, enabling machines to reason about data. Ontologies serve several purposes:

Defining a common vocabulary for a specific domain. Establishing the semantics of terms and relationships. Ensuring semantic consistency during reasoning. The W3C has proposed two standards for representing ontologies: RDFS and OWL. Each standard is defined as follows:

1.5.3.2 RDFS

RDFS (RDF Schema) [23] is a standard used for schema representation. It allows:

- Definition of an RDF vocabulary for a given domain.
- Presentation of relationships between different classes and their relations.
- Representation of lightweight ontologies with basic reasoning capabilities.

1.5.3.3 OWL

OWL (Web Ontology Language) [24] is a more expressive language for representing ontologies proposed by the W3C. It provides richer constraints on classes and relationships, such as cardinality, subclass, typing, transitivity, and inverse relationships.

OWL supports the formalization of complex ontologies with advanced reasoning capabilities. It is available in several versions, depending on the degree of expressiveness: OWL Lite, OWL DL, and OWL Full.

1.5.3.4 Specific Schemas

Various data schemas exist on the web today, designed to represent concepts related to specific domains. Some examples include:

- SKOS (Simple Knowledge Organization System): For representing thesauri, taxonomies, and classifications.
- Dublin Core: For metadata about documents.
- Creative Commons: For licensing rights.
- FOAF (Friend of a Friend): For describing people.

1.5.4 Trust Components

The components related to Proof, Trust, and Cryptography are not yet standardized as they are still under development. Their objectives include:

Defining a rule-based reasoning system. Utilizing cryptographic techniques. Developing a language that offers greater expressiveness for relationships.

1.6 Applications of the Semantic Web

The Semantic Web holds significant potential across a range of applications, transforming how data is accessed, managed, and utilized. Here, we explore two prominent examples of how Semantic Web technologies can be applied:

- E-commerce: The Semantic Web can enhance e-commerce by enabling agents to retrieve product data from multiple stores, thereby offering consumers the best deals available. By integrating and analyzing data across various sources, businesses can provide more accurate recommendations and optimize their offerings.
- Knowledge Management in Organizations: The Semantic Web facilitates the representation of organizational knowledge in the form of concepts and relationships. This allows decision-makers to access answers to complex, semantically-driven queries, improving decision-making and information retrieval within the organization.

In Europe, several initiatives have advanced the use of Linked Data, demonstrating its practical applications:

- DE – Bibliotheksverbund Bayern: This project involves Linked Data extracted from 180 libraries across Germany, enhancing access to bibliographic information.
- NL – Building, and Address Register: In the Netherlands, this initiative involves a Linked Data approach to managing the register of addresses and buildings, streamlining data access and integration.

1.7 Linked Data

Linked Data represents a method for publishing and interlinking structured data on the web, making it accessible and integrable across various domains [25]. This methodology leverages the Resource Description Framework (RDF) to encode data in a machine-readable format and utilizes Uniform Resource Identifiers (URIs) to identify and connect data to other resources. By adhering to these principles, Linked Data facilitates the integration of information from disparate sources and promotes the sharing and reuse of data across diverse applications. The concept of Linked Data is often illustrated through the "5-star" deployment scheme, ranging from making data available on the web (1 star) to providing it as fully linked data (5 stars). The core principles of Linked Data are [26]:

- Use URIs to Identify Things: Every piece of data is uniquely identified by a persistent URI, which should be resolvable and provide useful information about the identified entity.
- Use HTTP URIs: HTTP URIs ensure data is accessible on the web through the standard HTTP protocol, with the expectation that these URIs will resolve to information about the identified entity.
- Provide Useful Information Using RDF: RDF offers a framework for describing and linking data using subject-predicate-object statements, enhancing data integration and interoperability.
- Link Data to Other Data: Linked Data encourages creating a web of interconnected data by linking related datasets through URIs, fostering a more navigable and explorative data ecosystem.

These principles have gained traction among a diverse range of entities, including governmental bodies, academic institutions, and private enterprises. Adherence to these principles fosters a more open and connected web, benefiting both human users and machine applications and enabling novel insights and discoveries.

1.8 Application of Linked Data in Real World

Linked Data holds substantial potential across various domains and industries. For instance [10, 27–30]:

- Finance: Linked Data facilitates data integration for risk analysis, fraud detection, and investment decisions.

- Healthcare: It supports the integration of data from electronic health records, clinical trials, and research studies, advancing personalized medicine and drug development.
- Smart Cities: Linked Data helps integrate data from sensors, transportation systems, and public services, contributing to more efficient and sustainable urban environments.
- Education: Linked Data enhances curriculum development, learning analytics, competency-based education, open educational resources, and personalized learning pathways.
- Search Engines: The Google Knowledge Graph exemplifies the use of Linked Data to enhance search results by providing structured information about entities and their relationships, improving user experience and information retrieval.

In addition, Linked Data supports the creation of new applications and services, such as recommendation systems, chatbots, and virtual assistants. However, successful adoption requires a significant investment in data modeling, integration, and quality assurance, as well as the development of supportive tools and technologies.

1.9 Challenges and Issues of the Semantic Web

Despite its potential, the Semantic Web faces several challenges that researchers are actively working to address. These challenges include:

- Lack of Standardization: The absence of widely accepted standards for data representation and interoperability complicates the integration of diverse datasets.
- Scalability: Ensuring that Semantic Web applications can efficiently handle growing amounts of data and user queries presents a significant challenge.
- Interoperability and Integration: Achieving seamless interoperability between different systems and integrating heterogeneous data sources is essential for realizing the full potential of the Semantic Web.
- Data Uncertainty and Temporal Data: Handling uncertainty in data and managing temporal aspects pose significant difficulties for Semantic Web applications.
- Natural Language Processing and Formal Knowledge Representation: The complexity of natural language processing and the formal representation of knowledge remain major hurdles.

Table 1.2: Multi-criteria Comparison Between Web Versions

Criteria	**Current Web**	**Linked Data**	**Semantic Web**
Type of Web	Text Web	Web of Data	Knowledge Web
Type of Information	Web Pages	RDF Triplets	Ontologies
Search Tool	Search Engine	Query Engine	Query Engine
Standards	URL, HTTP, HTML	URI, HTTP, RDF, SPARQL, RDFS	URI, HTTP, RDF, SPARQL, OWL
Search Mode	Syntactic search by keywords	Search within RDF graphs	Logical search (using inference)

- Data Structuring in the Existing Web: Transforming and structuring the vast amounts of unstructured data on the current web into a format suitable for Semantic Web technologies is a complex task.

1.10 Synthesis

In this section, we present a comparative synthesis of the current web, the web of data, and the Semantic Web. Table 1.2 provides a multi-criteria comparison of these different versions of the web, inspired by the analysis conducted by Rousset [31]. It highlights the various standards, techniques, and methodologies employed in each version.

1.11 Analysis and Discussion

The Semantic Web aims to build upon the current web by adding a new layer that enhances its capabilities. This evolution involves making web resources accessible to machines, thereby facilitating the creation of new knowledge through the implementation of Linked Data.

This advancement is expected to significantly improve the quality of web applications and the relevance of search results based on semantic and knowledge-driven principles. To support this evolution, the W3C has proposed several standards to ensure interoperability among developed systems. However, challenges such as data uncertainty, scalability issues, and the need for seamless interoperability between diverse systems remain.

The handling of uncertain data is crucial, as reasoning over such data can lead to inaccurate and irrelevant results. Our work focuses on addressing these issues, particularly in the context of Linked Data, to enhance the accuracy and usefulness of the Semantic Web.

1.12 Conclusion

The Semantic Web represents a transformative leap forward from the current web, introducing a framework that not only connects data but also imbues it with meaning and context. By enhancing data interoperability, enabling sophisticated querying capabilities, and fostering advanced reasoning through ontologies, the Semantic Web paves the way for more intelligent and insightful data interactions.

Applications in e-commerce and organizational knowledge management illustrate the practical benefits of this approach, showcasing how it can revolutionize data access, improve decision-making, and optimize information retrieval. The European initiatives, such as Bibliotheksverbund Bayern and the Building and Address Register, further demonstrate the growing impact of Linked Data in real-world scenarios.

Despite its promising potential, the Semantic Web faces significant challenges. Issues related to data uncertainty, natural language processing, and the structuring of existing web data need to be addressed to fully realize its benefits. Additionally, challenges concerning scalability and interoperability must also be tackled to ensure seamless integration of diverse data sources. As research progresses, it will be crucial to develop solutions that handle these complexities effectively.

The evolution from a web of documents to a web of data and semantics represents a major advancement in how we interact with and derive value from information. While challenges remain, ongoing efforts to refine standards and address technical hurdles will drive the continued development of the Semantic Web. By enhancing our ability to reason about and connect data, this evolution promises to enrich our digital interactions and knowledge discovery processes significantly.

Chapter 2

Uncertainty in Artificial Intelligence and Semantic Web

2.1 Introduction

With the advent of computing, many researchers have grappled with the challenge of modeling information and knowledge in the presence of uncertainty, imprecision, contradiction, and more. In the realm of computer science, particularly in artificial intelligence, uncertainty is a somewhat nebulous term. It encompasses all forms of information deficiency, redundancy, inconsistency, imprecision, incompleteness, and contradiction. These issues make the exploitation of uncertain data difficult or even impossible, resulting in generated or inferred unreliable outcomes.

The anomalies above pose significant challenges when using so-called uncertain data in reasoning, decision-making, and the generation of new knowledge. To address these issues, several approaches have been proposed, notably the application of probability theory and possibility theory.

In an effort to delineate the concept of uncertainty, researchers have proposed various classifications depending on the application domain and context. In this chapter, we explore the most significant classifications that help elucidate this concept. Subsequently, we will discuss several approaches used to manage uncertainty in the field of artificial intelligence.

2.2 Classification of Uncertainty in Artificial Intelligence

In the study of uncertainty within artificial intelligence, a structured classification system is essential for comprehending and managing the myriad forms of uncertainty encountered.

Classifying uncertainty allows researchers and practitioners to identify its sources, understand its impacts, and develop appropriate strategies for handling it. Various frameworks have been proposed to categorize uncertainty based on different criteria, reflecting the diverse contexts and applications within the field. This section describes multiple prominent classification approaches, highlighting their methodologies and the specific types of uncertainty they address.

2.2.1 Classification by Dubois et al.

According to the ADRIA group (Argumentation, Decision, Reasoning, Uncertainty, Learning) [32], uncertainty can be classified into different types based on the causes that generate it:

- Uncertainty due to Lack of Information: Occurs when there is insufficient data to make a clear determination of the truth or falsity of a proposition.
- Uncertainty due to Contradictory Information: Arises when the available data is inconsistent or conflicting, making it difficult to establish a reliable conclusion.
- Uncertainty due to Phenomena Instability: Emerges when the unpredictable or unstable behavior of certain phenomena makes it challenging to forecast outcomes or decisions.

2.2.2 Classification by Denœux et al.

The authors of this book [33] emphasize the distinction between elements of objective information and subjective information on one hand, and between singular information and generic information on the other.

- Subjective information refers to any information that can be generated from sensors or direct observation.
- Objective information refers to any information expressed by an individual or generated without considering external observation.
- Singular information encompasses all sorts of particular facts or information obtained from observations.
- Generic information is obtained from a sample (e.g., a law) or a class of situations, etc.

In this book, the concept of information imperfection is classified according to three criteria: imprecision, contradiction, and uncertainty. Each of these criteria will be explained in the following sections.

2.2.2.1 Imprecision

Imprecise information corresponds to incompleteness and/or lack of information. In other words, information is considered imprecise if it cannot answer queries with precision.
Examples:

- Imprecision in the form of disjunction: Temperature $\in \{21°, 25°, 27°, 31°\}$, i.e., Temperature $= 21° \vee 22° \vee 23° \vee 24°$. In this example, the temperature value lies within the previously mentioned set. It is not known precisely.
- Temperature $\in [40°, 50°]$. This interval can answer questions like "What will the weather be like tomorrow?" However, the answer to a query asking for tomorrow's exact temperature will be imprecise (since the response will be in the form of an interval).

2.2.2.2 Contradiction

Contradiction refers to a relationship between two or more propositions where one affirms what the other denies.
Example: The following two propositions are contradictory (they cannot both be true simultaneously):

- "ADAM is young."
- "ADAM is 60 years old."

Another source of contradiction lies in particular cases (exceptions) in-laws (such as mathematical laws).

2.2.2.3 Uncertainty

Uncertainty is defined as the inability to determine whether a proposition is true or false. This arises due to a lack of information or the variability of phenomena (difficulty in observing phenomena or predicting their next state).
Example:

- The temperature tomorrow $= 23°$ with a probability of 0.6.

2.2.3 Classification by Li et Du

In their book [34], the authors propose a classification of uncertainty that addresses the uncertainty of knowledge by considering three important criteria: incompleteness of knowledge, incoordination of knowledge, and impermanence of knowledge. Each of these types is explained below.

2.2.3.1 Incompleteness of Knowledge

Incompleteness of knowledge refers to any knowledge that does not represent phenomena in a comprehensive manner. This inability to represent completely stems from the inability to observe the external world accurately, the use of equipment that provides imprecise results, and other similar issues. Consequently, parts of the information will not be acquired, leading to inherently uncertain results derived from such incomplete information.

2.2.3.2 In-coordination of Knowledge

The in-coordination of knowledge is defined as the internal contradiction within the knowledge itself. Several levels of in-coordination have been explained:

- Redundancy: Handling redundant knowledge poses problems such as time wastage, resource consumption, and performance degradation. Reasoning results can be negatively impacted by redundancy, potentially leading to erroneous conclusions.
- Interference: This occurs when certain knowledge disrupts other knowledge, thereby leading to reasoning errors. For example, interference is caused by noise during knowledge extraction.
- Conflict: This refers to all forms of contradiction between the components of knowledge.

2.2.3.3 Impermanence of Knowledge

Knowledge that is impermanent refers to knowledge that is not stable over time. This issue complicates the formalization of knowledge, as the knowledge may change or become outdated, requiring continuous updates and adjustments to maintain accuracy.

2.2.4 Classification by Mcheick & al.

The authors of this article [35] approached the classification of uncertainty differently. They proposed three types of uncertainty:

- Epistemic Uncertainty: Characterized by a lack of information, incompleteness of information, etc. This type of uncertainty arises from a lack of knowledge or incomplete information about a system or phenomenon. It is often due to limitations in data or gaps in our understanding of the underlying processes.
- Aleatory Uncertainty: Describes the non-systematic nature of data (variability, irreducibility) and the natural variability of a system.
- Error: A gap not due to a lack of knowledge.

2.2.5 Classification According by Thunnissen

In this article [36], several classifications of uncertainty were proposed (depending on the domain of activity). We focus on the classification of uncertainty for computer modeling and simulation. This classification distinguishes three classes:

- Variability: This refers to the natural, inherent variation within a system or process that cannot be controlled or eliminated. It represents fluctuations or changes in the system's behavior over time due to external factors, random events, or the inherent unpredictability of the system.
- Uncertainty: This class of uncertainty is driven by incomplete or imprecise knowledge about a system or process. It can arise from several sources:
  - Imprecision: When data or measurements are not precise, leading to ambiguous results. For example, measurements taken with low-resolution instruments may lack exact values.
  - Non-specificity: When the available information is too broad or general, making it difficult to pinpoint specific details about the system.
  - Conflicts: Arise when different sources of data or models provide contradictory information, making it hard to determine which is correct.
  - Contradictions: Occur when existing knowledge or assumptions about a system conflict with newly gathered data, leading to inconsistencies in understanding.
- Error: This has two subclasses:

  - Recognized Error: Approximations during the modeling of a process (simplification).
  - Unrecognized Error: Any other unidentified type of error.

2.2.6 Classification by Xiao

Another classification inspired by Aguilar-Martin [37], Bonissone & Tong [38], and Smets [39] was proposed in this thesis [40]. The author considered the notion of imperfection to encompass both uncertainty and imprecision. He defines uncertain information as any information whose validity we doubt. In contrast, imprecision is defined as any information that does not describe reality completely.

The proposed subcategories for each type of imperfection are mentioned below:

- Uncertainty: This category represents situations where the validity of information is in question. Each type of uncertainty is illustrated with an example:
  - Probability: What are the chances of Trump becoming the president of the United States again?
  - Credibility: I assume that all precautions have been taken.
  - Possibility: The possibility of passing the semester.
- Imprecision: This refers to information that lacks exactness or fails to describe reality fully. The sub-types are explained with examples:
  - Set: He visited one of the French cities.
  - Interval: This car costs between 100 and 200 million cents.
  - Incompleteness: The RAM size of this PC is 16 GB (the RAM size alone does not suffice to describe all the characteristics of a PC).
  - Fuzziness: She is old (the age is not specified).

2.3 Approaches for Handling Uncertainty in Artificial Intelligence

Several mathematical approaches have been proposed in the literature for managing and processing uncertainty in artificial intelligence. Three prominent approaches are the probabilistic approach, possibility theory, and fuzzy logic, which are briefly explained below [41].

2.3.1 Probabilistic Approach

Probability theory is a branch of mathematics primarily used to measure the probabilities of random events. It is the most suitable approach for handling and presenting uncertain information, relying on the concepts of frequentism and subjectivism [33].

2.3.2 Possibility Theory

Another frequently used approach for managing uncertainty is possibility theory. This theory is mainly based on fuzzy sets [42]. The fundamental principle of this theory is to assign a degree of possibility and a degree of certainty to each proposition [33].

There are two variants of possibility theory: qualitative possibility theory and quantitative possibility theory [32].

2.3.3 Fuzzy Logic

Fuzzy logic, introduced by Zadeh [42], is another essential approach for handling uncertainty, particularly when dealing with imprecise or vague information. Unlike classical logic, which relies on binary true/false values, fuzzy logic allows for varying degrees of truth. It operates with fuzzy sets, where an element's membership is expressed by a degree between 0 and 1, reflecting the uncertainty or ambiguity in the information. This approach is particularly useful when precise information is unavailable or when linguistic variables (e.g., "tall," "old") are involved.

2.4 Uncertainty According to the URW3-XG Group (W3C Uncertainty Reasoning for the World Wide Web Incubator Group)

In this section, we will briefly present the types of uncertainty proposed by the URW3-XG group [3], one of the W3C-affiliated groups, as well as the various formalisms used by this group to represent uncertainty.

2.4.1 Types of Uncertainty

Researchers from the URW3-XG group have identified six types of uncertainty [3], which are listed below:

- Ambiguity: This occurs when the terms used in a statement are not clearly defined, making it difficult to determine if the statement accurately describes the situation.
- Empirical: A statement about a situation or event is either true or false, but it's unclear in which specific cases it holds true. This can be clarified through additional information or observation, such as conducting an experiment.
- Randomness: This refers to a situation where a statement is part of a broader category governed by statistical rules, determining the likelihood of its truth.
- Vagueness: The terms in a statement lack precise boundaries, leading to an unclear relationship between the terms and the actual situation they describe.
- Inconsistency: This arises when no possible scenario can make the statement true.
- Incompleteness: This refers to a lack of sufficient information about a situation, leaving some details unknown or missing.

2.4.2 Formalisms for Managing Uncertainty

This group has proposed several formalisms for representing and managing uncertainty. Among these are the following theories: Probability Theory, Fuzzy Set Theory, Belief Functions, Random Sets, Rough Sets and Hybrid Approach (combination of several approaches)

The W3C Incubator Group has proposed an ontology to categorize different types of uncertainty, distinguishing between epistemic uncertainty (arising from limited knowledge) and aleatory uncertainty (resulting from inherent randomness). Figure 2.1 illustrates this uncertainty ontology. The uncertainty types ontology is presented in Figure 2.2.

2.5 Uncertainty in Linked Data

Uncertainty in Linked Data encompasses various forms of knowledge imperfection, such as incompleteness, imprecision, and ambiguity [1]. The primary sources of uncertainty include:

- Data exchanges between automated agents lack a standardized format for representing uncertainty.

[1] https://www.w3.org/2005/Incubator/urw3/group/draftReport.html

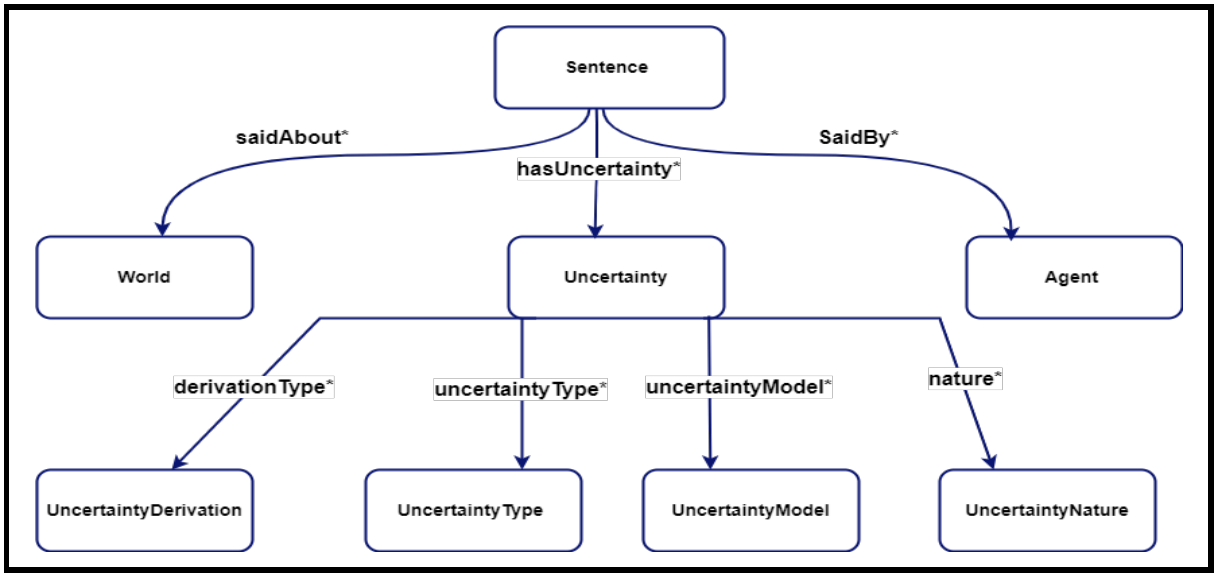


Figure 2.1: Uncertainty Ontology [3]

- Dataset construction from unstructured or semi-structured data, resulting in incomplete or heterogeneous formats.
- The use of different URIs to represent the same resources.
- Increased heterogeneity due to datasets created by various organizations.
- The dynamic nature of the web, which can alter the state of data.

As part of our contributions, we address several types and sources of uncertainty, which are detailed in the following subsections.

2.5.1 Incompleteness in Linked Data

Incompleteness is a significant aspect of uncertainty in Linked Data, referring to situations where information about the world is missing or incomplete [2]. This can include missing data, absent links, or incomplete triples. Incompleteness often arises from the limitations inherent in the source data and can be exacerbated during data integration from heterogeneous datasets. Several strategies have been explored in the literature to address missing data, focusing on discovering missing links and predicting missing entity types. The following section will review key works addressing these challenges.

[2]https://www.w3.org/2005/Incubator/urw3/group/draftReport.html

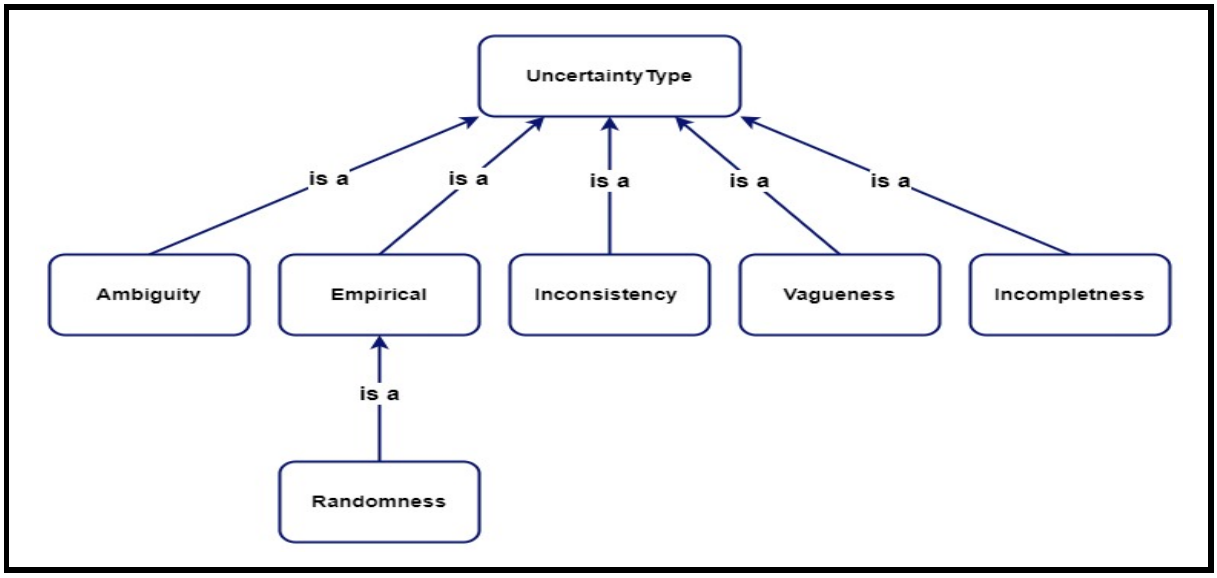


Figure 2.2: Uncertainty Types Ontology [3]

2.5.2 SameAs Links in Linked Data

SameAs links are used in Linked Data to connect resources that represent the same entity or concept [43]. These links are crucial for integrating data from various sources that may use different identifiers, such as names, unique IDs, or URLs. SameAs links help unify data across diverse systems, facilitating easier querying and analysis of information from multiple sources. They are often used alongside other link types, such as owl:sameAs (indicating semantic equivalence) and skos:exactMatch (indicating identical concepts).

2.5.3 Errors in Linked Data

Despite its advantages, Linked Data is susceptible to various errors that can compromise data accuracy and reliability [44–46]. Errors in Linked Data can be categorized as follows:

- Syntactic Errors: Issues arising from incorrect data extraction, such as invalid syntax or datatypes.
- Factual Errors: Incorrect factual statements, such as erroneous birthdates.
- Conceptual Errors: Violations of conceptual relationships, such as assigning an invalid type to an entity.
- Schema Errors: Errors in domain/range declarations or inconsistent ontology modeling, leading to semantic mismatches.

- Missing Data: Absence of factual triples due to incomplete information extraction.
- Outdated Data: Data attributes or relationships that are no longer accurate due to the dynamic nature of knowledge.

Errors in Linked Data can undermine its usability and trustworthiness. They can lead to inaccurate results, inconsistent data integration, reduced user confidence, and increased manual curation efforts. Detecting and mitigating these errors is essential to enhancing data quality, ensuring reliability, and managing knowledge graphs effectively.

2.6 Synthesis of Studies on Uncertainty

Initially, we examined various classifications of the term "uncertainty" by reviewing existing literature. The following table illustrates the different classifications obtained, specifying the types relative to each classification, as well as the formalisms used for handling uncertainty for each type.

Through this synthesis, it is evident that several formalisms have been proposed to address one or more types of uncertainty. Additionally, each type of uncertainty can be represented by multiple formalisms, as shown in Table 1.1.

2.7 Analysis and Discussion

The term "uncertainty" is quite broad and can signify different meanings depending on the domain and context of the work. Indeed, various authors in the literature have provided a diverse range of definitions for this term, as well as different classifications.

In our case, we focused on defining uncertainty in the field of artificial intelligence. To this end, we consulted several definitions, classifications, and approaches for representing uncertainty proposed by various researchers, as well as those proposed by the URW3-XG group. The classifications studied vary according to the sub-domain and the specific problem addressed by the author.

Despite the broad differences in classification among authors, we noticed that the majority based their classifications on similar criteria. After studying all these variations, we can summarize the different points used in classification into the following four sub-types:

Table 2.1: Summary of Different Uncertainty Classifications

Author	**Classification**	**Formalism**
Dubois et al. [32]	Uncertainty due to Lack of Information	Theory of Possibility
	Uncertainty due to Contradictory Information	N/A
	Uncertainty due to Phenomena Instability	N/A
Denœux et al. [33]	Imprecision	Belief Functions Theory, Imprecise Probability Theory
	Uncertainty	Probability Theory, Theory of Possibility
	Contradiction	N/A
Li et Du [34]	Incompleteness of Knowledge	Probability Theory, Fuzzy Set, Rough Sets
	In-coordination of Knowledge	
	Impermanence of Knowledge	
Mcheick et al. [35]	Epistemic Uncertainty	Dempster-Shafer Theory, Evidence Theory, Theory of Possibility
	Aleatory Uncertainty	Probability Theory
	Error	N/A
Thunnissen [36]	Variability	N/A
	Uncertainty	N/A
	Error	N/A
Xiao [40]	Uncertainty	Probability Theory, Theory of Possibility
	Imprecision	Probability Theory, Theory of Possibility
URW3-XG Group [3]	Ambiguity	Probability Theory, Fuzzy Set, Theory Belief Functions, Random Sets, Rough Sets, Hybrid Approach
	Empirical	
	Randomness	
	Vagueness	
	Inconsistency	
	Incompleteness	

- Uncertainty: When the information is not certain. [32, 33, 35, 40]
- Imprecision: In cases of lack of clarity. [33, 34, 40]

- Lack of Information: When part or all of the information is missing. [32, 34]
- Error: Any mistake made during the representation (writing) of the information. [35, 36]

This analysis shows that while the conceptualization of uncertainty may vary, there is a common underlying framework that can be used to understand and address it across different contexts and domains. However, it is important to acknowledge the limitations of probability theory and possibility theory, especially concerning scalability. As the complexity and volume of data increase, traditional probabilistic models can become computationally expensive and less efficient, while possibility theory may struggle with handling large-scale uncertainties and integrating diverse data sources effectively. Additionally, both approaches can be limited in their ability to accommodate real-world phenomena characterized by high degrees of vagueness or ambiguity, which often require more nuanced frameworks for accurate representation and reasoning.

2.8 Conclusion

In this chapter, we explored the concept of uncertainty, examining various definitions, classifications, and mathematical approaches for handling it in the context of artificial intelligence. We reviewed several frameworks and theories, including probabilistic approaches and the theory of possibilities. We also discussed the classifications provided by notable researchers and organizations, such as the URW3-XG group, which helped us understand the multifaceted nature of uncertainty.

Our analysis underscored the complexity and breadth of the term "uncertainty," showing that it encompasses different aspects such as lack of information, imprecision, and errors. Despite the differences in classification methodologies, a common thread was identified: the need for robust and flexible formalisms to effectively represent and manage uncertainty in various domains.

We established a comprehensive understanding of uncertainty in artificial intelligence, providing a foundation for future research and practical applications. This foundational knowledge is crucial for developing advanced systems capable of reasoning and decision-making in uncertain environments, ultimately enhancing the completeness and efficiency of AI-driven solutions. In the context of our work, we employ advanced deep learning techniques to address the limitations of existing methods, particularly regarding scalability and the extraction of semantic dependencies between data.

Chapter 3

Basics of Deep Learning for Resolving Uncertainty in Linked Data

3.1 Introduction

In recent years, Artificial Intelligence (AI) has made significant strides across various domains, including image processing, robotics, automated reasoning, decision-making systems, and bioinformatics. Among these applications, the management of uncertainty in Linked Data has gained increasing attention, particularly with the advent of Deep Learning.

Deep Learning, a specialized branch of AI derived from Machine Learning, leverages artificial neural networks to model complex patterns within large datasets. Its capacity to handle and mitigate uncertainty in data has been instrumental in enhancing approaches to automatic data integration and analysis in Linked Data systems. In this chapter, we will present the Deep Learning concepts used in our context and within our contributions for resolving uncertainty in Linked Data. This chapter introduces the fundamental concepts of Deep Learning, with a focus on its application to managing uncertainty in Linked Data. After a brief overview of AI and Machine Learning, we talk about neural networks and then explore Deep Learning in detail. Special attention is given to Recurrent Neural Networks (RNNs) and their variants, which play a crucial role in modeling sequential data and managing uncertainties in Linked Data environments.

3.2 Artificial Intelligence

Artificial Intelligence (AI) stands as one of the most innovative fields within science and engineering. The term "Artificial Intelligence" was first coined by McCarthy in 1956

during a conference at Dartmouth College, and it has since become synonymous with the study of intelligent systems [47].

AI involves the simulation of human intelligence through machines, particularly computer systems. These systems exhibit behaviors traditionally associated with human cognition, including planning, reasoning, problem-solving, knowledge representation, learning, and perception [47].

In 1993, Heer defined AI as "the art of creating machines capable of performing functions that require intelligence when done by humans" [48]. The ultimate goal of AI is to develop systems capable of executing complex tasks with a level of competence that matches or surpasses human abilities, especially in areas where uncertainty and complexity are prevalent, such as Linked Data [49].

AI is intricately linked with concepts such as Knowledge-Based Systems, Expert Systems, Intelligent Systems, Knowledge Acquisition, and Machine Learning. Machine Learning, in particular, plays a pivotal role in AI by providing the methods necessary for systems to learn and adapt, a critical aspect when dealing with the inherent uncertainty in Linked Data.

3.3 Machine Learning

Machine Learning (ML) is a critical subfield of AI that focuses on the automatic extraction and utilization of knowledge from large datasets. This process is essential in contexts where data may be incomplete or uncertain, such as in Linked Data environments. ML encompasses various objectives, including data mining, classification, variable selection, discrimination, regression, model selection, rule generation, and inference, all of which are crucial in managing and reducing uncertainty in data.

According to H. Simon, "Learning in a system is indicated by the changes it undergoes. These changes are adaptive in the sense that they enable the system to perform the same task, or tasks from the same population, more effectively and efficiently the next time it is performed" [50]. In essence, "Machine Learning tools acquire, expand, and refine the knowledge available to the system, enabling it to adapt to new challenges, such as those posed by uncertain or incomplete data in Linked Data environments" [49].

Today, Machine Learning is recognized as a powerful tool for enhancing knowledge acquisition processes, particularly in the context of Linked Data. The rise of Deep Learning, a variant of neural networks within ML, has further advanced the ability of AI systems to manage and interpret uncertain data. The following sections will delve deeper into these concepts, beginning with a discussion of neural networks, which form the foundation of

Deep Learning. Figure 3.1 illustrates the relationship between AI, Machine Learning, Neural Networks, and Deep Learning.

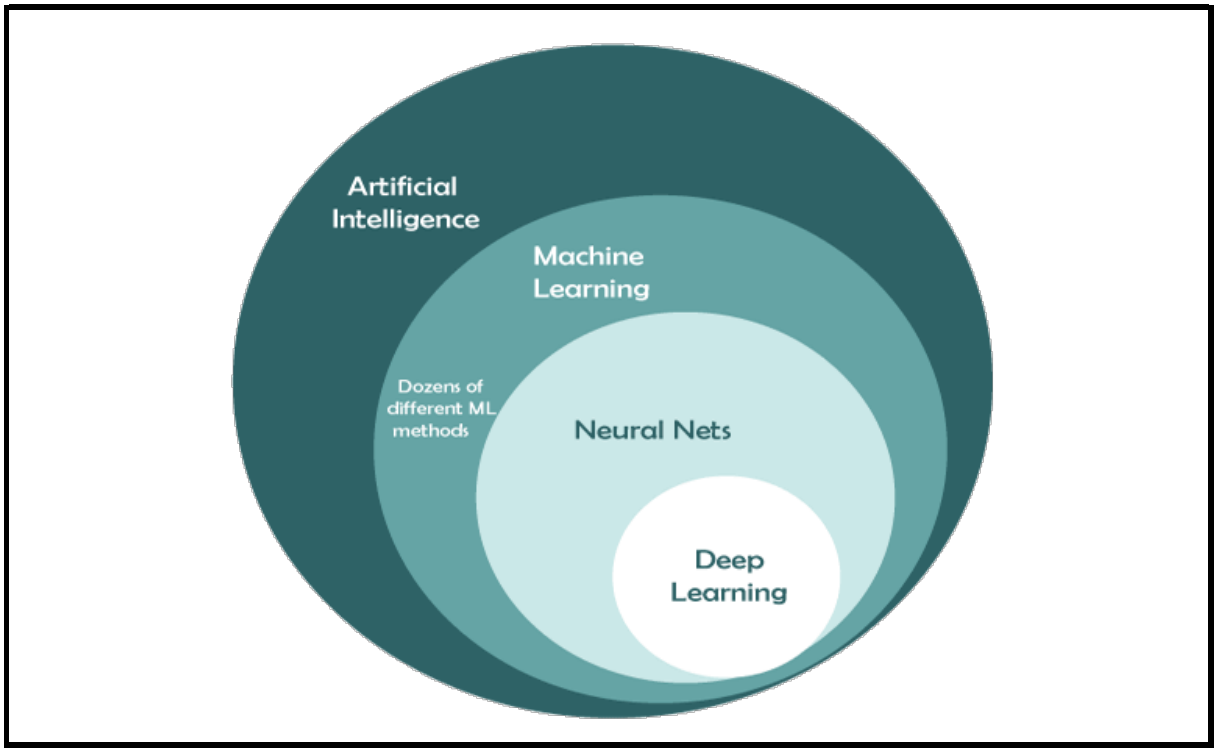


Figure 3.1: Artificial Intelligence, Machine Learning, Neural Networks, and Deep Learning

3.4 Neural Networks

Neural networks, inspired by the biological neurons in the human brain, consist of interconnected layers of neurons designed to solve complex problems, such as uncertainty management in Linked Data. These networks have shown significant success in tasks such as image recognition, natural language processing, and, more recently, in managing and interpreting uncertainty in Linked Data systems [51].

3.4.1 Operation of Neural Networks

A neural network operates similarly to neural networks in the human brain, processing information through a series of interconnected layers. The first layer is the input layer, which receives the data, while the last layer is the output layer, which produces the result. This layered structure is particularly effective in managing complex, uncertain datasets.

Each layer consists of neurons, where the outputs of these neurons serve as inputs for the subsequent layer, enabling the network to learn and adapt from data. The architecture of a neural network is illustrated in Figure 3.2.

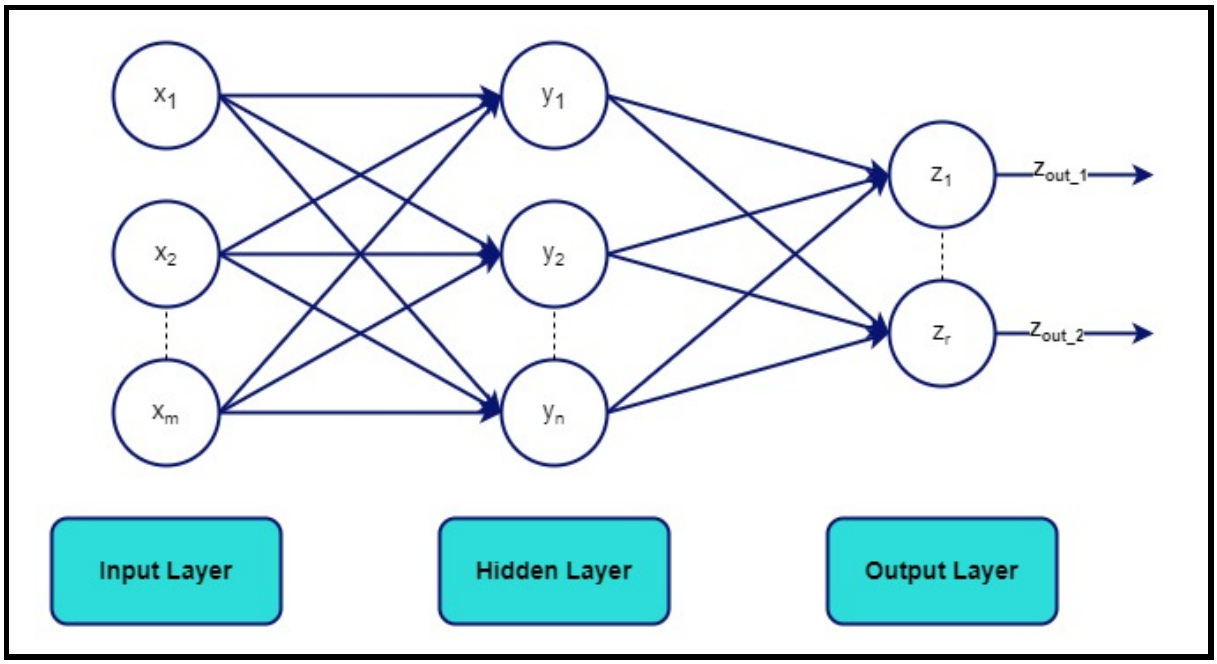


Figure 3.2: Architecture of a Neural Network [52]

3.4.2 Activation Functions

Activation functions are critical in neural networks as they adjust the output value of a neuron Y. This value, which can range from $-\infty$ to $+\infty$, is computed using the following equation:

$$Y = \sum(\text{weight} \times \text{input}) + \text{bias}$$

The bias is an additional parameter that helps the network better fit the data, particularly in scenarios where data is uncertain or incomplete. Activation functions limit the value of the output Y and determine whether the neuron should be activated. Several types of activation functions are discussed in Appendix A.

3.5 Deep Learning

Deep Learning represents a significant advancement in Machine Learning, offering a framework for learning hierarchical representations of data through multiple layers of neural

networks. Unlike traditional Machine Learning algorithms, which often require manual feature extraction, Deep Learning automatically learns features directly from raw data. This capability has led to breakthroughs in areas such as image and speech recognition, natural language processing, and autonomous systems.

The core idea behind Deep Learning is that each layer in the network learns to represent data at increasing levels of abstraction. Lower layers might capture simple features, such as edges in an image, while higher layers could represent more complex structures, such as shapes or objects.

Two major types of Deep Learning architectures are:

- Convolutional Neural Networks (CNNs): Primarily used for processing grid-like data such as images. CNNs are designed to automatically and adaptively learn spatial hierarchies of features, making them highly effective for tasks like image recognition and classification.

- Recurrent Neural Networks (RNNs): Specialized for sequence data, RNNs are employed in applications involving time series or sequential inputs, such as uncertainty detection in linked data, speech recognition, and translation. The following section details these sequence models, emphasizing their importance in managing tasks involving sequential data.

3.5.1 Sequence Models

Sequence models are crucial in Deep Learning for tasks that involve sequential or time-dependent data. These models are capable of capturing temporal dependencies and are widely used across various Linked Data applications:

- Uncertainty in Linked Data: Detecting and managing inconsistencies and incomplete information in Linked Data, which is crucial for ensuring data reliability and accuracy in semantic web applications.

- Error Detection in RDF Triple Stores: Identifying incorrect or conflicting RDF triples within a dataset helps maintain the integrity and consistency of knowledge graphs used in various domains such as knowledge management and semantic search.

- Validation of SameAs Links: Assessing the correctness of sameAs links between entities in different datasets, which ensures that these links accurately represent entity equivalence and improves data integration and interoperability.

- Conflict Resolution in Ontologies: Detecting and resolving conflicts between ontologies and data, which is essential for maintaining coherent and consistent semantic frameworks in complex systems.

Sequence models, particularly RNNs and their advanced variants like Long Short-Term Memory (LSTM) networks and Encoder-Decoder architectures, are powerful tools in Deep Learning. They enable the modeling of sequential dependencies and have become indispensable in tasks where the order of data significantly impacts the output.

3.5.2 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) [53] are a class of neural networks designed to process sequential data by maintaining a form of memory across time steps. This capability is crucial for tasks involving sequences where previous ones influence current inputs. The standard architecture of an RNN is depicted in Figure 3.3.

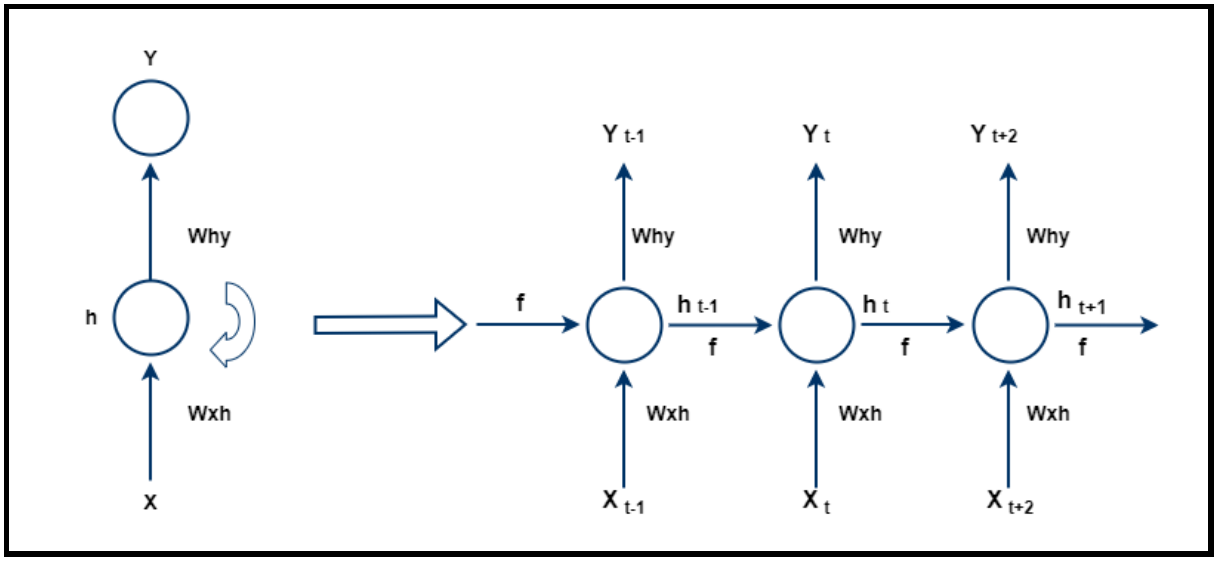


Figure 3.3: Architecture of a Standard RNN [54]

In an RNN, each time step's output $h(t)$ depends on the previous output $h(t-1)$ and the current input $x(t)$. The network's structure allows it to capture temporal dependencies within the data. Various RNN architectures cater to different types of problems, as illustrated in Figure 3.4:

- One-to-One: The most straightforward RNN model, where a single input produces a single output. For instance, this can be applied to image classification tasks.

- One-to-Many: Generates a sequence from a single input, such as creating a sentence from an initial word.
- Many-to-One: Produces a single output from a sequence of inputs, commonly used in tasks like sentiment analysis.
- Many-to-Many: Can be represented in two forms:
  - Sequence-to-Sequence: Outputs a sequence after processing the entire input sequence, useful in tasks like detecting types in Linked Data.
  - Sequence-to-Sequence (Online): Generates outputs as inputs are received, such as labeling each video frame in real time.

Despite their capabilities, RNNs have several limitations, particularly when applied to Linked Data:

- Short-Term Memory: RNNs primarily focus on recent data, which can be insufficient for capturing long-range dependencies in sequential Linked Data.
- High Processing Time: Due to their sequential nature, RNNs can be computationally intensive, which is challenging when dealing with large-scale Linked Data.
- Ignoring Future Sequences: Standard RNNs do not utilize future data beyond the current time step, which can be a drawback for tasks requiring a comprehensive understanding of the entire sequence.

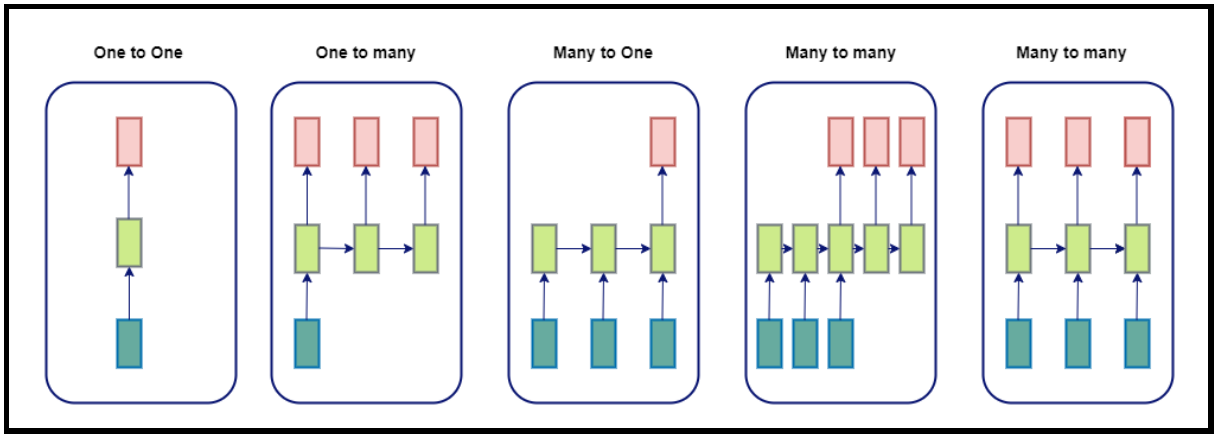


Figure 3.4: Different RNN Architectures [55]

3.5.3 Bidirectional Recurrent Neural Networks (BRNNs)

A notable limitation of traditional RNNs is their inability to incorporate future information, which is critical for comprehensive analysis in many applications. Bidirectional Recurrent Neural Networks (BRNNs) address this issue by processing data in both forward and backward directions. The BRNN architecture is depicted in Figure 3.5.

The principle of BRNNs involves two passes through the data: one forward and one backward. This dual processing allows BRNNs to utilize both past and future context when making predictions at time t. This approach provides significant advantages in understanding sequences where future context is as important as past context. However, BRNNs can struggle with very long sequences due to the vanishing gradient problem, where gradients diminish rapidly during training, leading to difficulties in learning long-term dependencies.

Another issue prevalent in RNN training, especially with many layers, is the vanishing and exploding gradient problems. These problems arise when gradients either become too small (vanishing) or too large (exploding) during backpropagation, which adversely affects the network's training stability and performance.

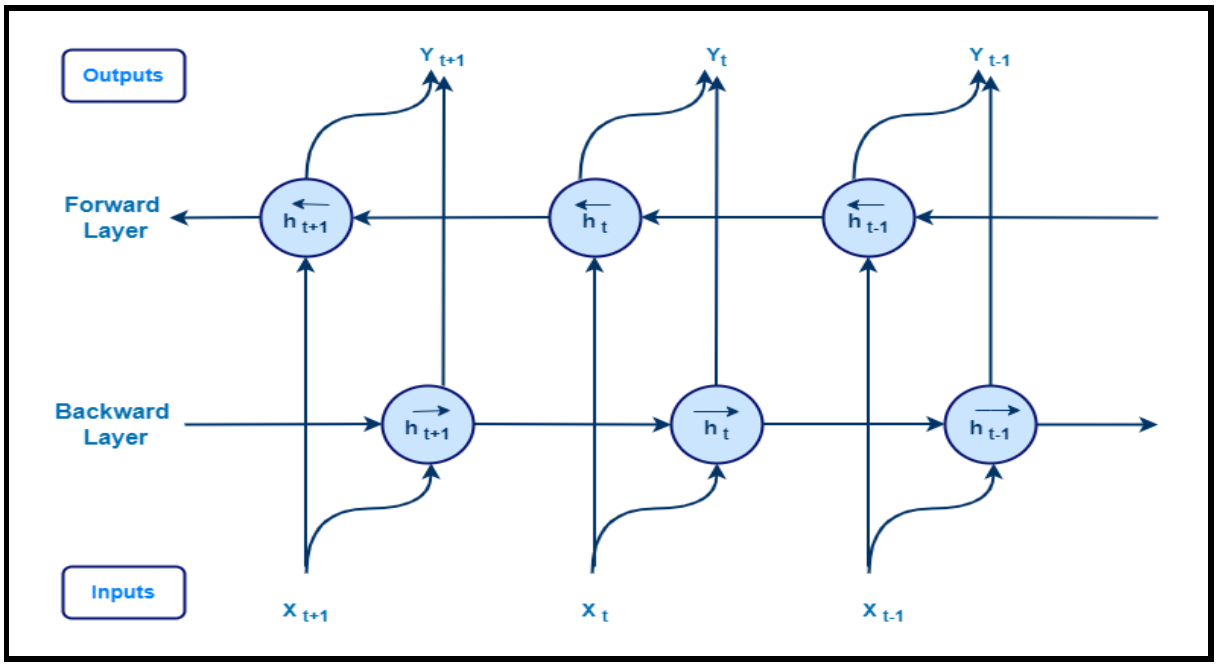


Figure 3.5: Architecture of a Bidirectional RNN [56]

3.5.4 Long Short-Term Memory (LSTM) Networks

Long Short-Term Memory (LSTM) networks [57] represent an advanced variant of Recurrent Neural Networks (RNNs) designed to effectively manage long-range dependencies in sequential data. Introduced by Hochreiter and Schmidhuber in 1997 [58], LSTMs are widely employed for various tasks involving temporal sequences and are particularly beneficial in contexts where the data exhibits significant dependencies (Figure 3.6).

Unlike traditional RNNs, LSTMs address the vanishing gradient problem by incorporating a sophisticated gating mechanism and a memory cell, as shown in Figure 3.6. This architecture enables the network to retain information over extended sequences, making it particularly useful for handling the uncertainty and complexity inherent in Linked Data.

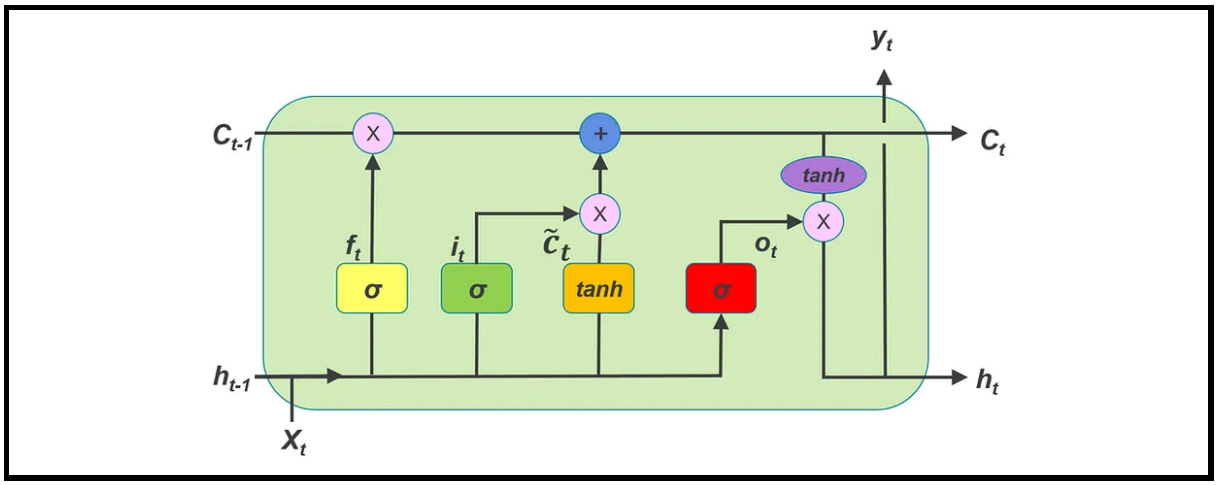


Figure 3.6: Detailed Architecture of an LSTM Cell [59]

An LSTM network processes inputs including $h(t-1)$, the output from the previous LSTM unit, $X(t)$, the input at the current time step, and $C(t-1)$, the memory from the previous unit. The LSTM's gates manage the flow of information through the network, allowing the model to selectively remember or forget information, which is crucial for managing Linked Data.

LSTMs employ three primary gates:

3.5.4.1 Forget Gate

The forget gate regulates the information to be discarded from the previous cell state $C(t-1)$. This is determined using the following function:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

where $[h_{t-1}, x_t]$ denotes the concatenation of the previous output h_{t-1} and the current input x_t, W_f represents the weight matrix, and b_f is the bias term. The output f_t is a vector of values between 0 and 1, representing the relevance of each element in the previous cell state $C(t-1)$.

The updated cell state C'_t is computed as:

$$C'_t = f_t \cdot C_{t-1}$$

Values close to 0 are suppressed, while values near 1 are retained.

3.5.4.2 Input Gate

The input gate governs the incorporation of new information based on the current input x_t. It involves two key functions:

1. Candidate Information:

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

   This function generates candidate values for the cell state. The output vector ranges between -1 and 1.

2. Input Gate Activation:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

   This function determines which elements of $\tilde{C}_t$ will be added to the current cell state C_t. The final cell state is updated as:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

3.5.4.3 Output Gate

The output gate determines the output of the cell at time t. It first applies a sigmoid function to select which values should be output:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

Subsequently, the output h_t is computed by combining the cell state C_t with the output gate activation:

$$h_t = o_t \cdot \tanh(C_t)$$

3.5.5 Gated Recurrent Unit (GRU)

Gated Recurrent Units (GRUs) are designed to address the vanishing gradient problem, similar to LSTMs but with a simplified architecture. Unlike LSTMs, which utilize three gates and a cell state, GRUs employ only two gates and a single hidden state mechanism. Figure 3.7 illustrates the GRU cell architecture.

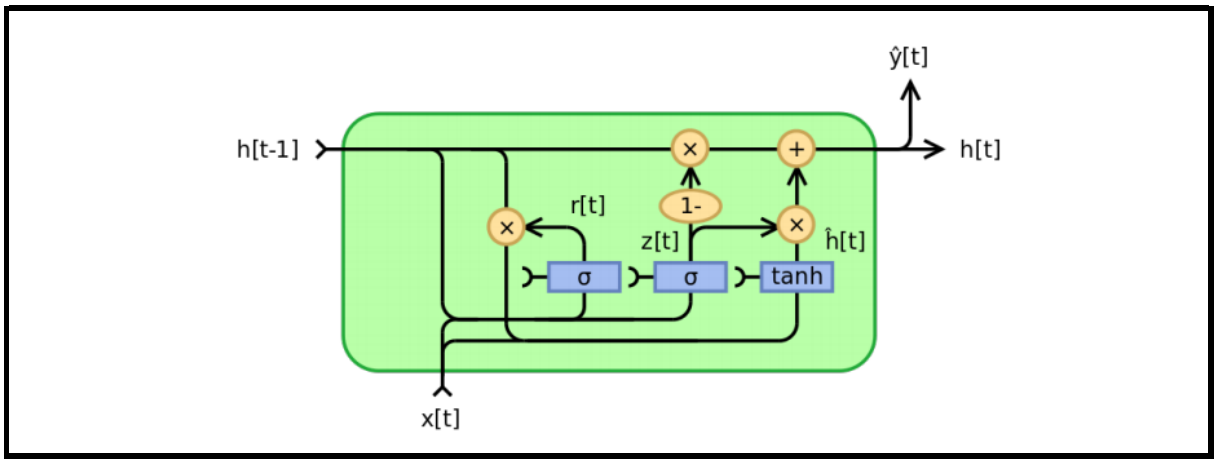


Figure 3.7: Architecture of a GRU Cell [60]

3.5.5.1 Update Gate

The update gate controls how much of the past information is retained in the hidden state. It is computed using:

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

where W_z and U_z are the weight matrices for the input x_t and the previous hidden state h_{t-1}, respectively.

3.5.5.2 Reset Gate

The reset gate determines how much of the past information should be forgotten. It is given by:

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

where W_r and U_r are the weights associated with x_t and h_{t-1}, respectively.

3.5.5.3 Current Memory Content

GRUs use a new memory content $\hat{h}_t$, which is computed with the help of the reset gate. It is calculated as:

$$\hat{h}_t = \tanh(Wx_t + r_t \circ Uh_{t-1})$$

Here, $r_t \circ Uh_{t-1}$ represents the element-by-element multiplication between the reset gate and the previous hidden state.

3.5.5.4 Final Memory at the Current Time Step

The final hidden state h_t is a combination of the previous hidden state h_{t-1} and the current memory content $\hat{h}_t$, weighted by the update gate:

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \hat{h}_t$$

3.5.6 Bidirectional LSTM Networks (Bi-LSTM)

Bidirectional LSTMs (Bi-LSTMs) extend the LSTM architecture by processing data in both forward and backward directions. The output $\hat{y}_t$ of a Bi-LSTM is computed as:

$$\hat{y}_t = g(W_b[\vec{a}_t, \hat{a}_t] + b_b)$$

Figure 3.8 depicts the general structure of a Bi-LSTM network.

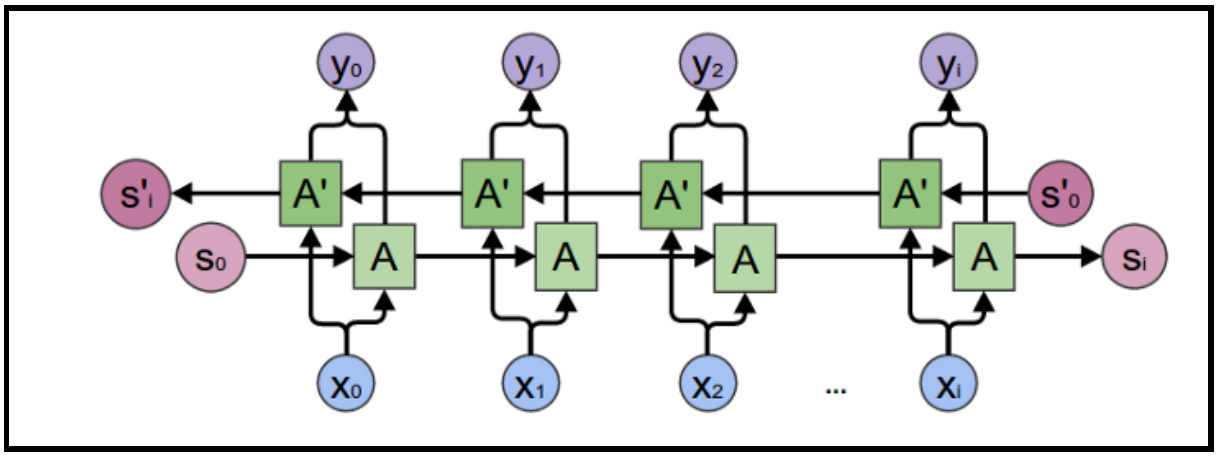


Figure 3.8: General Architecture of a Bidirectional LSTM Network (Bi-LSTM) [61]

While sequence models like LSTMs and GRUs handle fixed-size input and output sequences (one-to-one, one-to-many, many-to-one), they do not address sequence-to-sequence generation (many-to-many). The encoder-decoder model tackles this challenge.

3.5.7 Encoder-Decoder

The encoder-decoder model, as shown in Figure 3.9, is designed to transform an input sequence into a sequence output. This architecture comprises an encoder and a decoder, both of which typically use RNNs to process variable-length sequences.

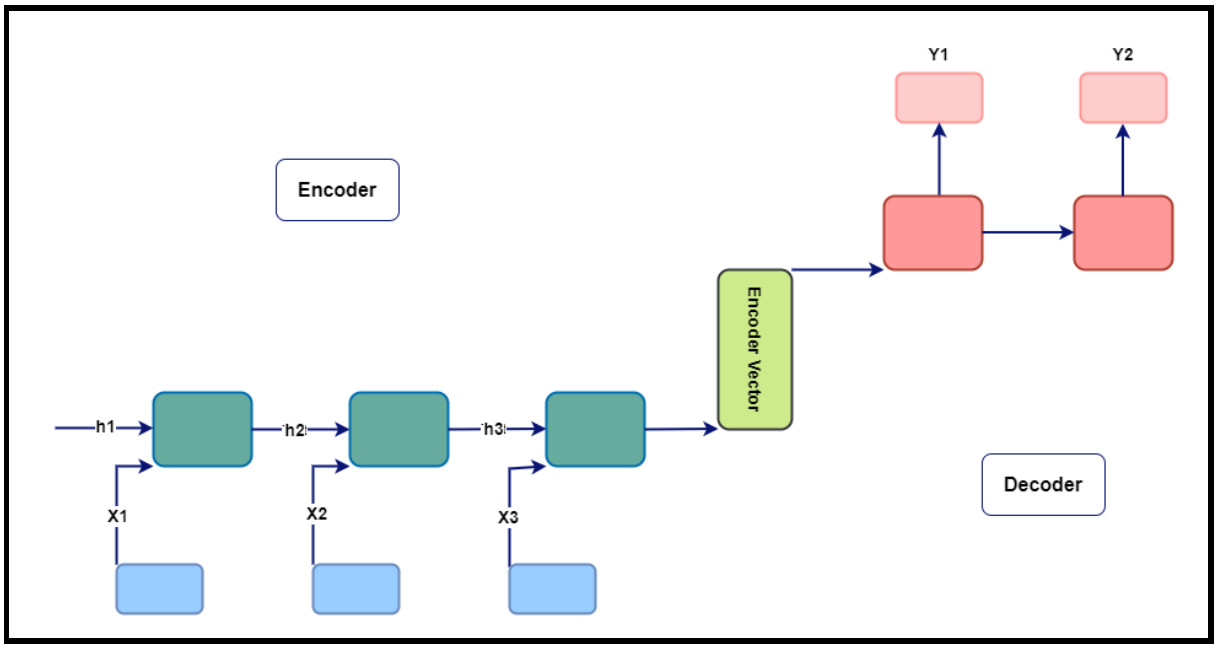


Figure 3.9: Architecture of the Encoder-Decoder Model [62]

Key benefits of the encoder-decoder approach include:

- Training a unified model on both source and target sequences.
- Handling variable-length input and output sequences effectively.

In this model, each encoder unit processes an input x_i and accumulates information, which is passed to the next unit through forward propagation. The final hidden state of the encoder, or encoder vector, is then transferred to the decoder. The decoder generates outputs y_t based on this encoder vector, where each output at time $t > 1$ depends on the previous hidden state and output.

3.6 Attention Mechanism

The attention mechanism significantly enhances neural networks by allowing them to focus on the most relevant parts of the input sequence. This approach greatly improves the performance of sequence-to-sequence models, such as the encoder-decoder architecture [63–65]. One of the primary advantages of the attention mechanism is its ability to overcome limitations associated with processing long sequences.

In sequence-to-sequence models, especially those involving long sequences, the traditional approach of using only the final hidden state of the encoder to inform the decoder can be inadequate. This limitation arises because the final hidden state might not capture all the necessary information from the entire sequence. The attention mechanism addresses this issue by incorporating all the hidden states of the encoder into the context vector for each time step of the decoder.

3.6.1 Calculation of Alignment Scores

The attention mechanism operates by calculating alignment scores between the decoder's current hidden state and all the hidden states generated by the encoder. These alignment scores are computed to determine how much focus should be given to each part of the input sequence when producing the output at the current time step.

Mathematically, the alignment score between the decoder's hidden state h_t and each encoder's hidden state h_i is often computed using a similarity function, such as the dot product or a learned function:

$$e_{ti} = \text{score}(h_t, h_i)$$

where e_{ti} represents the alignment score between the decoder's current hidden state h_t and the i-th encoder hidden state h_i.

3.6.2 Generation of the Context Vector

The alignment scores are then normalized using the softmax function to ensure that they sum up to one and represent a probability distribution. This normalization process converts the alignment scores into attention weights α_{ti}:

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_j \exp(e_{tj})}$$

The context vector c_t is computed as a weighted sum of the encoder hidden states, where the normalized alignment scores give the weights:

$$c_t = \sum_i \alpha_{ti} h_i$$

3.6.3 Integration with the Decoder

The context vector c_t is then combined with the decoder's previous hidden state h_{t-1} and used to generate the final output. This combination allows the decoder to make predictions based on the most relevant parts of the input sequence:

$$\text{Output}_t = \text{Decoder}(h_{t-1}, c_t)$$

Here, the decoder utilizes both the previous hidden state and the context vector to produce the output at the current time step. This approach ensures that the decoder can focus on different parts of the input sequence dynamically, leading to improved performance in tasks such as Dealing with uncertainty in Linked Data, machine translation, and text summarization.

3.7 Conclusion

In this chapter, we explored the advances in deep learning techniques relevant to handling various challenges in linked data. We examined how sophisticated architectures like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) address the limitations of traditional Recurrent Neural Networks (RNNs), particularly in managing long-range dependencies and mitigating the vanishing gradient problem. By understanding the functionality of these models, we can better apply them to complex tasks such as detecting semantic links, identifying types, and managing sameAs links in linked data.

The chapter also covered the encoder-decoder model, a powerful architecture for sequence-to-sequence tasks. This model's ability to generate output sequences from input sequences provides a strong foundation for various linked data applications. The introduction of attention mechanisms further enhances the performance of these models by allowing them to focus on relevant parts of the input sequence, thereby improving their effectiveness in processing long sequences and generating accurate results.

The subsequent contributions chapters will apply these deep learning techniques to tackle specific linked data challenges. The focus will be on leveraging these advanced models to detect links, types, sameAs links, and errors in linked data. By integrating these approaches, we aim to enhance the quality and reliability of linked data systems, paving the way for more accurate and meaningful data analysis and management.

Chapter 4

Overview of Related Work for Uncertainty Resolution in Linked Data

4.1 Introduction

In the evolving landscape of data management, Linked Data has become a pivotal method for connecting and integrating information from diverse sources across the web. However, the inherent complexity of Linked Data systems introduces various sources of uncertainty, which can impact the completeness and usability of the data.

This chapter provides a comprehensive literature review on uncertainty in Linked Data, focusing on key areas such as link detection, type detection, sameAs link detection, and error detection. By examining the advantages and limitations of current approaches and synthesizing their contributions, we highlight the gaps in existing research. This exploration not only clarifies the challenges faced in these critical areas but also sets the stage for the presentation of our novel contributions in the subsequent chapter.

The related work is organized as follows: we begin by reviewing the literature on type detection in Linked Data, where we analyze the techniques used to classify and infer types within data sets. Next, we focus on sameAs link detection in Linked Data, a critical aspect of ensuring data consistency across diverse sources. Following this, we explore semantic link detection in Linked Data, structured into three subsections: the handling of incompleteness in Linked Data, link detection methodologies, and the application of deep learning techniques for link detection. Finally, the last section is dedicated to error detection in Linked Data. In each part, we present the relevant studies along with a detailed analysis, highlighting the strengths and limitations of the various approaches.

4.2 Types Detection in Linked Data

In the context of Linked Data, type detection plays a crucial role in understanding and organizing data within RDF datasets. By accurately predicting types for entities, we can enhance data integration, querying, and inferencing capabilities, which are essential for a wide range of applications, including knowledge graphs, semantic search, and data analytics. The challenge lies in the heterogeneity and incompleteness of RDF datasets, which often lack explicit type information or contain noisy data. Various approaches have been proposed to address this problem, leveraging different techniques ranging from statistical heuristics to advanced machine learning and deep learning models. This section reviews notable works in this area, highlighting their methodologies, strengths, and limitations.

In this part, we will explore a number of related works on type prediction in RDF datasets.

- Statistical Heuristic Link-based Type Prediction: Paulheim [6] proposed a statistical heuristic link-based type prediction mechanism evaluated on DBpedia.
- Supervised Hierarchical SVM Classification: Kliegr et al. [66] introduced a supervised hierarchical SVM classification approach for DBpedia by exploiting the contents of Wikipedia articles.
- Multi-label Classification using Word Embedding: Biswas et al. [67] presented a multi-label classification algorithm based on word embeddings such as Word2Vec, FastText, and GloVe to capture the semantic relationships between entities and relations.
- Class Assignment Detector: Barati et al. [68] developed the Class Assignment Detector to identify correct and incorrect class assignments for entities in RDF data by analyzing class characteristics.
- Type Prediction using Twitter Profiles: Nechaev et al. [69] addressed the type prediction problem by using data extracted from the Twitter profiles of RDF entities as features in training data.
- Predicting Infobox Types using Word and Network Embedding: Biswas et al. [70] used word embedding and network embedding to predict infobox types for Wikipedia articles, which aids in the type generation procedure for RDF entities.
- Binary Classifier using Structural Data: Mihindukulasooriya et al. [71] proposed a binary classifier using structural data and machine learning techniques to predict RDF entity types.

- Text Classification for Type Prediction: Zhang et al. [72] introduced an approach for type prediction through text classification, employing two classifiers to accomplish this task.
- Quality Control of Protein Databases: Kondratyeva et al. [73] described an approach for quality control of protein databases, highlighting the challenges posed by data incompleteness in datasets like UniProt.

4.2.1 Analysis of Related Work for Types Detection in Linked Data

Table 4.1 presents a comprehensive multi-criteria analysis of literature on Types Detection in Linked Data. This summary aids in understanding the features, benefits, and limitations of each approach.

Examining these approaches reveals several limitations:

- Most works do not account for the semantic relationships between the different components of triples.
- Various techniques used in related works include supervised hierarchical SVM classification, statistical heuristics, machine learning techniques, text classification, and word embedding.
  - SVM Algorithm: Not suitable for large datasets and high feature counts.
  - Feature Extraction in Machine Learning: Manual extraction by programmers can lead to lower data quality. This task is automated in neural networks, which extract more significant features, improving results.
  - Word Embedding and Statistical Heuristics: These methods do not extract significant features or semantic relationships between triples, negatively impacting result quality.
- Results can be improved by proposing alternative solutions, particularly by employing methods that address issues of scalability and the semantic relationships between entities.
- Several works test methods on subsets of DBpedia data, but specifics are often not mentioned.
- Semantic relationships between different types (classes) and resources are generally not addressed.

- Proposed methods do not assign weights to triples based on their importance during type detection tasks.

To address these limitations, we propose an encoder-decoder neural network with embedding to explore semantic relationships between different components of RDF triples. Deep learning models' ability to learn from large amounts of data, combined with an attention mechanism to weight inputs by their importance, can significantly enhance type detection accuracy.

Table 4.1: Comparison of Approaches for Types Detection in Linked Data

Work	**Approach**	**Techniques**	**Datasets**	**Metrics**	**Limitations**
Paulheim [6]	Statistical Heuristic	Heuristic Links	DBpedia, OpenCyc	F-measure, Precision, Recall, Precision@95%, Accuracy	Limited by heuristic rules, may not generalize well to diverse datasets
Kliegr et al. [66]	Hierarchical SVM	Supervised Learning	DBPedia	Hierarchical Precision, Hierarchical Recall, Hierarchical F-measure	Not scalable for large datasets with high feature dimensionality
Biswas et al. [67]	Word Embedding	Multi-label Classification	DBPedia	Hits@1, Hits@3	Depends on quality of word embeddings, may not capture complex semantic relations
Barati et al. [68]	Class Characteristics	Entropy Analysis	DBPedia	Accuracy CA, Accuracy ICA	Limited to detecting incorrect class assignments, does not predict new types

Table 4.1: Comparison of Approaches for Types Detection in Linked Data

Work	Approach	Techniques	Datasets	Metrics	Limitations
Nechaev et al. [69]	Twitter Profiles	Profile Features	DBPedia, Twitter Data	Recall, Precision, F-measure	Relies on availability and accuracy of external social media data
Biswas et al. [70]	Word & Network Embedding	Embedding Techniques	DBPedia	F-measure, TF-IDF, Accuracy	Complexity in combining word and network embeddings, computationally intensive
Mihindukulasooriya et al. [71]	Structural Data & ML	Binary Classification	DBPedia	Recall, Precision, F-measure, Accuracy	Limited to binary classification, may not handle multi-class type prediction
Zhang et al. [72]	Text Classification	Dual Classifiers	N/A	N/A	Requires extensive text data, performance depends on text quality and availability
Kondratyeva et al. [73]	Quality Control	Data Analysis	Uniprot, EcoCyc	N/A	Focused on data quality, does not directly address type prediction

4.3 SameAs links detection in linked data

SameAs links are a type of link used in linked data to connect resources that refer to the same concept or entity. These links can be used to connect data from different sources that use different identifiers, such as a person's name, a unique identifier, or a URL.

SameAs links facilitate the integration of data from various sources, making it easier to query and analyze data collectively. Often used alongside other types of links like owl:sameAs (indicating semantic equivalence) and skos:exactMatch (indicating identical concepts), SameAs links are pivotal in enhancing data connectivity and interoperability in RDF datasets.

Numerous studies in the Semantic Web community have focused on linking RDF datasets. These studies aim to improve the connectivity and interoperability of RDF datasets by identifying and linking related resources. In this section, we present some of the existing works on linking RDF datasets, highlighting the various methodologies and challenges encountered.

- Ding et al. [74] analyzed the deployment status and implications of SameAs links in linked data. Their work highlights the challenges associated with the incorrect use of SameAs links, which can lead to erroneous inferences in the linked data ecosystem.

- Raad et al. [75] proposed a method for detecting contextual identity links in a knowledge base. They emphasized the importance of considering context when linking entities, as ignoring context can result in incorrect identity links

- Al-Moslmi et al. [5] introduced a framework using both syntactic and semantic features to identify contextual identity links. They utilized probabilistic Datalog and an inference algorithm, ProbFR, for uncertain reasoning in inferring SameAs facts.

- Euzenat and Shvaiko [76] conducted a comprehensive survey of ontology matching techniques. They discussed linguistic-based, structure-based, and instance-based ontology matching techniques.

- Faria et al. [77] developed AgreementMakerLight (AML), an ontology-matching framework. They utilized matching algorithms like lexical, string similarity, and cross-referencing methods for large biomedical ontologies.

- Saïs et al. [78] introduced methodologies for automated invalidation of links, focusing on detecting erroneous owl:sameAs links. They proposed a novel data linking approach incorporating explicit contexts and discussed data enrichment in linked data and knowledge graphs.

- Suchanek et al. [79] proposed PARIS, a probabilistic algorithm for the automatic alignment of ontologies by matching instances, classes, and relations simultaneously. They achieved high precision in large-scale ontology experiments without manual input or parameter tuning.

4.3.1 Analysis of Related Work for SameAs Links Detection in Linked Data

Table 4.2 provides a summary of a multi-criteria analysis of literature works on SameAs link detection in Linked Data. This synthesis helps to understand the characteristics, advantages, and limitations of each approach.

Reviewing these approaches highlights various limitations:

- SameAs link-based approaches [5, 74, 75, 78] :
  - Difficulty in accurately identifying and discovering SameAs links automatically.
  - Limited consideration of contextual information, leading to potential erroneous SameAs links.
  - Lack of probabilistic reasoning and uncertainty management in inferring and validating SameAs links. This limitation means that many approaches do not quantify the uncertainty associated with SameAs links, which can lead to lower confidence in their accuracy and effectiveness. Without probabilistic frameworks, it becomes challenging to assess the correctness of inferred links, especially in contexts where data is noisy or incomplete.
- Ontology matching approaches [76, 77, 79] :
  - Scalability issues, particularly with large-scale ontologies.
  - Heterogeneity of ontologies necessitating hybrid approaches that combine linguistic, structural, and instance-based techniques.
  - Difficulty in handling evolving ontologies, requiring continuous updates and maintenance.
  - Reliance on probabilistic estimates that may not accurately capture complex semantic relationships, reducing effectiveness with highly heterogeneous ontologies.

To overcome these limitations, we propose a Siamese deep learning approach for detecting sameAs links between entities. Deep learning models' capability to learn from extensive datasets greatly improves the accuracy of sameAs link detection.

Table 4.2: Comparison of Approaches for SameAs Links Detection in Linked Data

Work	Approach	Methodology	Dataset	Metrics	Limitations
Ding et al. [74]	Analysis of SameAs links in Linked Data	Examination of SameAs links and their implications in Linked Data	Billion Triple Challenge 2010	N/A	Limited to the analysis of deployment; does not address contextual information or probabilistic reasoning, which can affect the confidence in link validity.
Raad et al. [75]	Detection of contextual identity links	Proposed method for detecting identity links with contextual information	CellExtra-Dry, Carredas	Rule's average error rate, Rule's support	Does not address scalability, probabilistic reasoning, or the handling of evolving ontologies
Al-Moslmi et al. [5]	Contextual identity link detection using syntactic and semantic features	Utilized probabilistic Datalog and inference algorithms for SameAs detection	DBPedia, MusicBrainz, INA	Recall, Precision, F-measure	Limited scalability; challenges in handling heterogeneity and evolving ontologies

Table 4.2: Comparison of Approaches for SameAs Links Detection in Linked Data

Work	Approach	Methodology	Dataset	Metrics	Limitations
Euzenat and Shvaiko [76]	Comprehensive survey on ontology matching techniques	Discussed various ontology matching techniques (linguistic, structural, instance-based)	N/A	N/A	Scalability issues, difficulty in handling evolving ontologies, reliance on probabilistic estimates
Faria et al. [77]	Ontology matching framework (AML)	Utilized matching algorithms for large-scale biomedical ontologies	Anatomy and Large Biomedical Ontologies tracks of the OAEI 2012	Recall, Precision, F-measure	Scalability challenges, handling of highly heterogeneous ontologies, dependence on parameter tuning
Saïs et al. [78]	Automated invalidation of SameAs links	Focused on detecting erroneous SameAs links with contextual information	CellExtra-Dry, Carredas	Error rate, Support	Does not address scalability, limited in managing evolving ontologies, and challenges in erroneous link detection

Table 4.2: Comparison of Approaches for SameAs Links Detection in Linked Data

Work	Approach	Methodology	Dataset	Metrics	Limitations
Suchanek et al. [79]	Probabilistic algorithm (PARIS) for ontology alignment	Matching of instances, classes, and relations simultaneously	DBPedia, Yago	Recall, Precision, F-measure	Challenges in handling large-scale ontologies without manual input; effectiveness varies with highly heterogeneous ontologies

4.4 Links Detection in Linked Data

Incompleteness is a type of uncertainty in Linked Data that arises when some information about the world is missing or incomplete. There are various types of incompleteness, such as missing data, missing links, missing triples, etc. Incompleteness within Linked Data primarily stems from inherent limitations in the source data itself. Extracting heterogeneous data from various formatted datasets during integration can further exacerbate this issue.

Several approaches have been proposed in the literature to address missing data in Linked Data, particularly focusing on discovering missing links and predicting missing entity types. In the following section, we present the most relevant works dealing with these issues.

Discovering links is one of the most important tasks in addressing the issue of incompleteness in Linked Data. Link discovery involves predicting missing links between RDF resources, essentially enriching datasets. In this section, we will cite and discuss the most important approaches from the literature that address the problem of link discovery and incompleteness in Linked Data.

4.4.1 Incompleteness in Linked Data

Link discovery is considered an important solution for tackling incompleteness in Linked Data, which manifests as missing information, triplets, links, etc. The proposed ap-

proaches deal with detecting statements that can be repaired, predicting missing types, etc. The main challenges in this task go beyond data quality and include issues such as scalability, heterogeneity of data sources, and the ambiguity in the semantics of data. The most pertinent related works are presented below:

- The Missing Path: [80] proposed a visualization tool to identify coherent subsets of entities that can be repaired in knowledge graphs, showing the importance of user-friendly tools in improving data completeness
- Completeness Framework: [81] proposed a framework for inserting statements into RDF data to express completeness in knowledge graphs, emphasizing how critical it is to establish formal mechanisms for ensuring knowledge graph completeness.
- ProWD: [82] addressed the completeness profiling problem in Wiki-data using the Class-Facet-Attribute profiles, demonstrating that structuring knowledge graph completeness by facets is crucial for understanding data gaps.
- Meta-information in Wiki-data: [83] demonstrated how to use meta-information to provide data completeness information on Wiki-data, showcasing the value of leveraging meta-data to offer insights into the reliability and completeness of information.
- Type Incompleteness: As presented earlier, the issue of type incompleteness has been addressed through various methods, including statistical heuristics proposed by [6], classification approaches introduced by [67], [66], and [71], text classification by [72], Twitter profile information by [69], and word and network embedding techniques by [70].

4.4.2 Link Discovery in Linked Data

In recent years, link discovery in Linked Data has become increasingly important. However, it also presents significant challenges. Data quality issues like incompleteness, noise, or inconsistencies can significantly hinder the accuracy of link prediction. Additionally, processing large datasets or limitations of specific algorithms can lead to runtime challenges. Most approaches in the literature employ similarity measures to tackle the dataset alignment problem, which involves detecting sameAs links (as presented earlier). The most important approaches in the literature are listed below:

- NAISC: [84] presented Nearly Automatic Integration of Schemas, a method for linking datasets based on a combination of textual and structural similarity measures for ontology alignment.

- LIMES: [85] proposed a framework for link discovery in the semantic web, addressing time efficiency and improving link discovery quality with Machine Learning solutions.
- HLSD: [86] introduced a Hybrid Similarity measure for Linked Data, combining several similarity measures and exploiting information found in RDF literals and existing links between resources.
- Probabilistic Framework: [5] proposed a probabilistic framework for modeling and reasoning on uncertain data, aiming to find sameAs links with an inference algorithm based on RDF facts and rules.
- Context-Independent Approach: [87] proposed a context-independent approach for Linked Data alignment based on ontological model components and data dimensions, executing alignment directly on data sources.
- Semantic Distance Measures: [88] used a set of semantic distance measures to determine relatedness between resources, applicable in recommendation systems.
- Structural and Semantic Approach: [89] proposed a structural and semantic approach for link prediction, generating explanations for identified linked entities.
- Entity Interlinking Framework: [90] presented a framework for entity interlinking while tackling various ambiguity issues within Linked Data.

4.4.3 Deep Learning Methods for Links Detection in Linked Data

In this subsection, we will present related works that use Deep Learning methods to tackle the problem of detecting links in Linked Data.

- Encoder-Decoder Neural Network: [7] proposed an encoder-decoder neural network using textual descriptions of entities in Knowledge graphs.
- Link Prediction: [8] proposed a method for predicting links in knowledge graphs by studying the representation of relations and entities.
- Hyper-network Architecture: [9] introduced a hyper-network architecture generating simplified relation-specific convolutional filters to generate missing links in knowledge graphs.
- Convolutional Neural Networks: [91, 92] proposed using Convolutional Neural Networks for link detection.

- Translating Embeddings: [93] proposed a method for modeling relations by interpreting them as translations operating on the low-dimensional embedding of the entities.

4.4.4 Analysis of Related Work for Links Detection in Linked Data

Tables 4.3, 4.4 and 4.5 offer a comprehensive overview of a multi-criteria analysis of literature on incompleteness and detecting links in Linked Data. This summary aids in understanding the key characteristics, strengths, and limitations of the various approaches.

Examining various approaches to the problem of link discovery and incompleteness in Linked Data has allowed us to identify their limitations and characteristics. This analysis aids us in proposing our contributions. The main limitations of the proposed approaches are cited below:

- Single Link Type Detection: The majority of proposed methods detect only one type of link. Some works address the dataset alignment problem, which consists of finding the sameAs links between the resources of various datasets [5]. Other works propose solutions for a single type of link, such as missing type prediction [6, 66, 67, 69–72]. These works do not support other types of semantic links.

- Incorrect Link Prediction: The task of link discovery may produce incorrect links. This issue, identified in several evaluations of existing methods, must be addressed to ensure the quality of the links.

- Dataset Alignment Limitation: Some related works propose solutions for dataset alignment. These approaches do not address the problem of detecting links between resources in the same dataset [5].

- Overlooked Hidden Relations: Link discovery often overlooks the hidden relations between triples in the dataset. For example, [7] uses paths and textual descriptions of entities while ignoring semantic relations between subject and object components.

- Neural Network Data Weighting: The proposed approaches based on neural networks do not give higher weights to more meaningful data when discovering links [8, 9, 91–93].

- Textual Description Dependence: The method presented in [7] predicts links using textual descriptions of entities. Textual descriptions of some entities may be less detailed.

In order to deal with these limitations, we propose our contribution, which consists of using advanced Deep Learning techniques to discover all types of links.

Table 4.3: Comparison of Approaches for Incompleteness in Linked Data

Work	**Approach**	**Key Contributions**	**Dataset**	**Metrics**	**Limitations**
Destandau et al. (2021) [80]	Visualization tool	Identification of coherent subsets of entities that can be repaired in knowledge graphs	Knowledge Graphs	N/A	Limited to visualizing subsets of entities, may not handle large-scale datasets effectively
Darari et al. (2018) [81]	Framework for RDF data completeness	Inserting statements into RDF data to express completeness in knowledge graphs	DBPedia	N/A	Framework complexity and scalability issues
Wisesa et al. (2019) [82]	ProWD framework	Completeness profiling in Wiki-data using Class-Facet-Attribute profiles	Wikidata	N/A	Specific to Wiki-data, may not generalize well to other datasets
Prasojo et al. (2016) [83]	Meta-information usage	Providing data completeness information on Wiki-data	Wikidata	N/A	Dependence on quality and availability of meta-information
Paulheim (2013) [6]	Statistical heuristics	Addressing type incompleteness	DBpedia, OpenCyc	F-measure, Precision, Recall, Precision@95%, Accuracy	Heuristics may not capture all nuances, potentially less accurate for complex datasets

Table 4.3: Comparison of Approaches for Incompleteness in Linked Data

Work	**Approach**	**Key Contributions**	**Dataset**	**Metrics**	**Limitations**
Biswas et al. (2020) [67]	Classification methods	Entity type prediction	DBPedia	Hits@1, Hits@3	Classification accuracy depends on feature quality, may not handle all entity types well
Kliegr et al. (2016) [66]	Classification methods	Entity type prediction	DBPedia	Hierarchical Precision, Hierarchical Recall, Hierarchical F-measure	Similar to other classification methods, limited by training data quality
Mihindukulasooriya et al. (2018) [71]	Classification methods	Entity type prediction	DBPedia	Recall, Precision, F-measure, Accuracy	Classification methods may struggle with ambiguous or incomplete data
Zhang et al. (2017) [72]	Text classification	Entity type prediction	N/A	N/A	Dependent on the quality of textual data, may not generalize across different text corpora
Nechaev et al. (2018) [69]	Twitter profile information	Entity type prediction	DBPedia, Twitter Data	Recall, Precision, F-measure	Limited to Twitter data, which do not represent broader entity characteristics

Table 4.3: Comparison of Approaches for Incompleteness in Linked Data

Work	Approach	Key Contributions	Dataset	Metrics	Limitations
Biswas et al. (2018) [70]	Word and network embedding	Wikipedia article type prediction	DBPedia	F-measure, TF-IDF, Accuracy	Embedding methods depend on comprehensive training data, do not capture rare types

Table 4.4: Comparison of Approaches for Link Discovery

Work	Approach	Key Contributions	Dataset	Metrics	Limitations
McCrae et al. (2018) [84]	NAISC	Linking datasets using textual and structural similarity measures for ontology alignment	WordNet-Wikipedia, OAEI, SemEval STS	Precision, Recall, F-measure	May produce incorrect links, relies heavily on textual and structural data quality
Ngonga Ngomo et al. (2021) [85]	LIMES	Time-efficient link discovery, Machine Learning solutions for improving link discovery quality	DBPedia	Precision, Recall, F-measure	Scalability issues with very large datasets, potential for incorrect link predictions
Silva et al. (2020) [86]	HLSD	Combining similarity measures, exploiting RDF literals and existing links	DBpedia	Precision	Overlook hidden relations, depends on the quality of RDF literals and existing links

Table 4.4: Comparison of Approaches for Link Discovery

Work	**Approach**	**Key Contributions**	**Dataset**	**Metrics**	**Limitations**
Al-Moslmi et al. (2016) [5]	Probabilistic framework	Finding sameAs links using ProbFr, an inference algorithm based on RDF facts and rules	DBPedia, MusicBrainz, INA	Recall, Precision, F-measure	Complexity in handling probabilistic data, the potential for scalability issues
Barbosa et al. (2022) [87]	Context-independent approach	Linked Data alignment using ontological model components and data dimensions	OAEI	Precision, Recall, F-measure	Limited by the quality of ontological models and data dimensions
Piao et al. (2016) [88]	Semantic distance measures	Relatedness determination between resources, application in recommendation systems	Linked Open Dataenabled recommender systems challenge Dataset, DBpedia	Precision, Recall, MRR	Dependence on accurate semantic distance measures, struggles with noisy data
D'Aquin et al. (2021) [89]	Structural and semantic approach	Generating explanations for identified linked entities, enhancing understanding of the prediction process	FB15k-237, WN18, DBpedia15k	Recall, Avg support	Potential for incorrect explanations, complexity in structural and semantic integration

Table 4.4: Comparison of Approaches for Link Discovery

Work	Approach	Key Contributions	Dataset	Metrics	Limitations
Achichi et al. (2019) [90]	Entity interlinking framework	Tackling ambiguity issues within Linked Data	OAEI	Precision, Recall, F-measure	Potential ambiguity in entity interlinking, relies on accurate initial data

Table 4.5: Comparison of Approaches for Deep Learning Methods for Links Detection in Linked Data

Work	Approach	Key Contributions	Dataset	Metrics	Limitations
Biswas et al. (2021) [7]	Encoder-decoder neural network	Predicting links using textual descriptions of entities in Knowledge graphs	DBPedia, FB15K, FB15K-237, WN18, WN18RR, YAGO3-10	MRR, Hits@1, Hits@3, Hits@10	Dependent on the quality of textual descriptions, potential issues with less detailed entities
Sun et al. (2019) [8]	Relation and entity representation study	Inferring semantic relations in knowledge graphs	FB15K, FB15K-237, WN18, WN18RR, YAGO3-10	MRR, Hits@10	Overlooks important contextual information, complexity in representation learning

Table 4.5: Comparison of Approaches for Deep Learning Methods for Links Detection in Linked Data

Work	**Approach**	**Key Contributions**	**Dataset**	**Metrics**	**Limitations**
Balazevic et al. (2019) [9]	Hyper-network architecture	Generating missing links in knowledge graphs using relation-specific convolutional filters	FB15K, FB15K-237, WN18, WN18RR, YAGO3-10	MR, MRR, Hits@1, Hits@3, Hits@10	Potential for overfitting, dependence on high-quality training data
Nguyen et al. (2017) [91]	Convolutional Neural Network	Links detection using CNNs	WN18RR, FB15K-237	MR, MRR, Hits@10	High computational cost, requires large training datasets
Dettmers et al. (2018) [92]	Convolutional Neural Network	Links detection using CNNs	FB15K, FB15K-237, WN18, WN18RR, YAGO3-10	MR, MRR, Hits@1, Hits@3, Hits@10	Similar limitations to other CNN-based methods, potential for overfitting
Bordes et al. (2013) [93]	Translation-based relation modeling	Modeling relations by interpreting them as translations on low-dimensional embeddings of entities	WN, FB15K, FB1M	MR, Hits@10	Limited by the quality of embedding, potential for inaccurate translations

4.5 Errors detection in linked data

Error detection in linked data has been an active area of research in recent years. Several approaches have been proposed to identify different types of errors and inconsistencies. In this section, we provide a comprehensive overview of the state of the art in Linked Data error detection, highlighting the main contributions and limitations of existing approaches.

- Debattista et al. [94]: Propose a preliminary approach for identifying potentially incorrect RDF statements in Linked Data to improve data quality. They focus on issues such as incomplete, misrepresented, and noisy data, particularly addressing undefined domains and ranges for properties.

- Topper et al. [11]: Introduce a method for identifying inconsistencies in the DBpedia dataset by enriching the DBpedia ontology using statistical methods. This enriched ontology is used to detect contradictions, identify incorrect facts, and generate correction suggestions.

- Melo et al. [95]: Present a hybrid approach called PaTyBRED for detecting relation assertion errors in knowledge graphs. This approach combines type and path features into local relation classifiers, aiming to address the limitations of methods relying solely on types.

- Li et al. [96]: Propose a probabilistic framework for detecting errors in numerical attributes in Linked Open Data. They employ an unsupervised error detection method, considering attribute relations.

- Paulheim et al. [97]: Introduce an unsupervised approach using multi-dimensional outlier detection to identify erroneous links between datasets in Linked Open Data. They focus on specific link types and suggest incorporating semi-supervised techniques or automatic parameter tuning.

- Liu et al. [98]: Propose Triples Accuracy Assessment (TAA) for validating RDF triples in knowledge graphs. This method finds consensus from other knowledge graphs, although it relies on external knowledge graphs and needs efficiency evaluation for large-scale datasets.

- Rico et al. [99]: Develop a data-driven approach to automatically detect incorrect mappings in DBpedia. They use machine learning to analyze instance data and ontological axioms but face challenges with data incompleteness and variations in representation.

- Wienand et al. [100]: Focus on detecting incorrect numerical data in DBpedia by applying unsupervised outlier detection methods and clustering. The approach is limited to numerical data and does not address other data types.

- Zhao et al. [101]: Propose a graph-based approach combined with ontology integration and word embedding techniques to detect and correct missing and incorrect RDF triples in Knowledge Graphs (KGs). However, they lack a comprehensive evaluation of the discovered knowledge and do not address the identification of owl links.

4.5.1 Analysis of Related Work for Errors Detection in Linked Data

Table 4.6 offers a summary of a multi-criteria analysis of the literature on Error Detection in Linked Data. This overview facilitates an understanding of the characteristics, benefits, and limitations associated with each method.

The main limitations of the approaches cited are:

- Statistical and Manual Correction Approaches: Studies like those by Topper et al. [11] and Debattista et al. [94] rely on statistical methods and manual corrections. While effective in small-scale applications, these methods struggle with scalability in larger datasets, where manual intervention becomes impractical, leading to potential inconsistencies and biases.

- Hybrid and Probabilistic Frameworks: Approaches such as those by Melo et al. [95] and Li et al. [96] combine different features or utilize probabilistic frameworks. However, these methods are often limited by the quality of input data, which can vary significantly, and may not adequately capture the complexities of semantic relationships.

- Outlier Detection Techniques: Unsupervised outlier detection methods, including those by Paulheim et al. [97] and Wienand et al. [100], aim to identify anomalies within RDF triples. While useful, they can be constrained by their focus on specific error types and may struggle with high data dimensionality, complicating the distinction between true outliers and legitimate variations.

- Graph-Based Approaches: Graph-based methods, as seen in Zhao et al. [101], leverage structural relationships within data. However, they often face scalability challenges in large, complex graphs, where computational efficiency can decrease significantly.

- Machine Learning Models: Rico et al. [99] propose a data-driven approach using machine learning techniques. While promising, these models are typically constrained by the quality and completeness of training data, risking the perpetuation of existing inaccuracies without addressing underlying semantic relationships.

To tackle the limitations of current methods like scalability and extraction of semantic dependencies between entities, we introduce a new approach for detecting errors in Linked Data that leverages advanced deep learning techniques based on LSTM and Embedding.

Table 4.6: Comparison of Approaches for Errors Detection in Linked Data

Work	**Approach**	**Focus**	**Techniques**	**Dataset**	**Metrics**	**Limitations**
Debattista et al. [94]	Preliminary Approach	Incorrect RDF Statements	Addressed incomplete, misrepresented, and noisy data	DBPedia	Recall, Precision	Restriction to knowledge bases without blank nodes; Incomplete utilization of typed annotations; Limited applicability to hierarchical relations; Potential biases and uncertainty in sampling and results
Topper et al. [11]	Enrichment & Statistical Methods	DBpedia Inconsistencies	Statistical methods for ontology enrichment and contradiction detection	DBPedia	N/A	Reliance on manual correction of inconsistencies; Need for user interface for corrections; Further research needed on automatic triple detection

Table 4.6: Comparison of Approaches for Errors Detection in Linked Data

Work	Approach	Focus	Techniques	Dataset	Metrics	Limitations
Melo et al. [95]	Hybrid Approach (PaTyBRED)	Relation Assertion Errors	Combines type and path features into local classifiers	DBPedia, NELL	FMRR	Dependence on type and path information quality; Challenges in selecting relevant graph paths in large datasets
Li et al. [96]	Probabilistic Framework	Numerical Attributes	Unsupervised error detection considering attribute relations	DBPedia	Precision	No automatic error correction or missing value filling
Paulheim et al. [97]	Unsupervised Multi-dimensional Outlier Detection	Erroneous Links	Outlier detection methods for identifying erroneous links	DBPedia, DBTropes, Peel Sessions	AUC, F-measure	Limited evaluation with specific link types; Need for careful selection and parameterization of methods
Liu et al. [98]	Triples Accuracy Assessment (TAA)	RDF Triples Validation	Finds consensus from other knowledge graphs	DBPedia	F-measure	Reliance on external knowledge graphs; Efficiency evaluation required for large-scale datasets

Table 4.6: Comparison of Approaches for Errors Detection in Linked Data

Work	Approach	Focus	Techniques	Dataset	Metrics	Limitations
Rico et al. [99]	Data-driven Approach	Incorrect Mappings in DBpedia	Machine learning-based predictive model	DBPedia	TP Rate, FP Rate, Precision, Recall, F-measure, MCC, ROC Area, PRC Area	Challenges with data incompleteness and representation variations
Wienand et al. [100]	Unsupervised Outlier Detection	Numerical Data	Outlier detection methods and clustering	DBPedia	Precision	Focus on numerical data only; Does not address other data types
Zhao et al. [101]	Graph-based & Ontology Integration	Missing and Incorrect RDF Triples	Graph-based approach combined with ontology integration and word embedding	DBPedia, YAGO	N/A	Lack of comprehensive evaluation; Does not address owl links

4.6 Conclusion

This chapter has reviewed the current state of research on uncertainty in Linked Data, with a specific focus on linked detection, types detection, sameAs links detection, and errors

detection. Through a detailed examination of relevant works, we have identified both the strengths and limitations of existing methods, providing a synthesis that underscores the research gaps in each area. As we transition to the next chapter, we will introduce our innovative approaches designed to address these gaps, supported by a practical use case involving the UniProt dataset. Our contributions aim to enhance the accuracy and effectiveness of Linked Data systems, paving the way for more reliable and insightful data integration and analysis.

Part II

Contributions

Chapter 5

Encoder-Decoder Neural Network with Attention Mechanism for Types Detection in Linked Data

5.1 Introduction

In recent years, the Linked Open Data cloud has gained significant popularity. This success has led to the availability of numerous semantic datasets covering various domains on the Web in machine-understandable formats, primarily RDF (Resource Description Framework).

However, the rise of the semantic web and Linked Data has introduced challenges related to data uncertainty, such as imprecision and incompleteness. These issues stem from how datasets are constructed—often from incomplete data, heterogeneous formats, and semi-structured sources. These anomalies create problems when using uncertain data for reasoning, decision-making, and generating new knowledge.

According to [102], few approaches effectively utilize links among datasets, thus missing the full potential of the Semantic Web. Approaches that rely on a single dataset fail to leverage the comprehensive possibilities offered by Linked Data. This limitation is largely due to the complexity of link discovery in Linked Data applications.

To enhance the quality of RDF data, our focus is on addressing type incompleteness. Predicting missing types for dataset subjects can lead to more complete datasets, improving the results of applications utilizing these datasets. Our approach leverages the predicates and objects associated with subjects to predict their types. By employing an encoder-decoder model, we ensure the extraction of semantic relations between predicates and objects, enhancing the accuracy of type prediction. An attention mechanism is

incorporated to assign higher weights to significant inputs.

In this contribution, we apply our approach to the DBpedia dataset using advanced deep learning techniques. These techniques have recently been employed to solve various problems, with artificial intelligence systems utilizing deep learning for rapid computational tasks and complex problem-solving. Deep learning is particularly suitable for handling large datasets, offering tools for analyzing and interpreting Linked Data presented in RDF format.

In this chapter, we propose an encoder-decoder network for multi-labeling, incorporating an attention mechanism to model the links between data. Our approach aims to predict missing types for RDF entities based on their triples.

The remainder of this chapter is structured as follows: first, we introduce the research problem, followed by a detailed explanation of our proposed approach. Finally, we describe the experimental setup and present the results along with a discussion of the findings.

5.2 Type Detection Modeling

Our approach addresses the link detection challenge by framing it as a multi-label classification problem. In multi-label classification, the goal is to assign data points to multiple categories or classes where these categories are not mutually exclusive. This means that a single data point can belong to several classes simultaneously. Specifically, in multi-label classification, each example is associated with a set of labels Y from a larger set of disjoint labels L, such that $Y \subseteq L$.

In our model, the inputs consist of predicates $(P1, P2, ..., Pn)$ and objects $(O1, O2, ..., On)$ associated with a subject S. These inputs are utilized to predict the output, which denotes the types or categories of the subject S.

To provide context for our approach, we first address the issue of missing types within ontologies proposed by the World Wide Web Consortium (W3C) for modeling uncertain knowledge on the Semantic Web. This context, illustrated in Figure 5.1, highlights the challenges of incompleteness in existing ontologies. We tackle this problem using deep learning techniques to process and address these gaps.

Detecting links within Linked Data is crucial for identifying resource classes and uncovering new connections between datasets, which helps to minimize data incompleteness. By automatically detecting missing types, our solution enhances data quality and ensures more reliable query responses across applications that utilize Linked Data.

Our approach involves predicting types of resources within RDF datasets based on their predicates and object values. This deep learning-based model was trained using the

DBpedia dataset, with the training phase conducted on Google Collaboratory.

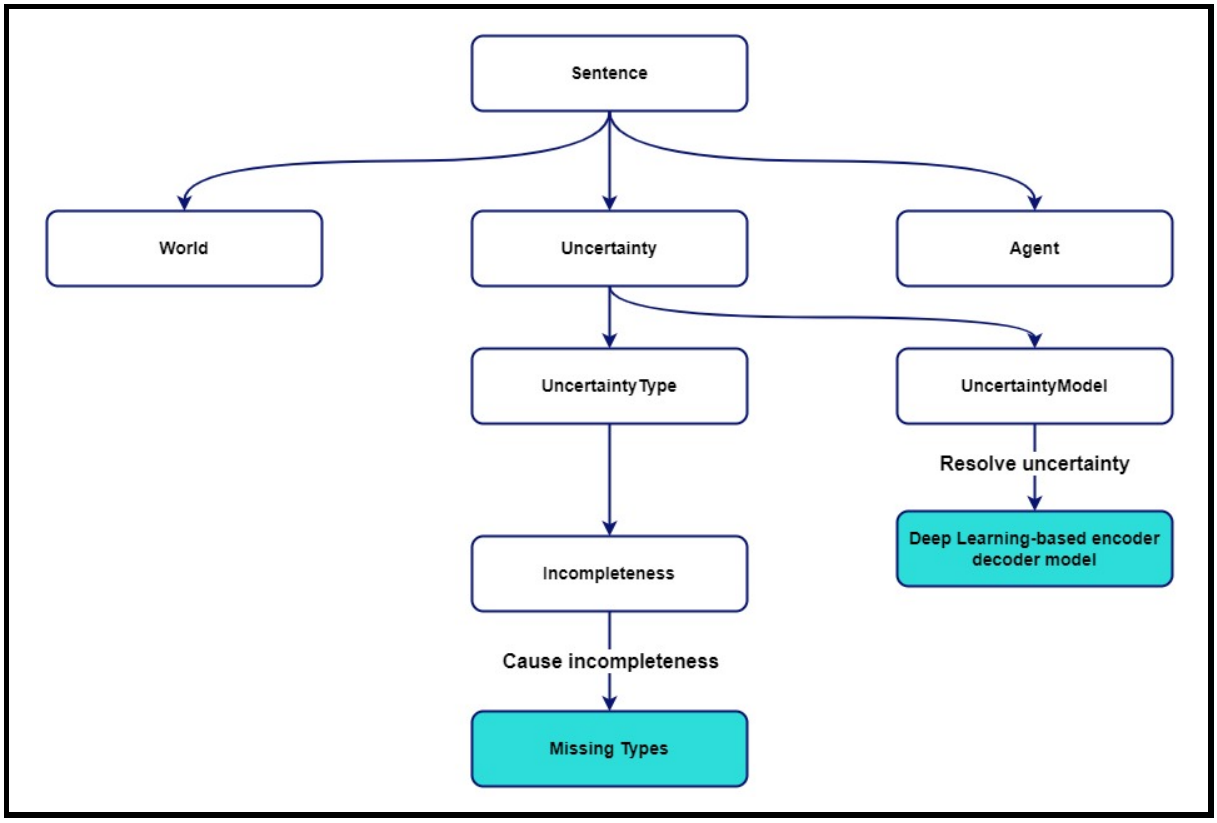


Figure 5.1: Types detection in uncertainty ontology [103]

5.3 Architecture of the Model

5.3.1 Pipeline

The various stages involved in constructing our model are detailed in Fig. 5.2 and in Algorithm 5.1.

- Dataset Preparation: To train our model, we utilize the DBpedia dataset, which adheres to the Linked Data principles [1]. This dataset comprises triples in the form of (Subject, predicate, object), where the predicate establishes the connection between the subject and the object.

  Our objective is to predict missing types $(T1, T2, \ldots, Tn)$ for subjects S_i using their predicates $(P1, P2, \ldots, Pn)$ and objects $(O1, O2, \ldots, On)$. For a given subject S_i, we use all associated predicates and objects as inputs to predict its potential classes

[1] http://gaia.infor.uva.es/hdt/dbpedia2016-10/dbpedia2016-10.hdt

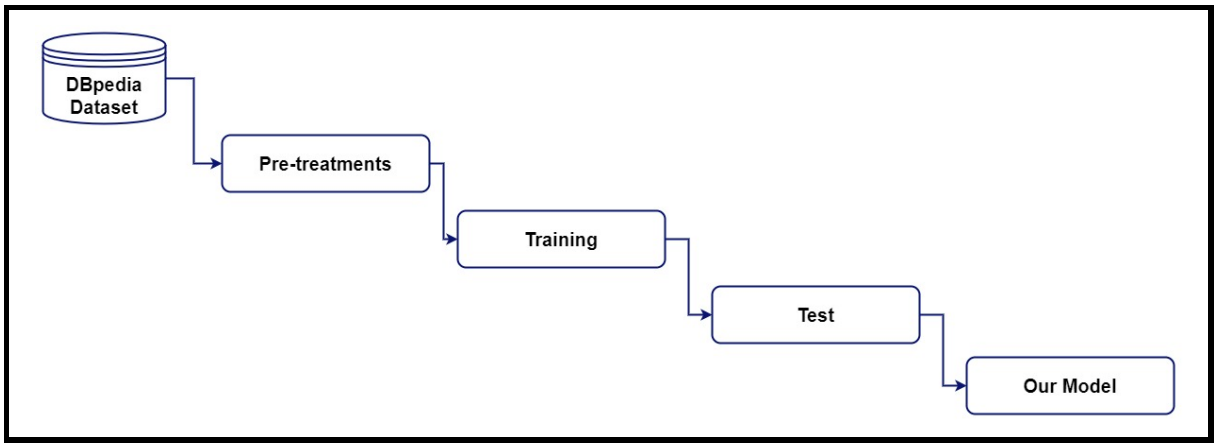


Figure 5.2: Steps of our approach to type detection for Linked Data [103]

or types. By inferring these new triples, we aim to enhance the completeness of the dataset.

The process begins by reading the dataset and selecting the relevant triples for the various learning phases. It then transforms the triples into a numerical format suitable for deep learning models.

Subsequently, during the pre-processing phase, we convert these triples into a format compatible with deep learning models, ensuring they are numerically represented.

Finally, we constrain the number of predicates and objects associated with each subject to 205, while the outputs are the possible types.

- Pre-treatments: Data preprocessing is a critical step in Machine Learning and Deep Learning, as the quality and format of the data can significantly impact model performance. In this step, we transform the raw data into a format that is more suitable for the model to learn from.

  For our model, data preprocessing involves converting each subject, object, and predicate into numerical representations. This transformation is essential because machine learning models, including deep learning networks, require numerical input to process data effectively. By assigning unique numbers to each subject, predicate, and object, we create a structured format that can be fed into the model.

  Figure 5.3 illustrates this transformation process, demonstrating how each component of the triples is converted into a numerical format. This numerical representation allows the model to process the data more efficiently and accurately, setting the stage for effective training and predictions.

Algorithm 5.1: Dataset Preparation and Preprocessing for Types Detection

Input : DBpedia dataset $\mathcal{D} = \{(S_i, P_i, O_i)\}$
Output: Numerically encoded dataset for deep learning model

Step 1: Dataset Preparation
Read the DBpedia dataset and extract RDF triples (S_i, P_i, O_i).
Select relevant triples for training and validation.
Define prediction task: infer missing types $(T_1, T_2, ..., T_n)$ for subjects S_i based on predicates and objects.
Restrict the number of predicates and objects per subject to 205.

Step 2: Preprocessing
foreach *triple* $(S_i, P_i, O_i) \in \mathcal{D}$ **do**
  Convert S_i, P_i, and O_i into unique numerical representations.
  Encode predicates and objects as input features.
  Assign possible types $(T_1, T_2, ..., T_n)$ as outputs.

Step 3: Format Conversion
Transform numerical representations into a structured format suitable for deep learning models.

Return: Preprocessed dataset ready for model training.

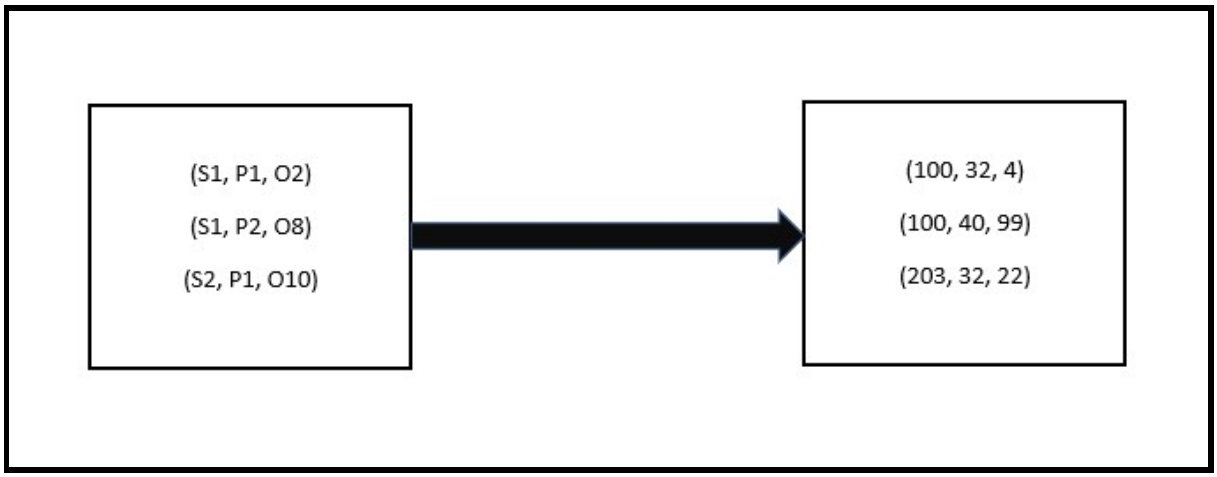


Figure 5.3: An example of the preprocessing process [103]

5.3.2 Our model proposal

Our model, named ED-MTD for (Encoder Decoder for Missing Types Detection) utilizes an encoder-decoder architecture with an attention mechanism and a neural network design pattern that supports generating an output sequence for each input sequence.

As depicted in Figure 5.4, the architecture comprises two components: the encoder and the decoder. Both components employ deep neural networks, specifically gated recurrent

units (GRUs), to manage input sequences of varying lengths.

1. Encoder-Decoder Architecture: The encoder-decoder architecture consists of two main components: the encoder and the decoder. Both components utilize gated recurrent units (GRUs) to handle sequences of variable length.

    - Encoder: The encoder processes the input sequences (predicates and objects) and converts them into a fixed-length vector that captures the essential information of the inputs. This vector is then used as the initial hidden state for the decoder.
    - Decoder: The decoder generates the output sequences (types) based on the information from the encoder. It consists of a GRU layer and a SoftMax layer:
        - GRU Layer: The GRU layer in the decoder takes the last hidden state from the encoder, the previous output, and the embedding vector as inputs to generate the next type of prediction.
        - SoftMax Layer: This layer predicts a probability distribution over all possible types and selects the type with the highest probability.

2. Embedding Layer: The embedding layer is crucial for reducing the dimensionality of the inputs while preserving the semantic relationships between the components of the subject. This layer helps in extracting more meaningful features from the data, enhancing the model's ability to make accurate predictions.

3. Gated Recurrent Units (GRUs): Gated Recurrent Units (GRUs) are used to address the vanishing gradient problem commonly encountered in recurrent neural networks. GRUs consist of two gates: the reset gate and the update gate. Unlike LSTM networks, which use a cell state and three gates, GRUs rely on a simpler mechanism but still manage to process long sequences effectively.

4. Attention Mechanism: The attention mechanism is a technique that improves the performance of sequence-to-sequence models, particularly in handling long sequences. It calculates attention weights for each input, allowing the model to focus on the most relevant parts of the input when generating predictions. This mechanism solves the issue of relying solely on the last hidden state from the encoder, which can be problematic for long sequences.

    - Self-Attention: This type of attention quantifies the importance of different elements within the input sequence itself.

- General Attention: This type of attention manages the interdependence between inputs and outputs, enhancing the decoder's ability to make accurate predictions.

By integrating these components, our model effectively predicts missing types based on the provided predicates and objects, improving data quality and enhancing the usability of Linked Data.

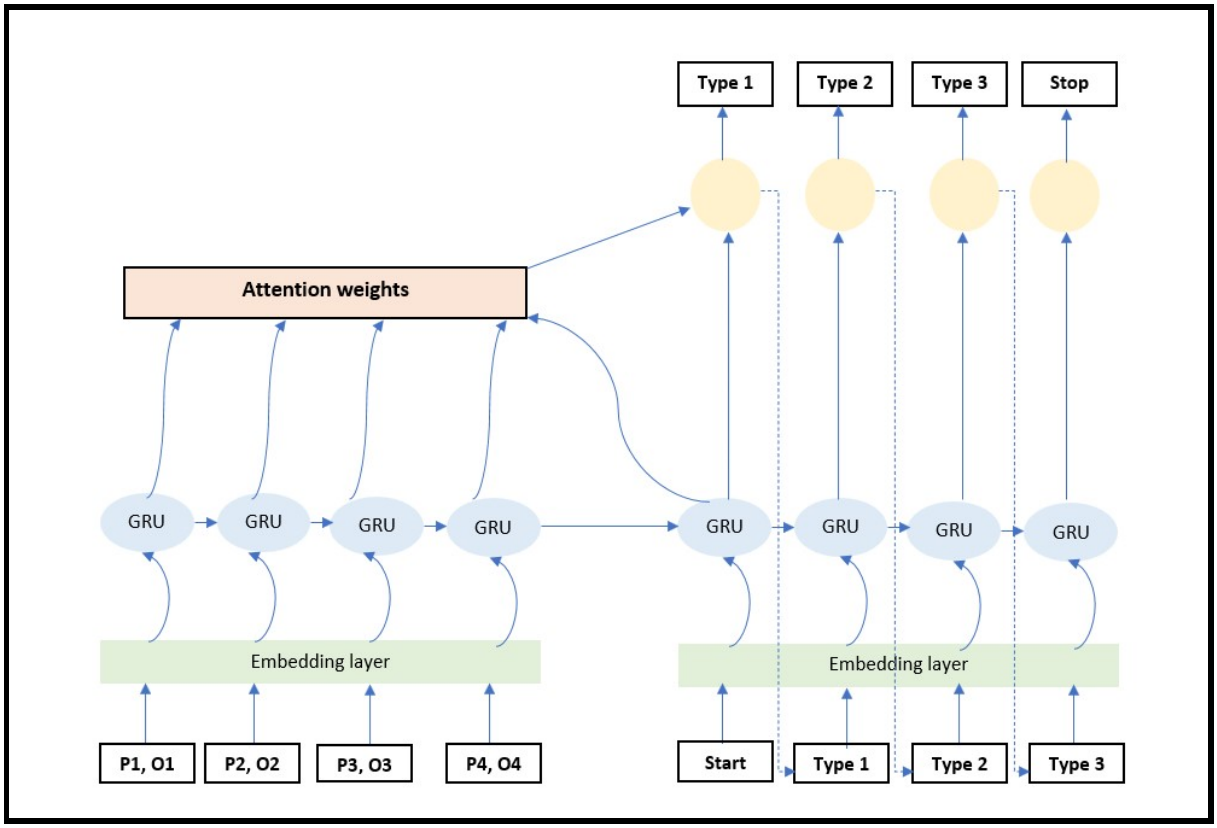


Figure 5.4: Architecture of our encoder-decoder model with attention mechanism [103]

5.4 Experiments

To evaluate the performance of our method, we conducted experiments using the DBpedia dataset.

Due to technical limitations related to RAM capacity, we used 15,292 RDF triples across the various phases of our study. These triples were allocated as follows: 60% for training, 20% for validation, and 20% for testing. Details of this division are provided in Table 5.1.

Our model employs the ADAM optimizer and the 'Sparse Categorical Cross Entropy'

loss function. The training phase encompassed 81 epochs, with further specifics outlined in Table 5.2.

Table 5.1: The number of records for each step [103]

Dataset	DBpedia
Training (60%)	9174
Validation (20%)	3059
Test (20%)	3059
Total (100%)	15292

Table 5.2: Hyper parameters values [103]

Hyper parameter	**Definition**
Optimization function	Function ADAM
Loss function	Sparse Categorical Cross Entropy
Number of GRU Nodes	1024
Batch Size	64
Embedding size	25

In DBpedia, an entity can belong to multiple types through the ‘rdf:type‘ relationship. Our model effectively detects and generates such type links, ensuring accurate classification of entities in Linked Data.

For instance, consider the entity Leonardo da Vinci. In DBpedia, our type detection model generates the following triples:

```
1 <http://dbpedia.org/resource/Leonardo_da_Vinci>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://dbpedia.org/ontology/Person> .
2 <http://dbpedia.org/resource/Leonardo_da_Vinci>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://dbpedia.org/ontology/Artist> .
3 <http://dbpedia.org/resource/Leonardo_da_Vinci>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://dbpedia.org/ontology/Scientist> .
```

These automatically generated links indicate that Leonardo da Vinci is correctly classified as a ‘Person‘, ‘Artist‘, ‘Scientist‘, and ‘Engineer‘. Our model captures these semantic relationships with high accuracy, enriching Linked Data with well-structured entity classifications.

5.5 Results and Discussion

To assess our type prediction method in Linked Data, we use standard evaluation measures: precision, recall, and F-measure. In multi-label classification, these metrics are defined as follows:

These metrics evaluate the performance of the multi-label classification system by comparing the predicted labels with the gold standard labels for each test example. We focus on three key example-based metrics: recall, precision, and F-measure, as detailed in Appendix B.

Table 5.4 presents the results for two scenarios: the encoder-decoder model with Attention Mechanism (AM), which represents our solution, and the encoder-decoder model without the attention mechanism.

The histogram in Figure 5.6 visualizes the evaluation results based on the standard metrics defined in Table 5.4.

During training, our model with the attention mechanism achieved a cost function value of 0.0217, whereas the model without the attention mechanism had a cost function value of 0.0937.

After computing the recall, precision, and F-measure values, we found that our model significantly outperformed the encoder-decoder model without the attention mechanism.

The results presented in Table 5.4 and Figure 5.6 indicate that our model performs well across the evaluation metrics. Specifically, our approach achieves higher values compared to the encoder-decoder model without the attention mechanism, which recorded a precision of 86.92%, recall of 89.00%, and F-measure of 87.95%.

The findings underscore the importance of incorporating the attention mechanism. By assigning weights to inputs based on their relevance, the attention mechanism enhances the accuracy of the output predictions. Additionally, the use of GRU cells has positively impacted the quality of the results by improving the efficiency of feature extraction from inputs.

In contrast, the encoder-decoder model without the attention mechanism shows less impressive results, as it cannot assign importance to the inputs during the type prediction process.

These results could be further improved by conducting the training phase on more powerful hardware and by utilizing larger datasets.

Table 5.3: Results of type predictions with our model and a simple encoder-decoder model [103]

Model	**Precision**	**Recall**	**F-Measure**
Encoder-decoder model with AM	86.92	89.00	87.95
Encoder-decoder model without AM	75.16	79.02	77.04

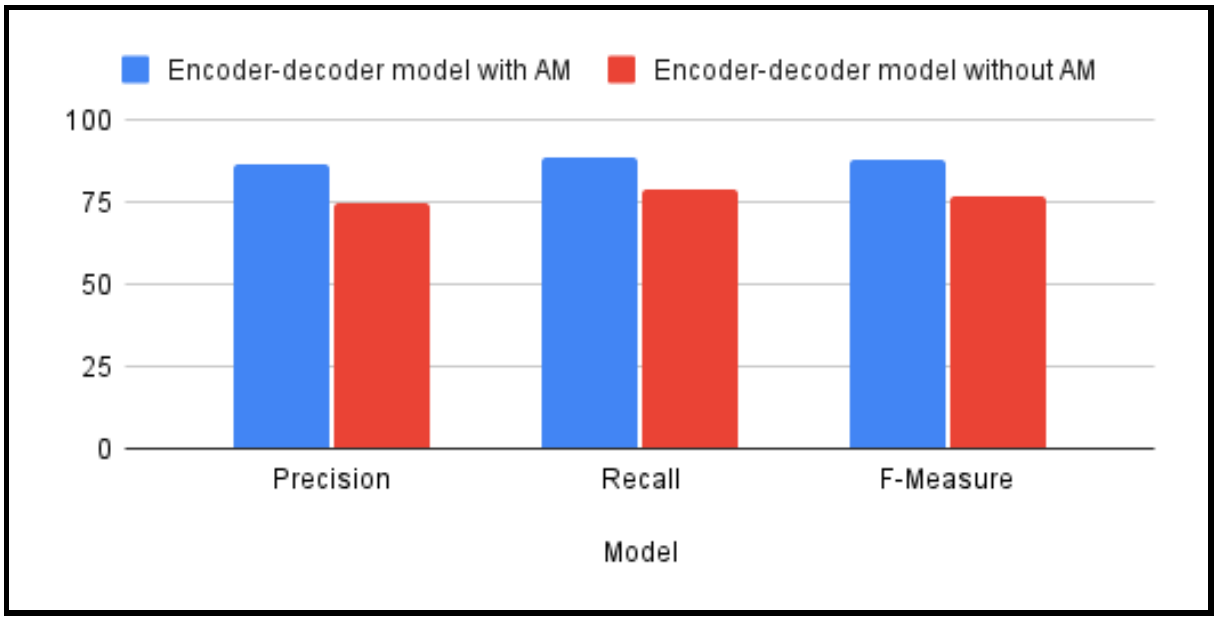


Figure 5.5: Histogram of evaluation results [103]

For further validation, we tested our trained model on the UniProt dataset using a variety of performance metrics: recall, precision, f-measure, MRR, and Hits@K. To compute the Hits@1, Hits@3, and Hits@10 scores, we applied the beam search algorithm, which generates multiple output sequences by exploring the most probable sequence paths. This method ranks potential sequences based on their likelihood, and depending on the beam size, multiple sequences are produced for evaluation.

After running these evaluations, our model achieved recall, precision, and f-measure values of 76.47%, 71.68%, and 73.00%, respectively. Additionally, the MRR, Hits@1, Hits@3, and Hits@10 metrics were 75.53%, 73.98%, 76.81%, and 77.51%. These results are summarized in Table 5.4 and illustrated in Fig. 5.6. The strong performance on the UniProt dataset reinforces the versatility of our model, demonstrating that it maintains high accuracy across different datasets. This evaluation highlights the robustness of our approach in yielding reliable outcomes, even when applied to new datasets.

Table 5.4: Results of type predictions of our model on UniProt dataset [27]

MRR	Hits@1	Hits@3	Hits@10	Precision	Recall	F-Measure
75.53	73.98	76.81	77.51	76.47	71.68	73.00

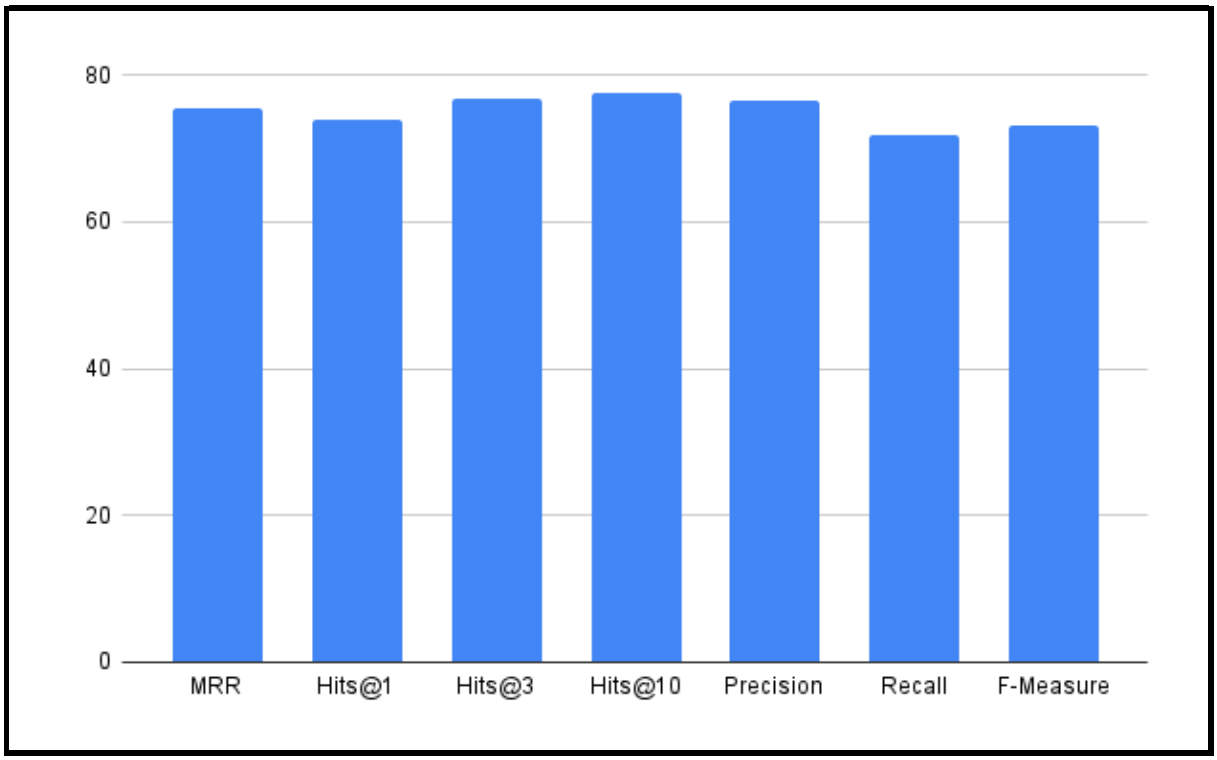


Figure 5.6: Histogram of evaluation results on UniProt [27]

5.6 Conclusion

In recent years, the challenge of detecting missing types in Linked Data has gained significant attention within the semantic web community. Accurate type detection is crucial for addressing uncertainty and enhancing the reliability of Linked Data, as it helps to ensure that entities are correctly classified and semantically related.

This chapter presents an encoder-decoder neural network model enhanced with an attention mechanism specifically designed for detecting missing types in Linked Data. Our approach leverages the attention mechanism to effectively identify and predict the types of entities that are not explicitly annotated. By doing so, it enhances the completeness and accuracy of the data, which is essential for mitigating uncertainty in Linked Data applications.

In this work, we address key limitations in type detection within Linked Data by

proposing a neural network-based multi-label classification model. Unlike existing approaches that suffer from limited analysis of semantic relationships between RDF triple components and rely on manual feature extraction—compromising quality—our model leverages an embedding-based representation to capture deeper semantic connections. We incorporate an attention mechanism, integrating both self-attention and global attention, to enhance relationship extraction between entities. Additionally, we employ a sequence-to-sequence model to effectively handle long sequences, mitigating scalability issues faced by previous methods. Our approach also reduces dependency on external data sources by utilizing RDF2Vec embeddings for numerical representation of resources and predicates within the DBpedia dataset. These enhancements contribute to a more robust, scalable, and automated solution for type detection in Linked Data.

Our evaluation of the proposed model on the DBpedia dataset demonstrates its effectiveness. The model's ability to accurately predict missing types contributes to more reliable and interconnected datasets. This improvement is vital for various applications, such as data integration, knowledge graph enhancement, and decision-making processes, where well-defined types are essential for reducing ambiguity and improving data quality.

Chapter 6

Deep Sequence to Sequence Semantic Embedding with Attention for Entity Linking in Context of Incomplete Linked Data

6.1 Introduction

In recent years, numerous organizations have increasingly embraced the principles of Linked Data and open data in publishing their datasets. Linked Data aims to advance the semantic web, catering to diverse user needs such as providing relevant results and facilitating the use of data in various applications. This approach is integral to many applications, including recommendation systems and knowledge graphs.

Despite its potential, several challenges arise in the publication, integration, and querying of Linked Data. These challenges primarily revolve around the quality of the data and the various forms of uncertainty it encompasses. Uncertainty in Linked Data refers to different types of knowledge imperfections.

To address these issues, we focus on the problem of missing links in Linked Data. Missing links within a Linked Data knowledge graph result in incomplete knowledge about entity relationships, limiting the accuracy of conclusions. The W3C classifies incompleteness as a subtype of uncertainty. When entity links are missing, a form of incompleteness is introduced into the dataset, thus contributing to the overall uncertainty. Addressing missing links is essential to mitigate uncertainty and enhance the reliability of Linked Data for various applications and analyses.

6.2 Overview of the Contribution of Link Detection through Deep Learning Techniques

Missing data in Linked Data knowledge graphs, often a result of the creation methods, significantly hinders applications that depend on this data. For example, information retrieval and content recommendation systems may yield incomplete or inaccurate results, leading users to encounter unsatisfactory outcomes and raising concerns about the reliability of these applications' conclusions.

The task of link discovery in Linked Data involves identifying missing semantic links between resources. This process can occur within a single dataset to reveal hidden relationships, enrich available information, and improve overall coherence and completeness. Additionally, link discovery is used between datasets during data alignment to establish connections between corresponding entities.

The primary limitations of these approaches include focusing on only one type of link, such as the sameAs link or the entity type, and neglecting other types. A significant limitation is the lack of comprehensive solutions for discovering various missing links within a single dataset. No treatment has been proposed to facilitate the link discovery process by extracting the semantic relations between the dataset's triples.

In this chapter, we propose a method for detecting missing links between different resources in RDF datasets. This method aims to reduce dataset incompleteness, thus providing more comprehensive results for users of various RDF dataset-based applications.

In contrast to existing methods limited to specific link types, we propose an advanced deep learning approach using an sequence-to-sequence model. This model can detect various hidden relationships between elements within an RDF dataset, making it suitable for handling large amounts of data. A sequence-to-sequence model is chosen for its ability to process long data sequences, which is important for capturing complex dependencies and patterns in data, often characterized by extensive temporal or sequential information. Further enhanced by an attention mechanism that prioritizes data relevant to link prediction. This mechanism assigns higher weights to such data points, strengthening evidence for potential links between entities in LinkED-S2S. The attention mechanism performs a linear combination of encoder and decoder states. Regarding weight determination in the attention mechanism, the model calculates relevance scores for each element in the sequence. These scores reflect how important each element is to the current focus point.

The following key points highlight the main contribution of our work and the essential components of the proposed solution:

1. LinkED-S2S: We propose an innovative Encoder-Decoder Model augmented with

an Attention Mechanism for Link Discovery in Linked Data. This model stands as a novel contribution to the domain, providing a comprehensive solution for link discovery tasks.

2. Focus on all types of semantic links: Unlike previous works that target specific link types (e.g., sameAs), our proposed method, LinkED-S2S, aims to discover all types of missing links between resources in Linked Data. This comprehensive approach addresses a broader range of potential data incompleteness issues.

3. Link discovery within and between datasets: Existing solutions primarily focus on discovering links between different datasets. Our approach extends beyond this by enabling the discovery of missing links within the resources of a single dataset as well. This allows for a more thorough identification of potential connections within the data.

4. Deep Learning for hidden relation extraction: We leverage deep learning techniques with an encoder-decoder model to uncover the underlying semantic relationships within Linked Data. This approach goes beyond traditional methods that rely solely on surface-level features, potentially leading to a more accurate discovery of missing links.

5. Attention mechanism for importance weighting: Our model incorporates an attention mechanism that assigns greater weight to crucial data points during the link discovery process. This focuses the model on the most relevant information, potentially improving the accuracy and efficiency of link prediction.

6.3 Link Discovery Modeling

Discovering links between resources is a major challenge in Linked Data. To address this challenge, we need to define the framework within which link discovery operates.

Let S, S', and R be three sets where S and S' are two sets of RDF resources, and R represents the set of potential semantic relations between the elements of S and S'. Link discovery involves finding triples $T(S_i, R_k, S'_j)$ that represent hidden links $R_k \in R$ between the resources $S_i \in S$ and $S'_j \in S'$, either within a dataset or across multiple datasets. These relations R_k can be any semantic relation, not limited to the "sameAs" relation.

Link discovery becomes particularly challenging due to the vast amount of data involved in very large datasets and the need to consider multiple types of semantic relationships beyond just "sameAs" links. These factors significantly impact the time efficiency

of proposed approaches, as processing large datasets and evaluating various relationships can be computationally expensive. Furthermore, data heterogeneity, such as inconsistent schema, diverse data formats, and varying quality levels, presents additional challenges for link discovery algorithms.

Detecting new links between resources decreases the dataset's incompleteness and allows for the creation of new knowledge.

In the following section, we propose our approach for discovering links in Linked Data to overcome the challenges mentioned above. Our method leverages deep learning techniques, specifically an encoder-decoder model augmented with an attention mechanism, to identify various hidden relationships within RDF datasets. By using an attention mechanism, the model can focus on the most relevant parts of the input data, allowing it to efficiently handle large-scale datasets. Furthermore, the deep learning framework enables the model to capture complex patterns and dependencies, addressing the high variability and semantic richness in Linked Data. This approach aims to tackle both the scale and complexity of link discovery tasks, ensuring enhanced scalability, precision, and robustness in processing large RDF datasets.

6.4 Architecture of the Model

In this section, we present our solution, LinkED-S2S, designed to discover links between Linked Data resources. Figure 6.1 illustrates a high-level schematic diagram of the entire workflow. Datasets 1 and 2, representing potential sources of missing links, undergo pre-processing steps to prepare the data for link prediction. The pre-processed data is then fed into the Link Prediction stage, where our model operates. This stage is responsible for identifying missing links within each dataset and potentially between entities across both datasets. Finally, the predicted links from both datasets are combined in the Output stage.

To explore the core of our work, we start by reading the DBpedia dataset and extracting the triples before converting them into numeric vector formats. This data is then fed into the Encoder-Decoder neural network for training. During link discovery, we use an attention mechanism to generate a context vector for inputs, assigning high weights to the most significant inputs. To achieve a richer and more meaningful representation of the inputs, we incorporate an embedding layer for both the encoder and decoder. The different steps and components of our solution are detailed below.

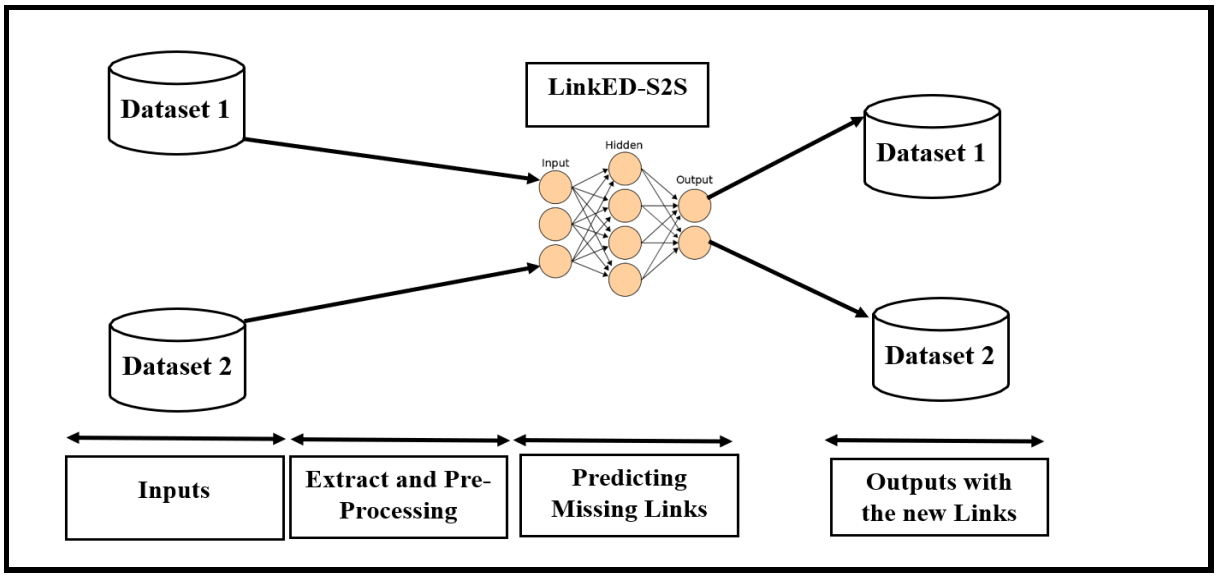


Figure 6.1: Overview of the LinkED-S2S Approach for Missing Link Detection in Linked Data [10]

We begin by establishing the context for our approach within the ontology of uncertainty proposed by the World Wide Web Consortium (W3C), as illustrated in Figure 6.2.

We consider Deep Learning as a model for addressing uncertainty in Linked Data due to its ability to process large amounts of data efficiently [104]. This method also enables the extraction of significant features from the inputs.

Missing links are regarded as a source of incompleteness in Linked Data. Discovering these links allows for the generation of new triples, thereby reducing the dataset's incompleteness and improving its quality.

In this study, we view the link discovery problem as a multi-label classification task. Our proposed model takes two RDF resources, $R1$ and $R2$, as inputs and produces the various possible links L_i between these two resources as outputs. Links L_i are included in L, representing the collection of all potential semantic links $L_i \subseteq L$.

Our approach leverages the powerful capabilities of deep learning, specifically through the use of an encoder-decoder model augmented with an attention mechanism. This combination allows for the identification of multiple types of semantic relationships within and across datasets, addressing both the scale and complexity of link discovery tasks. By improving the completeness and utility of Linked Data, our method aims to provide a more comprehensive and reliable foundation for various Linked Data applications.

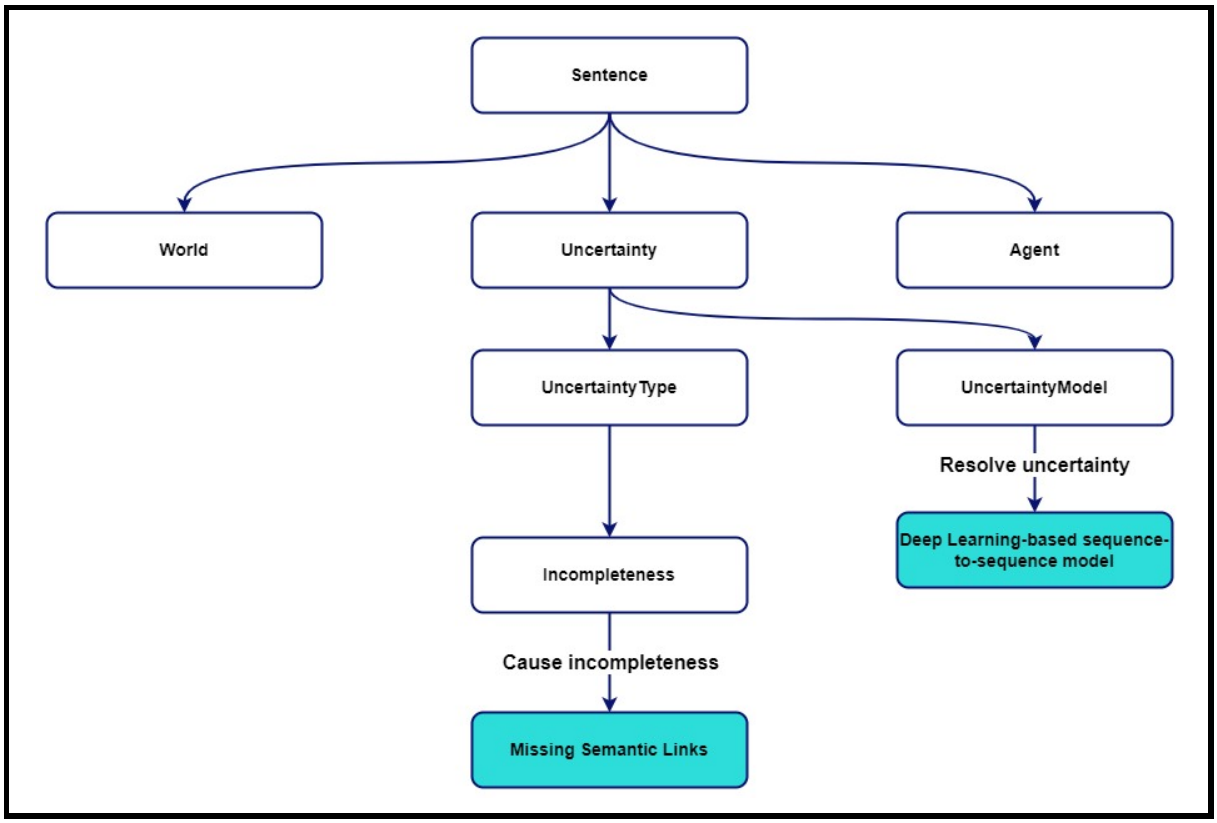


Figure 6.2: Links detection in uncertainty ontology

6.4.1 Pipeline

In this section, we outline the development process of our link discovery solution, as illustrated in Figure 6.3. The first phase focuses on data extraction from the RDF triples dataset, followed by a data pre-processing stage to prepare it for our deep neural network's training. Once the model is trained, we assess its performance to evaluate its efficacy in link discovery.

The initial two phases are explained in the following subsections, while the remaining steps will be discussed later in this chapter.

1. Dataset Creation: To build and train our deep neural network, we opted to use DBpedia, a well-known dataset consisting of RDF triples. Although DBpedia contains around 1.8 billion triples, we utilized a smaller subset due to resource constraints. RDF triples in DBpedia are structured as (subject, predicate, object). After loading the dataset, we move on to the pre-processing step, which is detailed in the following subsection and in Algorithm 6.1.

2. Pre-processing: In this stage, we perform the necessary data preparation steps,

including normalization and converting the data into numerical representations for training and testing our deep neural network. We begin by converting each triple (S_i, P_i, O_i) into a numeric format, where an integer is assigned to each subject, predicate, and object, and subsequently transformed into a One-Hot representation. Next, we construct the input-output pairs for the network's training and testing phases. For instance, consider two subjects: S_1 and S_2. For each subject entity, we extract the corresponding triples to serve as input to the neural network. The network will then predict the semantic links between this subject and other entities as the output. This process is repeated for every pair of subjects S_i, S_j connected by semantic links $L_i \subseteq L$ in the DBpedia dataset.

Algorithm 6.1: Dataset Creation and Pre-processing for Links Detection

Input: DBpedia dataset containing RDF triples (S, P, O)

Output: Numerical representation of dataset for training

`/* Step 1: Dataset Creation */`

Load RDF triples (S, P, O) from DBpedia;

Select a subset of the dataset due to resource constraints;

`/* Step 2: Pre-processing */`

foreach *triple* (S_i, P_i, O_i) *in dataset* **do**

  Assign a unique integer to S_i, P_i, and O_i;

  Convert each integer representation into One-Hot encoding;

Initialize input-output pair construction;

foreach *subject pair* (S_i, S_j) *linked by* $L_i \subseteq L$ **do**

  Extract triples associated with S_i as input;

  Predict semantic links to other entities as output;

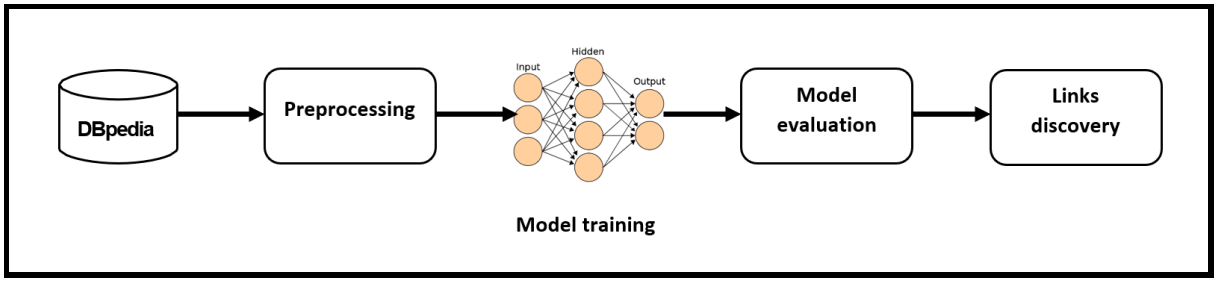


Figure 6.3: The pipeline of our approach [10]

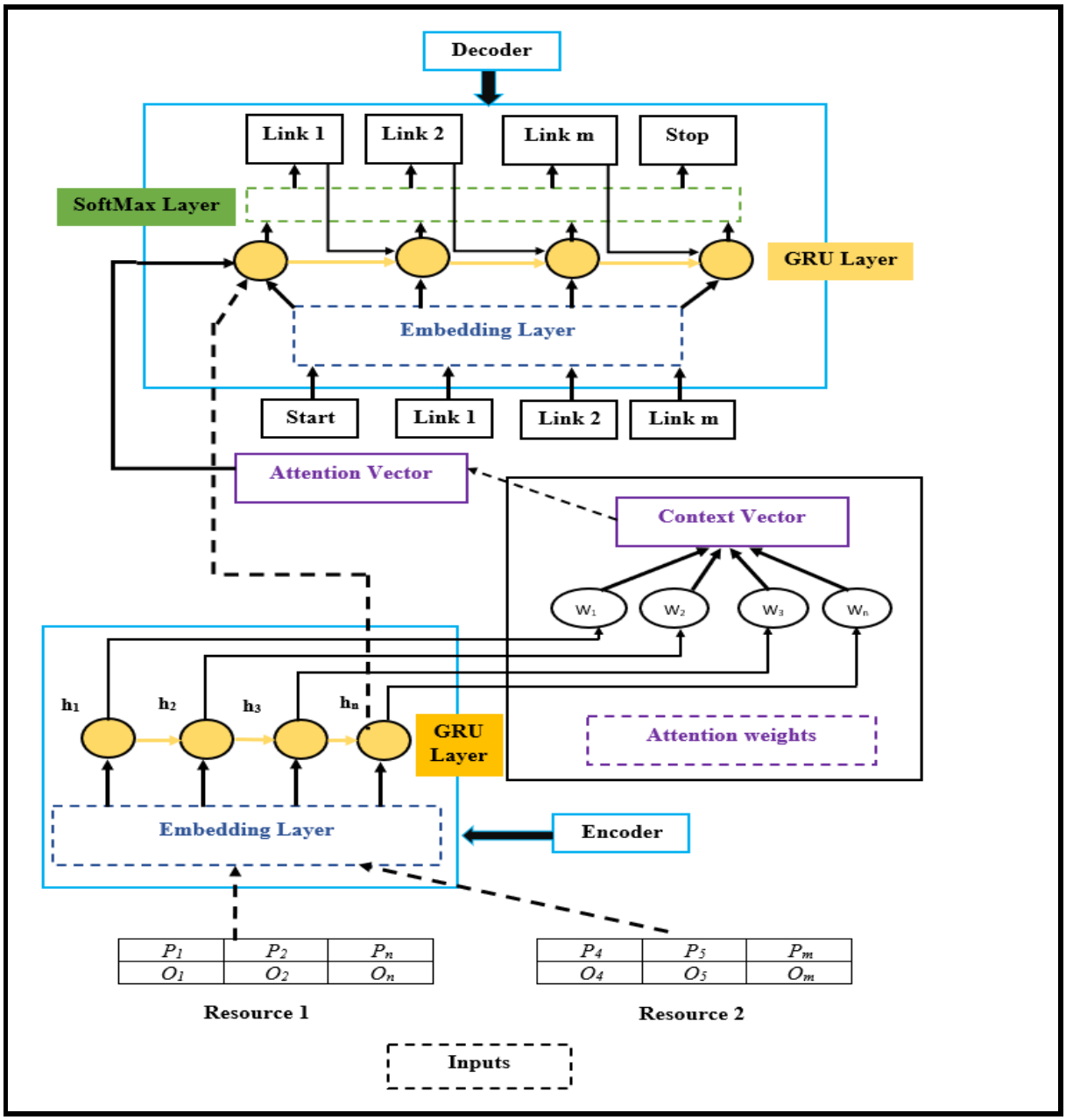


Figure 6.4: Architecture of our Deep Neural Network for link discovery in Linked Data [10]

6.4.2 Our model proposal

In this section, we provide a detailed description of the architecture of our sequence-to-sequence (Seq2seq) deep neural network, LinkED-S2S, designed for link discovery. We will also break down its key components. Our model takes as input pairs of subjects, S_1 and S_2 (input sequence), and outputs a list of potential semantic links L_i that might exist between them.

The LinkED-S2S model is built around two core components: the encoder and the decoder. This structure is selected due to its ability to generate an output sequence (the semantic links L_i) based on an input sequence (triples representing the subjects).

Both the encoder and decoder leverage an embedding layer, which reduces the One-Hot representation by mapping each element to a fixed-size vector.

We employ Gated Recurrent Unit (GRU) cells in both components, as they are effective in capturing relationships between the input from the encoder and the output from the decoder. Additionally, GRUs help to mitigate the vanishing gradient issue often encountered in recurrent neural networks.

To further enhance performance, we integrate an attention mechanism that assigns weights to the most significant inputs, allowing the model to better understand the dependencies between inputs and outputs.

Figure 6.4 visualizes our model's architecture, and the following subsections provide a deeper explanation of each component.

1. Encoder-Decoder: The encoder-decoder framework is ideal for processing lengthy sequences and offers flexibility in handling inputs and outputs of varying sizes. Our encoder structure includes two key elements: an embedding layer and a series of GRU cells (explained in the following subsections). Each input is passed through a GRU cell, which extracts features and sends them to the next GRU cell. The information passed along is stored in the hidden state h_t, computed using the previous hidden state h_{t-1} and the input vector as follows:

$$h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t) \tag{6.1}$$

The final hidden state, often called the encoder vector, captures information about all inputs and is used as the initial hidden state for the decoder.

The decoder also incorporates an embedding layer and GRU cells. At each time step t, the GRU generates a link L_i, using the previous hidden state to compute the current hidden state, which is then passed to the next GRU cell. The hidden states are updated using the following equation, applying a weight to the previous hidden state:

$$h_t = f(W^{(hh)}h_{t-1}) \tag{6.2}$$

The SoftMax layer produces the outputs by generating a probability distribution across possible links, selecting the most likely link based on the highest probability.

The output at time step t is computed using the hidden state and its associated weight, as shown:

$$y_t = SoftMax(W^s h_t) \tag{6.3}$$

2. Embedding Layer: To prepare RDF triples for neural network processing, we encode predicates and objects as one-hot vectors. Given the large storage demands of this representation, we employ an embedding layer to reduce vector dimensions. This layer maps elements to fixed-size vectors with real values, enhancing the network's ability to extract useful features from the data.

3. Gated Recurrent Units (GRUs): GRUs are a type of recurrent neural network architecture suited for handling long sequences of data. They help address the vanishing gradient problem, where the gradient of the loss function diminishes, making training challenging. GRUs mitigate this issue by employing two gates: the update gate and the reset gate, which regulate the flow of information and allow the network to retain or forget information as needed.

   GRUs are computationally efficient, consuming less memory and processing time compared to other RNN architectures, while capturing essential relationships in the data.

4. Attention Mechanism: The attention mechanism enhances the encoder-decoder architecture, particularly when working with long sequences. Without attention, the decoder relies solely on the final hidden state from the encoder, which can lead to errors in generating predictions.

   The attention mechanism addresses this by assigning weights to each hidden state, allowing the decoder to focus on the most relevant inputs. It calculates the importance of each input through self-attention, determining dependencies between the inputs, while the encoder employs global attention to evaluate the significance of elements within each triple. This approach helps the model focus on the most relevant parts of the subject's context when predicting potential links.

   The decoder applies attention to emphasize the most significant inputs. The attention weights for each input are computed as follows:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{6.4}$$

The context vector is obtained by combining the hidden states of the decoder with the attention weights:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{6.5}$$

The score function e_{ij} calculates a scalar value for each key-query pair:

$$e_{ij} = v^T tanh(W[s_{i-1}; h_j]) \tag{6.6}$$

In this equation, i and j represent the indices of the decoder and encoder, respectively. The term α_{ij} refers to the attention weights, h_j is the encoder output sequence being attended to, and h_i is the decoder's hidden state. The context vector c_i encapsulates this information.

The decoder uses the context vector, along with its own output at time step $t-1$, to create an attention vector that predicts the next link. The attention vector is computed as follows:

$$a_t = f(c_i, h_i) = tanh(W_c[c_i; h_i]) \tag{6.7}$$

6.5 Experiments

In this section, we assess the performance of our model for predicting links within Linked Data, specifically using the DBpedia dataset alongside various benchmark datasets.[1]

For training and testing, we utilized Google Colab's Tesla T4 GPU. Due to technical limitations, we worked with 100k RDF triples, which were divided into 60% for training, 20% for validation, and 20% for testing. The input's maximum dimension is 50, and the output's maximum dimension is [2, 205]. The following hyperparameters were used: a batch size of 32, an embedding dimension of 256, and 1024 GRU units in both the encoder and decoder. We employed Sparse Categorical Cross Entropy as the loss function and Adam as the optimizer. A detailed overview of the hyperparameters is provided in Table 6.1.

To evaluate our model, we compared its performance on several benchmark datasets (WN18RR, YAGO3-10, FB15k-237, WN18, and FB15K), as defined in Appendix B,

[1] https://github.com/villmow/datasets_knowledge_embedding

against a range of models cited in [7], as well as the model presented in the same paper. These models, known as embedding models, are capable of handling both lengthy input and output sequences.

We used two key metrics—MRR and Hits@k (also defined in Appendix B)—to compare our model's performance against the baseline models. For the DBpedia dataset, we measured the effectiveness of our model using Recall, Precision, and F-Measure.

Table 6.1: Hyper parameters of our model [10]

Hyper parameter	**Definition**
Optimization function	Function ADAM
Loss function	Sparse Categorical Cross Entropy
Number of GRU Nodes	1024
Batch Size	64
Embedding size	256

In DBpedia, it is common for two entities to be connected by multiple predicates. For example, the entity Barack Obama is linked to the entity United States through several relationships:

- nationality: indicates that Barack Obama has American nationality.
- presidentOf: specifies that he was the president of the United States.
- residence: states that he resides in the United States.

Our model effectively generates such links between entities, capturing semantic relationships in Linked Data. The following DBpedia triples illustrate these connections:

```
1 <http://dbpedia.org/resource/Barack_Obama>
    <http://dbpedia.org/ontology/nationality>
    <http://dbpedia.org/resource/United_States> .
2 <http://dbpedia.org/resource/Barack_Obama>
    <http://dbpedia.org/ontology/presidentOf>
    <http://dbpedia.org/resource/United_States> .
3 <http://dbpedia.org/resource/Barack_Obama>
    <http://dbpedia.org/ontology/residence>
    <http://dbpedia.org/resource/United_States> .
```

These triples indicate that Barack Obama holds American nationality, has served as the president of the United States, and resides in the United States. This example demonstrates how our model successfully identifies and generates meaningful entity links.

6.6 Results and Discussion

In this section, we analyze the evaluation results of our model with and without an attention mechanism, as well as the baseline models on the benchmark datasets defined in Appendix B. All the metrics are presented in Appendix B.

we conduct a comprehensive evaluation of our proposed model in comparison with several baseline models from the literature. The baseline models' results are referenced from the paper [7]. The evaluation was performed on widely-used benchmark datasets, including WN18RR, YAGO3-10, FB15k-237, WN18, and FB15K, which are also presented in the same paper. To assess the performance, we used standard evaluation metrics such as precision, recall, and F-measure, MRR and Hits@k ensuring consistency with the methodology outlined in [7]. By benchmarking our model against these baselines, we aim to highlight the effectiveness of our approach on improving the discovery of hidden relationships within RDF datasets.

Table 6.2 shows the evaluation results of our model on the DBpedia dataset. This table compares the impact of using the attention mechanism when computing the recall, precision, and f-measure metrics on two models: our model with and without the attention mechanism. The good results obtained are justified by the use of the attention mechanism as well as the extraction of semantic relations between triples.

Table 6.2: Evaluation of our model LinkED-S2S and an encoder-decoder without attention mechanism [10]

Encoder-decoder Model	Precision	Recall	F-Measure
With Attention mechanism	86.43	81.22	83.74
Without Attention mechanism	69.45	67.89	68.66

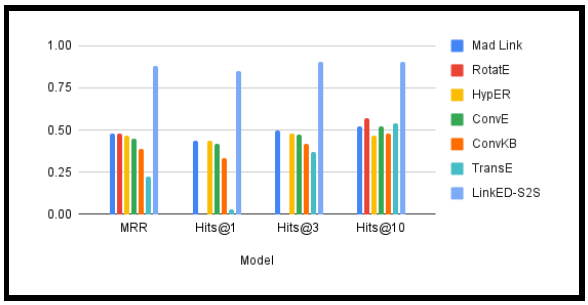
We chose to compare our model with attention to the same model without the mechanism specifically to isolate the impact of the attention mechanism on link prediction. This approach allows us to demonstrate the benefits of the attention mechanism. Additionally, comparing against a baseline variant of our own model is a common practice in the literature to evaluate the effectiveness of specific components within a larger architecture. Vaswani et al. [105] conducted an ablation study by comparing their Transformer model (with a multi-headed attention mechanism) with a simpler RNN baseline.

The comparison of the evaluation results of our model with and without the attention mechanism demonstrates the effectiveness of using the attention mechanism. We chose to compare our model with attention to the same model without the mechanism specifically to isolate the impact of the attention mechanism on link prediction.
This approach allows us to demonstrate the benefits of the attention mechanism. Table 6.2

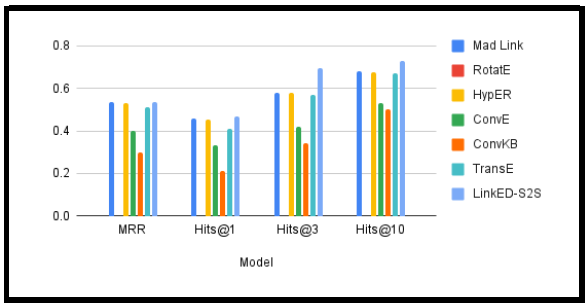
shows that our model outperforms the other model by 16.98%, 13.33%, and 15.08% for the recall, precision, and f-Measure metrics, respectively.

Tables from 6.3 to 6.7 and Figure 6.5 display the evaluation results of our model as well as the baseline models on the benchmark datasets WN18RR, YAGO3-10, FB15k-237, WN18, and FB15K. The results of the other models were obtained from [7].

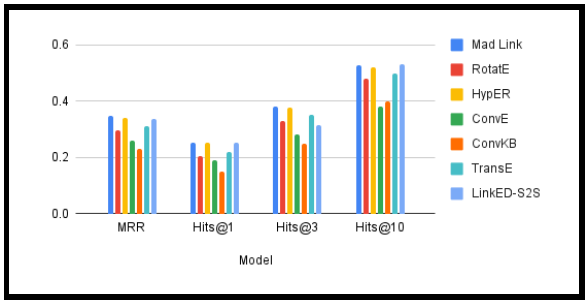
The evaluation results on the WN18RR dataset are shown in Table 6.3 and Figure 6.5a. Table 6.4 and Figure 6.5b show the results for the YAGO3-10 dataset. Table 6.5 and Figure 6.5c illustrate the evaluation outcomes on the FB15k-237 dataset. We also present the results of the WN18 dataset in Table 6.6 and Figure 6.5d. Finally, Table 6.7 and Figure 6.5e show the FB15K dataset results. Each table is followed by a discussion of the findings.



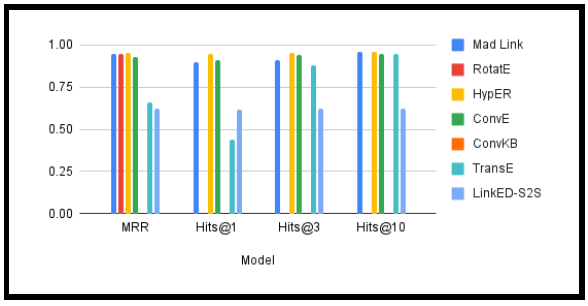
(a) WN18RR



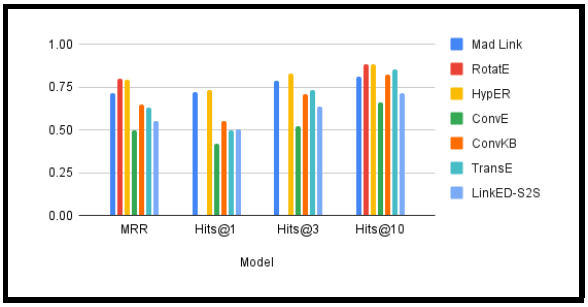
(b) YAGO3-10



(c) FB15k-237



(d) WN18



(e) FB15K

Figure 6.5: Histograms of evaluations of Linked-S2S with baselines on various datasets [10]

Our model clearly outperforms all current state-of-the-art results on the WN18RR dataset for all metrics, with significant improvements of 0.401, 0.413, and 0.408 for MRR metrics, Hits@1, and Hits@3, respectively, compared to the Mad Link model, and an improvement of 0.334 for Hits@10 compared to the RotatE model.

Table 6.3: Evaluation results of our model with baselines on WN18RR dataset [10]

WN18RR				
Model	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
Mad Link [7]	0.477	0.438	0.497	0.523
RotatE [8]	0.476	-	-	0.571
HypER [9]	0.465	0.436	0.477	0.465
ConvE [92]	0.45	0.42	0.47	0.520
ConvKB [91]	0.39	0.33	0.42	0.48
TransE [93]	0.22	0.03	0.37	0.54
LinkED-S2S	**0.878**	**0.851**	**0.905**	**0.905**

Our model outperformed other models on the YAGO3-10 dataset for the Hits@1, Hits@3, and Hits@10 metrics by 0.011, 0.115, and 0.05, respectively. For the MRR metric, the Mad Link model outperforms our model by a small margin of 0.004.

Table 6.4: Evaluation results of our model with baselines on YAGO3-10 dataset [10]

YAGO3-10				
Model	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
Mad Link [7]	**0.538**	0.457	0.580	0.68
RotatE [8]	-	-	-	-
HypER [9]	0.533	0.455	0.580	0.678
ConvE [92]	0.4	0.33	0.42	0.53
ConvKB [91]	0.3	0.21	0.34	0.5
TransE [93]	0.51	0.41	0.57	0.67
LinkED-S2S	0.534	**0.468**	**0.695**	**0.730**

We achieved good results on the FB15k-237 dataset. Our model performed better for Hits@1 and Hits@10, with improvements of 0.002 and 0.004, respectively. Mad Link's results outperformed ours by 0.012 in MRR and 0.067 in Hits@3.

Table 6.5: Evaluation results of our model with baselines on FB15k-237 dataset [10]

FB15k-237				
Model	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
Mad Link [7]	**0.347**	0.252	**0.38**	0.529
RotatE [8]	0.297	0.205	0.328	0.480
HypER [9]	0.341	0.252	0.376	0.520
ConvE [92]	0.26	0.19	0.28	0.38
ConvKB [91]	0.23	0.15	0.25	0.40
TransE [93]	0.31	0.22	0.35	0.5
LinkED-S2S	0.335	**0.254**	0.313	**0.533**

On the WN18 dataset, we noticed that our model performed less well. The nature of these datasets allows us to justify our findings. The test data are, in fact, the inverse relations of the training data. Other models' best results outperform ours by 0.33, 0.328, 0.335, and 0.338 for the metrics MRR, Hits@1, Hits@3, and Hits@10, respectively.

Table 6.6: Evaluation results of our model with baselines on WN18 dataset [10]

WN18				
Model	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
Mad Link [7]	**0.95**	0.898	0.911	**0.96**
RotatE [8]	0.949	-	-	-
HypER [9]	0.951	**0.947**	**0.955**	0.958
ConvE [92]	0.93	0.91	0.94	0.95
ConvKB [91]	-	-	-	-
TransE [93]	0.66	0.44	0.88	0.95
LinkED-S2S	0.62	0.619	0.62	0.622

As shown in Table 6.7 and Figure 6.5e, the HypER model outperforms our model by a factor of 0.238, 0.233, 0.192, and 0.173 for the MRR metrics, Hits@1, Hits@3, and Hits@10, respectively. The FB15K dataset has the same inconvenience as the WN18 dataset. The test part's relations are the inverse of the training part's. Therefore, our model works less efficiently on this type of dataset. The RotatE and HypER models are the most appropriate for these two datasets.

Table 6.7: Evaluation results of our model with baselines on FB15K dataset [10]

FB15K				
Model	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
Mad Link [7]	0.712	0.722	0.788	0.81
RotatE [8]	0.797	-	-	0.884
HypER [9]	**0.790**	**0.734**	**0.829**	**0.885**
ConvE [92]	0.5	0.42	0.52	0.66
ConvKB [91]	0.65	0.55	0.71	0.82
TransE [93]	0.63	0.5	0.73	0.85
LinkED-S2S	0.552	0.501	0.637	0.712

The outcomes depicted above were obtained by representing entities as a set of triples, as well as using the attention mechanism to extract the most important triples during link generation. In what follows, we will discuss the implications of employing these two concepts.

The attention mechanism guides the reasoning or link detection process by attending to a subset of triples considered the most valuable. It can form the output vector by establishing more direct and symmetric relationships between each input vector. Additionally, the mechanism allows for extracting relations between triples within the same input vector (self-attention). Employing this mechanism, we achieved excellent results comparable to those of the Mad Link model, which also utilizes an attention mechanism. The key difference between our model and Mad Link lies in the method of data representation used for training data. This distinction will be explained in detail in the following section.

By representing entities using their associated triples, our model can comprehend and extract the various relations between the entity and other elements within the dataset. This rich representation enables our model to consider not only the data structure (through the triples' order) but also the semantic meaning of the relationships (represented by the predicates in the triples).

The evaluation results on the FB15K and WN18 datasets indicated that our model does not perform well on datasets with a set of relations in the training part and their inverse relations in the test part. Our model only considers links in one direction.

6.7 Conclusion

Detecting links in Linked Data has become crucial for ensuring data completeness as its use has expanded. In this chapter, we introduced an encoder-decoder model enhanced with an attention mechanism to tackle the challenge of missing links in Linked Data.

In this work, we propose LinkED-S2S, an Encoder-Decoder model enhanced with an attention mechanism to address key challenges in link discovery within Linked Data. Unlike existing methods that focus on detecting only a single type of link (e.g., sameAs), our approach aims to uncover all types of missing links, providing a more comprehensive solution to data incompleteness. Additionally, while most techniques concentrate on linking entities between different datasets, LinkED-S2S extends link discovery to both inter-dataset and intra-dataset connections, ensuring a more thorough identification of hidden relationships.

To overcome common limitations such as incorrect predictions and overlooked hidden relations, our model leverages deep learning to extract semantic relationships beyond surface-level features. The attention mechanism further refines link prediction by assigning greater weight to crucial data points, addressing the data weighting problem found in previous approaches. Unlike text-dependent methods that struggle with minimal data, our approach minimizes reliance on textual descriptions, making it more adaptable to diverse datasets. These enhancements contribute to higher accuracy, better dataset alignment, and improved hidden relation extraction, pushing the boundaries of link discovery in Linked Data.

Our results show that this attention-based encoder-decoder model effectively predicts missing links, significantly enhancing the accuracy and completeness of knowledge graphs. This improvement can lead to more precise recommendations in recommendation systems and more effective reasoning in question-answering tasks. We believe this approach has broad applicability across various domains that depend on comprehensive and well-connected Linked Data.

The primary strength of our model lies in its ability to extract semantic relationships between subjects and objects through advanced data representation techniques. By leveraging the attention mechanism, the model effectively captures relations between input elements and between inputs and outputs. The evaluation results on the DBpedia and benchmark datasets evidence the effectiveness of our approach.

Chapter 7

Linked Data Interlinking with Siamese Neural Networks

7.1 Introduction

In recent years, the proliferation of Linked Data has significantly expanded the landscape of interconnected datasets, facilitating more seamless integration and sharing of information across various domains [12]. Nevertheless, ensuring the precision and quality of the links that bridge these datasets remains a substantial challenge. Of particular importance is the detection of sameAs links, which denote that two resources are equivalent and can be used interchangeably. Accurate identification and validation of these links are crucial for enhancing the connectivity and interoperability of Linked Data [3].

Traditional methods for detecting sameAs links have primarily relied on rule-based or statistical approaches. However, these techniques often struggle with the complexity and scale of extensive datasets. Conventional methods for linking RDF datasets have notable limitations. For instance, sameAs links typically presuppose strict equivalence between resources, which may not always hold, leading to possible inaccuracies. Moreover, the automatic discovery of sameAs links depends heavily on the efficacy of the algorithms employed, which may encounter scalability and accuracy issues. Ontology matching techniques also face difficulties in managing heterogeneity, adapting to evolving ontologies, and achieving high scalability. While automated link invalidation methods show promise, they may require further validation to confirm their effectiveness across various domains. These challenges underscore the need for more robust and scalable linking algorithms for RDF datasets.

The remainder of this chapter is organized as follows: we begin by presenting the contribution overview for the sameAs Link detection task through deep learning, followed

by the formulation of the research problem. Next, we introduce our proposed approach for detecting sameAs links. Then, we describe the experimental setup and evaluation results, comparing our model's performance with baseline models. Finally, we conclude the chapter.

7.2 Overview of the Contribution of SameAs Link Detection through Deep Learning Techniques

In parallel, deep learning models have emerged as transformative tools across natural language processing, computer vision, and data mining [57]. These models leverage the sophisticated capabilities of deep neural networks to capture complex patterns and relationships within data, thereby enabling more accurate and efficient detection of sameAs links [103].

In this chapter, we introduce a novel methodology for detecting sameAs links in Linked Data, utilizing deep learning Siamese neural networks. Siamese networks, characterized by twin subnetworks that share weights, have demonstrated remarkable success in learning similarity measures for various tasks [106]. By harnessing the power of Siamese networks, we aim to leverage the inherent similarities and dissimilarities between resources to identify genuine sameAs links amidst noisy and erroneous data.

Our primary objective is to develop a robust and effective model for automatically detecting same-as links in Linked Data. We address the challenges posed by the inherent incompleteness and heterogeneity of Linked Data, seeking to enhance the accuracy and efficiency of same-as link detection. Additionally, we explore the potential of deep learning techniques to overcome the limitations of traditional approaches and advance the state-of-the-art in this domain.

The principal contributions of our work can be summarized as follows:

- Enhanced Link Discovery Accuracy: Siamese neural networks improve the accuracy of sameAs link detection by capturing intricate patterns and dependencies in linked data, leading to more reliable link predictions.
- Automatic Feature Learning: Siamese networks can automatically learn relevant features from the data, eliminating the need for manual feature engineering and allowing the model to adapt to different types of linked data.
- Contextual Link Detection: By incorporating contextual information, such as entity attributes and relationships, Siamese networks enhance the model's ability to distinguish between valid and erroneous links.

- Scalability and Efficiency: Deep learning techniques, including Siamese networks, efficiently handle large-scale linked data, making them suitable for real-world applications with extensive datasets.
- End-to-end Learning: Siamese networks support end-to-end learning, allowing the model to learn the optimal link detection function directly from the data, reducing reliance on manual rule-based approaches.

These contributions highlight the potential of deep learning Siamese neural networks to enhance the accuracy, efficiency, and contextual awareness of same-as link detection in Linked Data.

7.3 SameAs Link Detection Modeling

Given a set of data sources $D = d_1, d_2, ..., d_n$, where each data source d_i contains a set of entities $E_i = e_1, e_2, ..., e_m$, and each entity e_j has a unique Uniform Resource Identifier (URI) u_j. The task of sameAs links detection is to find pairs of entities (e_a, e_b), where e_a is in E_i and e_b is in E_j. The task is to determine whether entities e_a and e_b truly represent the same entity or not.

The main aim of the work is to determine a similarity score for $s(e_a, e_b)$ to indicate how likely e_a and e_b are the same entity.

We define a set of pairs of entities, $P = (e_a, e_b)$, where $e_a \in E_i$ and $e_b \in E_j$. The target is to assign a label $y_p \in 0, 1$ to each pair $p \in P$ such that e represents positive sameAs link and $y_p = 0$ represents negative sameAs link. The task is to find a function $f(p)$ that maps the entity pair p to the label y_p.

In the next section, we will introduce our novel approach, a "siamese neural network for detecting same-as links in Linked Data."

Figure 7.1 illustrates the detection of sameAs links in the context of the W3C Uncertainty Ontology.

7.4 Architecture of the Model

We're using Siamese neural networks to solve the problem of detecting SameAs links in linked data. Here's the process: We start by creating a dataset with pairs of entities, marking them as either the same or different. For this, we use information from DBpedia and YAGO[1]. We then format this data so that our model can use it effectively.

[1] https://yago-knowledge.org/downloads/yago-4

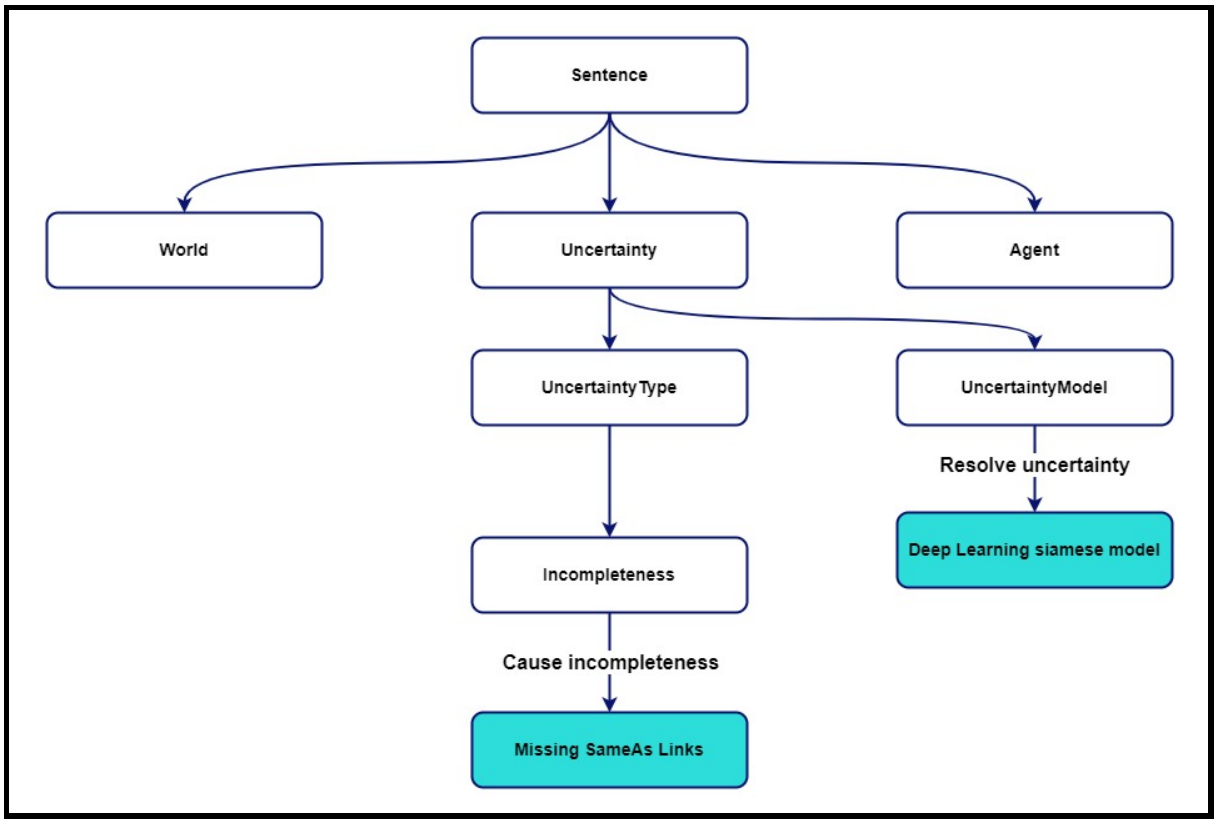


Figure 7.1: SameAs Links Detection in W3C Ontology

The Siamese network learns by comparing these pairs. It's trained to bring pairs that are the same closer together and push apart those that are different. This way, when it sees new data, it's better at deciding if two entities are the same.

Our approach is effective because it handles complex relationships well and is efficient, thanks to shared weights. It also doesn't need manual feature engineering, which makes it flexible for various data types.

In the next sections, we'll describe in detail how our model works and the steps we took to develop it.

In our approach to finding SameAs links, we use Siamese neural networks to make it straightforward. We want the model to determine if two entities are the same thing or different. It's all about figuring out if they should be considered the same in the real world.

We start by working with pairs of entities, represented as $(x1, x2)$, where each x_i is a numerical version of the entity. Our Siamese network learns to give a similarity score for these pairs using a function $f(x1, x2)$. By setting a threshold on this score, we determine whether the pair is a SameAs link or not.

We focus on tuning our network to distinguish between SameAs links and non-links.

This approach helps us use deep learning to deal with the complexities of linked data and improve our detection of SameAs links. Our goal is to boost the quality and completeness of linked data by accurately identifying these links.

7.4.1 Pipeline

The approach to detecting sameAs links in linked data with a Siamese neural network involves a sequence of well-defined stages. The process is detailed in Algorithm 7.1 starts with data preparation, which entails gathering relevant datasets that include linked data sources with their entities and their relationships. A specific dataset is then constructed for sameAs link detection, comprising pairs of entities along with labels indicating whether they are identical (sameAs) or distinct (non-sameAs).

After assembling the dataset, it undergoes a pre-processing phase. This step involves applying normalization and encoding techniques to prepare the data for the Siamese neural network, ensuring it is in a format that the network can effectively utilize.

With the data pre-processed, the Siamese neural network is trained on the dataset. During training, the network learns to derive meaningful representations of the entities and assess their similarities. The training involves using optimization algorithms and loss functions to adjust the network's weights, aiming to reduce the training loss and enhance the model's accuracy.

Following the training phase, the model's performance is evaluated using a separate evaluation dataset. Metrics such as accuracy, precision, recall, and F1-score are computed to determine how well the model detects sameAs links in the linked data.

This deep learning pipeline leverages the Siamese neural network's capabilities to identify sameAs links, utilizing large-scale linked data resources effectively. The detailed process is illustrated in Figure 7.2.

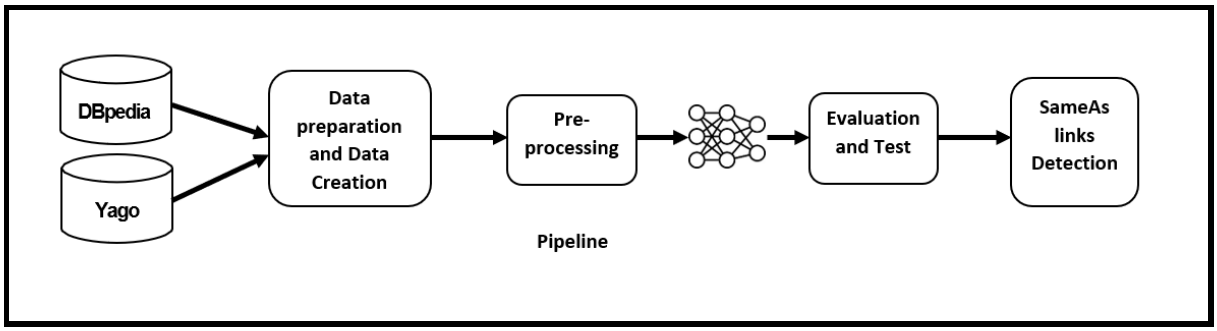


Figure 7.2: Pipeline of our sameAs Links Detection Approach

Algorithm 7.1: Dataset Creation and Pre-processing for SameAs Links Detection

Input: A reference dataset containing entities from DBpedia and YAGO with sameAs links

Output: Numerical representations of entity pairs for training and testing

Step 1: Data Preparation

foreach *entity* E_i *in the dataset* **do**
    Extract relevant objects O_i associated with E_i;
    Remove duplicate or inconsistent entities from the dataset;
end

Ensure only unique and consistent entities remain;

Step 2: Dataset Creation

Initialize an empty dataset D;

foreach *pair of entities* (E_i, E_j) *in the reference dataset* **do**
    if (E_i, E_j) *has a sameAs link* **then**
        Label (E_i, E_j) as "sameAs" (1);
    else
        Label (E_i, E_j) as "different" (0);
    end
    Add (E_i, E_j) to dataset D;
end

Ensure a balanced distribution of "sameAs" and "different" pairs;

Step 3: Pre-processing

foreach *triple* (S_i, P_i, O_i) *in the dataset* **do**
    Convert objects O_i into numerical format;
    Transform numerical values into a one-hot representation;
end

foreach *pair of subjects* (S_1, S_2) **do**
    Extract object values associated with each subject;
    Assign binary output: 1 for "sameAs", 0 for "different";
end

Return the processed dataset for model training;

7.4.1.1 Data Preparation

In our proposed method, we start by preparing the datasets to detect sameAs links between entities within these sets. This preparation requires meticulous curation and orga-

nization of the datasets to enable effective comparison. We specifically extract relevant attributes for each entity, focusing on their associated objects exclusively as features. These objects, encapsulating key information about the entities, form the core basis for identifying sameAs links. By using objects as features, we ensure that the comparison relies on the most pertinent and distinguishing characteristics of the entities.

To maintain the reliability and precision of our analysis, we carefully eliminate any duplicate or inconsistent entities from the datasets. Removing duplicates helps to avoid redundant information that might skew the results. Additionally, excluding inconsistent entities ensures that our comparisons are conducted only on valid and dependable data, thus bolstering the integrity of the analysis. This crucial step ensures that we are comparing unique and consistent entities, which enhances the quality and accuracy of the sameAs link detection process.

7.4.1.2 Dataset Creation

Next, we will create a dataset of entity pairs for comparison. To achieve this, we'll use a reference dataset containing entities from DBpedia and YAGO, along with their corresponding sameAs links. From this reference dataset, we will randomly sample pairs of entities, ensuring a balanced mix of sameAs and different pairs. Each pair will be labeled as either "sameAs" or "different," indicating whether they represent the same real-world entity or not. Ensuring a balanced distribution of sameAs links in the dataset is essential to avoid any biases in the training of our Siamese neural network, thereby promoting fair and accurate classification.

By leveraging the DBpedia, YAGO, and reference datasets, we can train our model to recognize the patterns and relationships necessary for accurate same-as link detection in linked data. This approach will enable our model to learn the distinguishing features that separate identical entities from different ones, improving the overall performance of the same-as link detection process.

7.4.1.3 Pre-processing

When detecting SameAs links in linked data using deep learning Siamese neural networks, an essential step is data pre-processing. This step focuses on normalizing the data and representing it numerically, specifically using objects as inputs and assigning binary values of 0 or 1 as outputs, where 0 indicates "different" and 1 indicates "sameAs."

The first task is to convert each triple (S_i, P_i, O_i) into a numerical format. This involves assigning unique integer values to the objects and transforming them into a one-hot representation. Following this, we construct the inputs for our neural network.

Consider a pair of subjects, S_1 and S_2. Our goal is to gather the object values related to each subject, which will serve as inputs to our neural network. These objects are converted into numerical representations, making them usable in the network's training and testing phases. The desired outputs are assigned binary values of 0 or 1.

For example, let's consider two subjects, S_1 and S_2, connected by a SameAs link. Our network's inputs will consist of the object values associated with each subject, and the output will be 1, indicating a "sameAs" link.

By adopting this data pre-processing method, we ensure that the inputs are represented numerically, focusing exclusively on the objects, while the outputs are binary values that indicate whether a SameAs link exists between the subjects.

7.4.2 Our Siamese Neural Network

To detect SameAs links, we use a Siamese neural network architecture designed to compare pairs of entities and output a similarity score ranging from 0 to 1, with 0 indicating no similarity and 1 indicating high similarity. This Siamese network employs two identical LSTM neural networks sharing the same weights. These networks are trained to generate similar outputs for similar input pairs and dissimilar outputs for dissimilar pairs.

Figure 7.3 shows the architecture of our proposed model, with subsequent subsections detailing each component.

Our Siamese neural network includes the following layers:

1. Input Layer: This layer accepts the two entity pairs as inputs and forwards them to the next layer.

2. Embedding Layer: This layer transforms the inputs into dense vector representations that capture the entities' semantic meanings.

3. Siamese Layer: Comprising two identical LSTM subnetworks with shared weights, this layer processes the entity embeddings through multiple fully connected layers to produce a similarity score between 0 and 1.

4. LSTM Components: These form a critical part of our deep learning architecture. LSTM (Long Short-Term Memory) networks are a type of recurrent neural network that effectively captures sequential dependencies in the input data. Each LSTM component processes a sequence of input tokens representing an entity's features, retaining and updating information over multiple timesteps. This allows the LSTM components to capture long-range dependencies and extract meaningful representations from the input data. By incorporating LSTM components, the model learns

and encodes complex temporal relationships in the linked data, enhancing its ability to discern entity similarities and accurately predict SameAs links.

5. Output Layer: This layer takes the similarity scores from the Siamese layer and outputs a binary classification (sameAs or different) based on a threshold value.

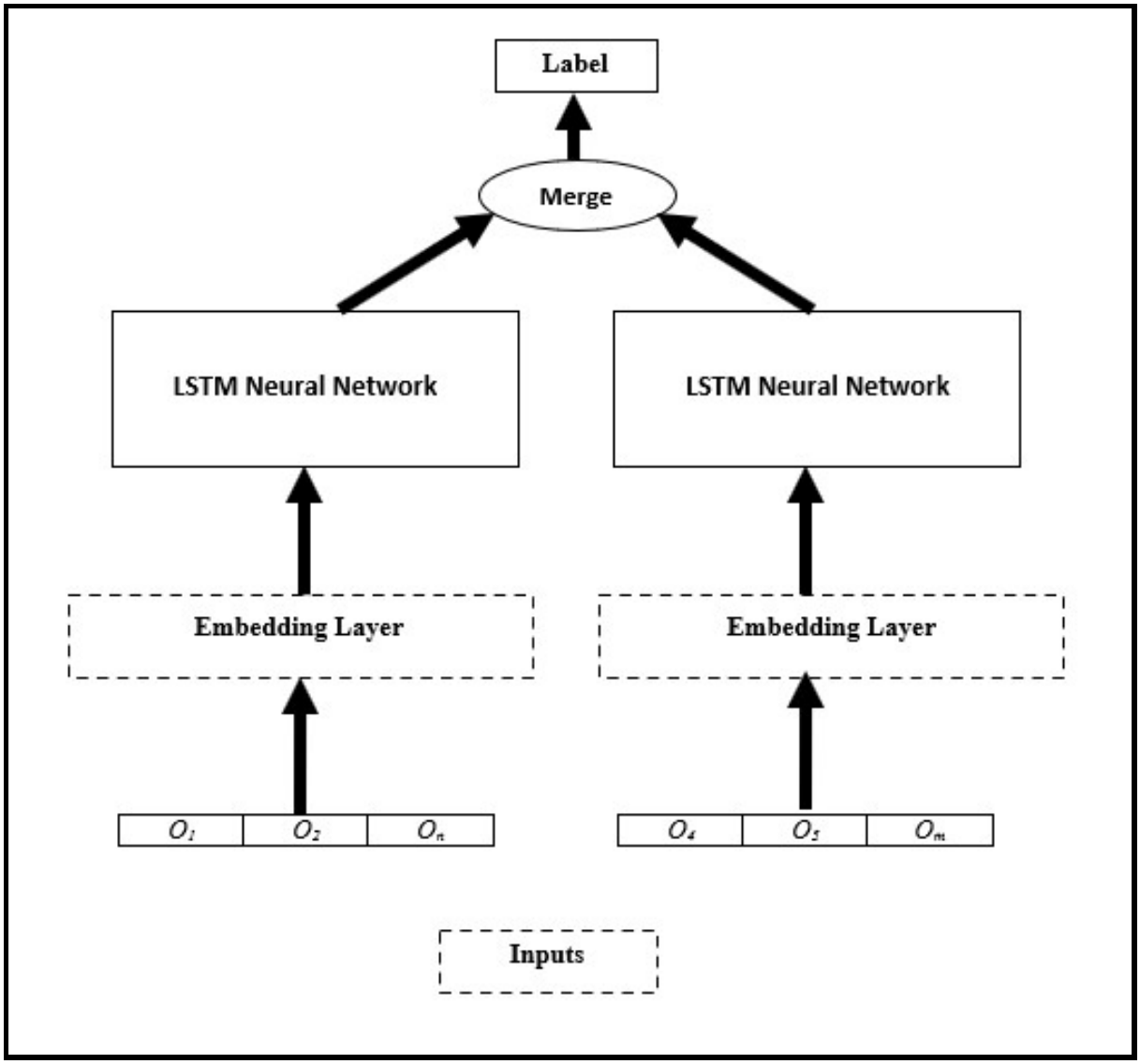


Figure 7.3: Architecture of our Siamese Neural Network for SameAs Links Detection in Linked Data

Let's denote a collection of data sources in Linked Data as $D = \{d_1, d_2, ..., d_n\}$. Each data source d_i contains a set of entities $E_i = \{e_1, e_2, ..., e_m\}$, where e_j represents an individual entity.

For each entity e_j in a data source d_i, we represent it as a sequence of tokens using an LSTM-based encoding mechanism. Let $x_j = (x_1, x_2, ..., x_k)$ be the token sequence representation of entity e_j, where x_k is the k-th token in the sequence.

To compare the similarity between two entities e_a and e_b, we use the Siamese neural network with LSTM components. The network consists of twin LSTM networks with shared weights. Each LSTM component processes a sequence of tokens to produce a fixed-length vector representation, known as entity embedding.

Let $f(x_j)$ denote the LSTM component's output, representing the entity embedding for entity e_j. The Siamese network computes the embeddings for entities e_a and e_b using their respective LSTM components, resulting in embeddings $f(x_a)$ and $f(x_b)$.

To measure similarity, we use a distance metric such as Euclidean distance or cosine similarity. Let $sim(e_a, e_b)$ represent the similarity score between entities e_a and e_b, computed as a function of their embeddings:

$$sim(e_a, e_b) = \text{distance}(f(x_a), f(x_b))$$

To train the Siamese network, we need a labeled dataset of entity pairs, where each pair (e_a, e_b) is annotated as either a positive sameAs link or a negative non-sameAs link. Let $Y = \{y_1, y_2, ..., y_t\}$ be the set of labels for the training data, with $y_i = 1$ indicating a positive sameAs link, and $y_i = 0$ indicating a negative non-sameAs link.

We use a contrastive loss function to train the Siamese network, encouraging the embeddings of positive pairs to be close in the embedding space while pushing the embeddings of negative pairs apart. The contrastive loss is defined as:

$$L = \sum [y_i \cdot \text{distance}(f(x_a^i), f(x_b^i))^2 + (1 - y_i) \cdot \max(margin - \text{distance}(f(x_a^i), f(x_b^i)), 0)^2] \tag{7.1}$$

Here, (x_a^i, x_b^i) represents the token sequences of the i-th entity pair, and the margin is a hyperparameter controlling the minimum distance between embeddings for negative pairs.

During training, the Siamese network learns to minimize the contrastive loss, resulting in embeddings that effectively discriminate between sameAs and non-sameAs link pairs.

By optimizing the Siamese neural network with LSTM components and contrastive loss, our approach aims to accurately detect SameAs links in Linked Data by capturing the semantic similarities between entities.

Table 7.1: Hyper Parameters of our SameAs Links Detection Model

Hyper parameter	**Definition**
Optimization function	Function ADAM
Loss function	Binary Cross Entropy
Number of LSTM Nodes	512
Batch Size	32
Embedding size	256

7.5 Experiments and Evaluation

In this section, we assess the performance of our deep learning Siamese neural network model for detecting SameAs links in Linked Data. We evaluate its effectiveness using the DBpedia and YAGO datasets and compare it against the state-of-the-art PARIS model [79], which has been tested on the same datasets.

We employ various metrics to measure our model's accuracy and precision. These results offer insights into the model's strengths and limitations and highlight its potential applications in linked data analysis. Our evaluation contributes to the advancement of SameAs link detection using deep learning approaches, providing a benchmark for future research in this area.

7.5.1 Experiments

We conducted the training and testing processes using the Tesla T4 GPU available on Google Collaboratory. The dataset was divided into three parts: 60% for training, 20% for validation, and 20% for testing. Each element of the input pair was limited to a maximum dimension of 100, with the output being a binary value of either 0 or 1.

Specific hyperparameters were employed during training to optimize performance. We used a batch size of 32, an embedding dimension of 256, and 512 LSTM units for both the encoder and decoder. The binary cross-entropy function was utilized as the loss function, and the Adam optimizer was employed for model optimization.

For a detailed overview of the hyperparameters used, please refer to Table 7.1.

Our model is designed to determine whether two entities from DBpedia and YAGO refer to the same concept by predicting 'owl:sameAs' links. Below, we present one correct (true) and one incorrect (false) example generated by our model.

For a correct match, our model detects that Albert Einstein in DBpedia and YAGO refer to the same individual:

```
1 <http://dbpedia.org/resource/Albert_Einstein>
    <http://www.w3.org/2002/07/owl#sameAs>
```

```
    <http://yago-knowledge.org/resource/Albert_Einstein> .
```

This link is valid as both URIs correspond to the famous physicist Albert Einstein.

Conversely, our model correctly identifies a false match between Paris (the city) in DBpedia and Paris (the Greek mythological figure) in YAGO:

```
<http://dbpedia.org/resource/Paris>
    <http://www.w3.org/2002/07/owl#sameAs>
    <http://yago-knowledge.org/resource/Paris_(mythology)> . (False
    Prediction)
```

Here, the entities are not the same: one refers to the capital of France, while the other represents a character from Greek mythology. Our model successfully distinguishes such cases, improving the accuracy of cross-ontology entity alignment.

7.5.2 Evaluation

In this section, we evaluate the performance of our proposed Siamese Neural Network Model (SASNN) for identifying sameAs links within Linked Data. The results demonstrate significant improvements over the state-of-the-art PARIS approach. We then discuss and analyze these findings, exploring the factors contributing to the SASNN's effectiveness.

7.5.2.1 Results

The evaluation of our proposed Siamese neural network model (SASNN) for sameAs link detection in Linked Data reveals substantial improvements over the state-of-the-art PARIS approach. As shown in Table 7.2 and Figure 7.4, our model achieves a precision of 98.31%, a recall of 93.54%, and an F-score of 95.87%. In comparison, the PARIS model achieves a precision of 90%, a recall of 73%, and an F-score of 81%. These results underscore the superior performance of our approach in accurately identifying sameAs links.

The training loss and accuracy curves for the SASNN model, depicted in Figure 7.5, illustrate the model's learning process over 200 epochs. The training accuracy rapidly increases and stabilizes around 95%, while the training loss consistently decreases, converging to a low value. This indicates the model's effectiveness in learning from the data and its ability to generalize well to new instances.

Table 7.2: Performance Evaluation of SASNN against the State-of-the-Art Model PARIS

Model	**Precision**	**Recall**	**F-score**
Our Model (SASNN)	**98.31**	**93.54**	**95.87**
PARIS [79]	90	73	81

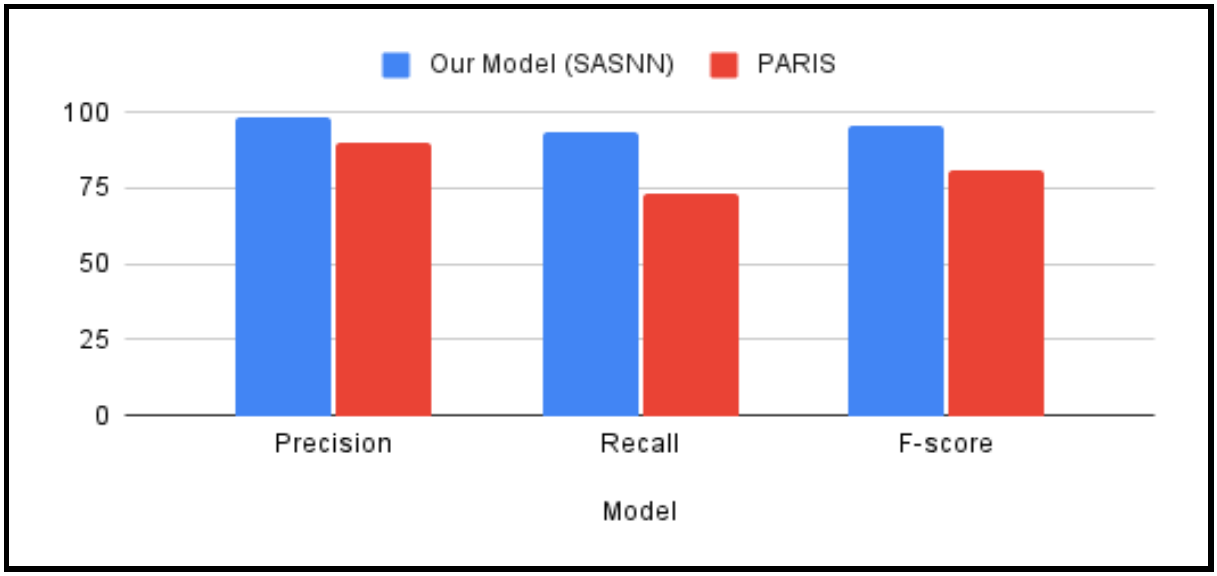


Figure 7.4: Histogram of Performance Evaluation of SASNN against the State-of-the-Art Model PARIS

7.5.2.2 Discussion

The results from our experiments clearly demonstrate the effectiveness of the SASNN model in the task of sameAs link detection. The significant improvement in precision, recall, and F-score over the PARIS model suggests that our approach can better handle the complexities and heterogeneity inherent in Linked Data. The high precision of 98.31% indicates that our model is highly accurate in predicting sameAs links, reducing the likelihood of erroneous links. Additionally, the high recall of 93.54% shows that our model is capable of identifying most of the true sameAs links, ensuring comprehensive coverage.

We interpret these results by considering the following points:

- Superior Accuracy: Our SASNN model achieves higher precision and recall than the PARIS approach, indicating superior accuracy in detecting sameAs links. This improvement highlights the effectiveness of deep learning in capturing complex patterns in linked data.

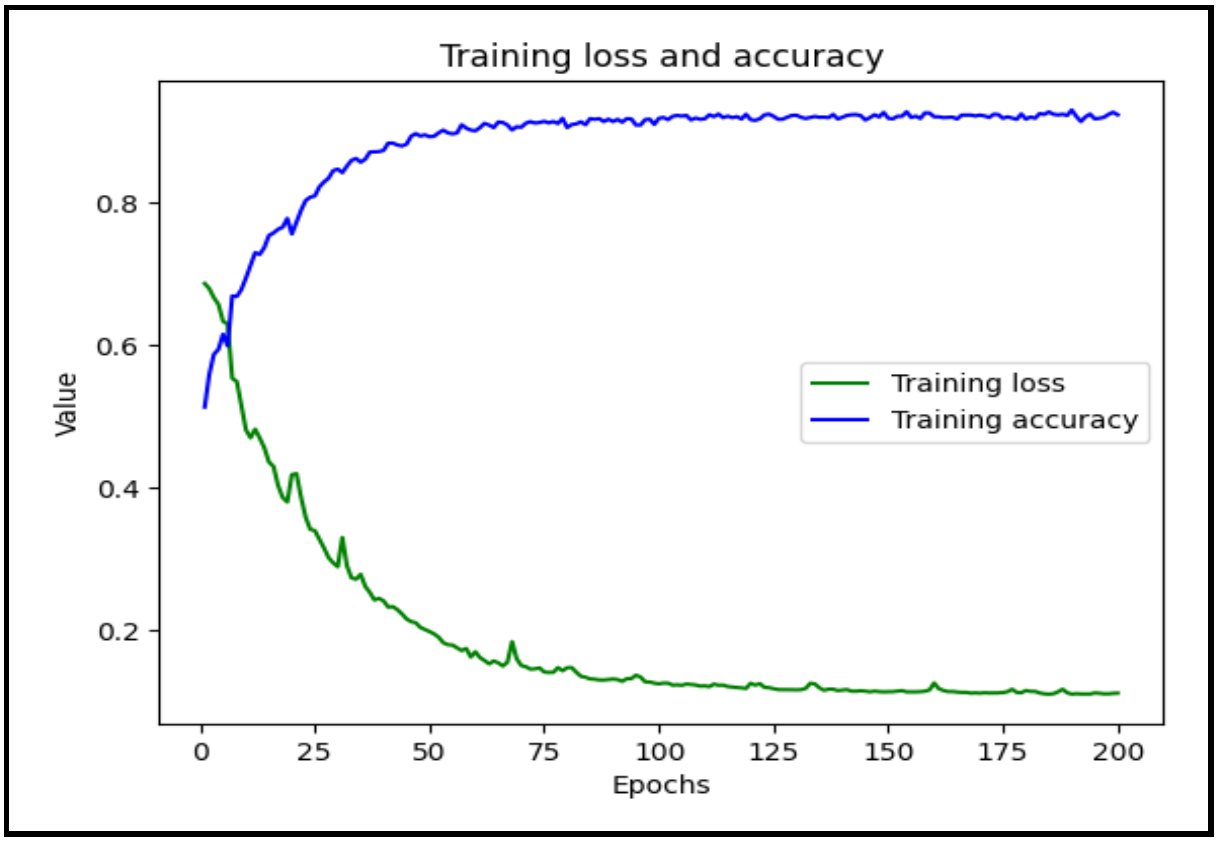


Figure 7.5: Training Loss and Accuracy of SASNN

- Automatic Feature Learning: The SASNN model automatically learns relevant features from the data, eliminating the need for manual feature engineering. This contrasts with PARIS, which relies on predefined features and may not capture intricate patterns as effectively.

- Contextual Information: By incorporating contextual information such as entity attributes and relationships, our model can better discriminate between valid and erroneous sameAs links. PARIS, on the other hand, has limited consideration of contextual information, which can lead to the inclusion of erroneous links.

- Siamese Neural Network Architecture: The use of a Siamese neural network architecture allows our model to learn similarities and dissimilarities between entity pairs effectively. This architecture enables the model to produce more accurate link predictions by leveraging shared weights and focusing on the relational aspects of data pairs. This enhances its ability to detect true sameAs links and reject false ones.

7.6 Conclusion

In this chapter, we introduced a novel approach for sameAs link detection in Linked Data using deep learning Siamese neural networks. Our model, SASNN, demonstrated exceptional performance, surpassing traditional methods with higher precision, recall, and F-score. These results highlight the potential of deep learning techniques to tackle the complexities and heterogeneity inherent in Linked Data. The advancements represented by our SASNN model mark a significant progression in the field of sameAs link detection, overcoming many limitations of conventional approaches.

Our work contributes notably to the field through enhanced accuracy in link discovery, automatic feature learning, and improved contextual link detection. By employing the Siamese neural network architecture, our model adeptly learns similarities and dissimilarities between entity pairs, resulting in more accurate and reliable link predictions. This is crucial for real-world applications, where the precision of sameAs links directly influences the quality and interoperability of interconnected datasets.

Addressing uncertainty in Linked Data is a significant aspect of our approach. Inaccurate sameAs links can lead to errors propagating through data integration systems, diminishing their reliability. By resolving this uncertainty, our model ensures that linked data remains robust, dependable, and valuable for various uses, including data analytics, semantic search, and knowledge graph construction.

In our work, we introduce a Siamese neural network-based approach for sameAs link detection, aiming to enhance accuracy, scalability, and adaptability in Linked Data. Traditional ontology matching methods often struggle with scalability, ontology evolution, and heterogeneous data, while existing Linked Data approaches suffer from contextual limitations and uncertainty handling. Our model addresses these challenges by leveraging deep learning techniques to automatically learn features, reducing dependency on predefined heuristics and improving generalization across diverse datasets. A key advantage of our approach is contextual link detection, where entity attributes, relationships, and semantic context are incorporated to refine link predictions and distinguish valid links from erroneous ones. Additionally, our method is scalable and efficient, enabling the processing of large-scale linked datasets while maintaining accuracy. Unlike traditional rule-based systems, our end-to-end learning framework eliminates the need for manual feature engineering, making it more robust and adaptable.

Looking ahead, our future work will concentrate on two primary areas: enhancing the scalability of our model and validating its effectiveness in practical, real-world scenarios.

Chapter 8

Deep Learning-based Erroneous Data Detection in Linked Data Context Using LSTM and Embeddings

8.1 Introduction

Knowledge graphs leveraging Linked Data are crucial in various domains, including life sciences, government, and e-commerce. However, the decentralized and open nature of Linked Data makes it vulnerable to errors during data collection, integration, and maintenance. These erroneous triples can significantly compromise the quality, reliability, and utility of knowledge graphs, emphasizing the need for robust error detection mechanisms.

Several methodologies have been proposed for detecting errors in Linked Data, but they often have notable limitations. Some methods are confined to identifying specific types of errors, such as type inconsistencies or missing triples. Certain techniques rely on external resources, like ontologies or other Linked Data graphs, which might not always be accessible or complete. Many approaches face scalability issues when applied to large Linked Data graphs.

This chapter is structured as follows: we begin by formally defining the problem, followed by an overview of our deep learning model architecture. Next, we provide a detailed description of the experimental evaluations. Finally, we conclude the chapter.

8.2 Overview of the Contribution of Errors Detection through Deep Learning Techniques

To address these challenges, this work introduces a novel deep learning approach for error detection in Linked Data. Our model employs Long Short-Term Memory (LSTM) networks to capture long-term dependencies and patterns within RDF triples. By embedding subjects, objects, and relations into dense vectors, the model captures their semantic meanings, providing a generalized solution applicable to various Linked Data sets. The proposed deep learning approach aims to advance the state of error detection in Linked Data, offering a scalable and generalized solution that addresses the limitations of existing methods.

The primary contributions of this chapter are:

1. Development of a deep learning architecture using LSTM networks to model complex RDF triple sequences.
2. Implementation of embedding layers to learn dense representations of entities and relations.
3. Creation of a scalable approach designed to handle large datasets effectively.
4. Consideration of semantic links between entities to enhance error detection capabilities.

8.3 Errors Detection Modeling

We're working on improving how we detect errors in linked data, especially within RDF triples. Our goal is to identify both types of errors: those related to structure and those related to meaning. Semantic errors, for instance, occur when triples are incorrect or don't fit with known information. We aim to create a model that can find these errors across a variety of datasets and domains without relying on specific schema or type details. To achieve this, we use deep learning to develop detailed representations of entities and their connections from large linked data.

The main challenge we're addressing is determining whether RDF triples are accurate. We need to decide if each triple, made up of a subject (S), an object (O), and a relation (R), is valid. This is important for applications that use RDF data, like knowledge graphs and semantic web tools.

Enhancing the detection of errors contributes to better data quality and more reliable application performance. We aim to develop a deep learning model that can analyze RDF triples to identify patterns and assess their validity with high accuracy.

We approach this as a binary classification issue. For every triple with a subject (S), object (O), and relation (R), we want to determine whether it's valid. This can be represented as:

$$Prediction = f(S, R, O) \tag{8.1}$$

Where f is the function that maps the inputs (S, R, O) to a binary result (true or false).

To demonstrate the error detection task in the W3C Uncertainty Ontology, we illustrate the process in Figure 8.1.

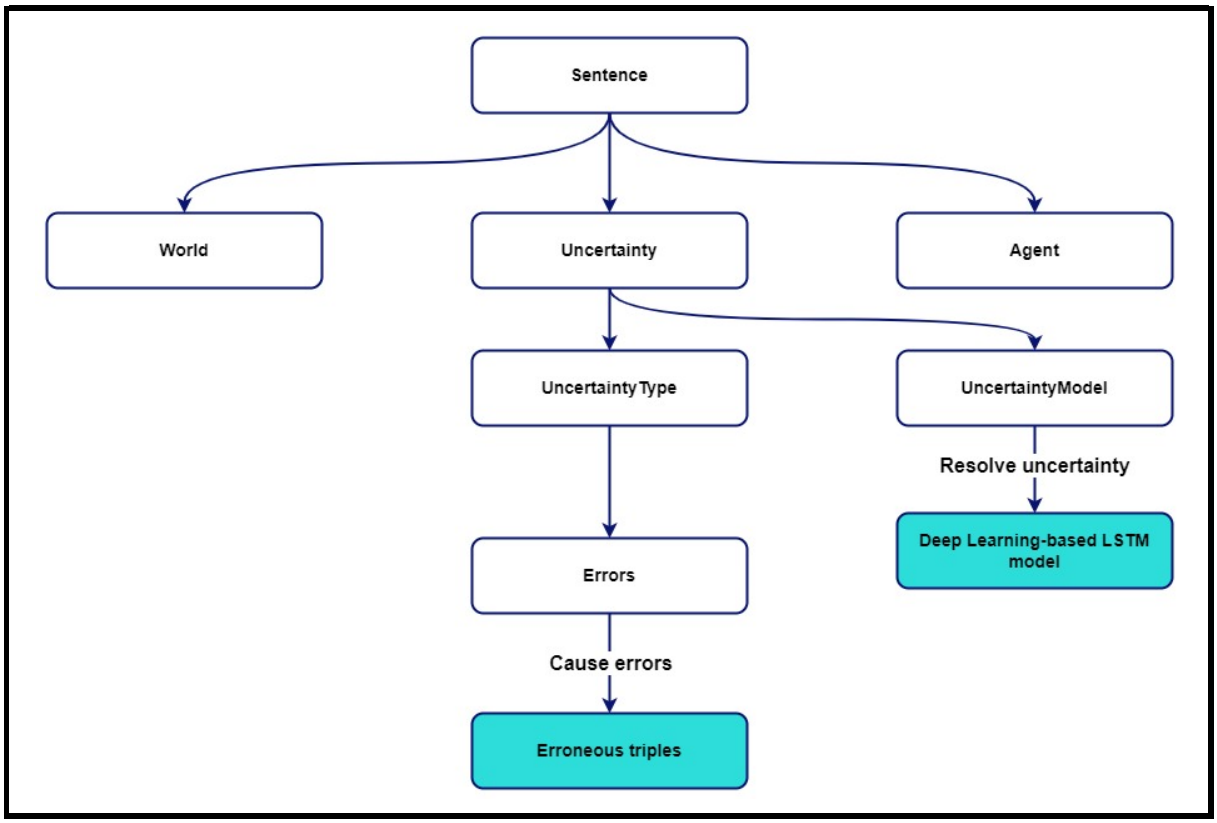


Figure 8.1: Errors Detection in W3C Ontology [107]

8.4 Architecture of the Model

8.4.1 Overview

We follow a well-defined pipeline (illustrated in Figure 8.2) consisting of several steps that process the dataset and train the deep learning model. The pre-processing steps are thoroughly described in Algorithm 8.1. The whole process is illustrated in the following steps:

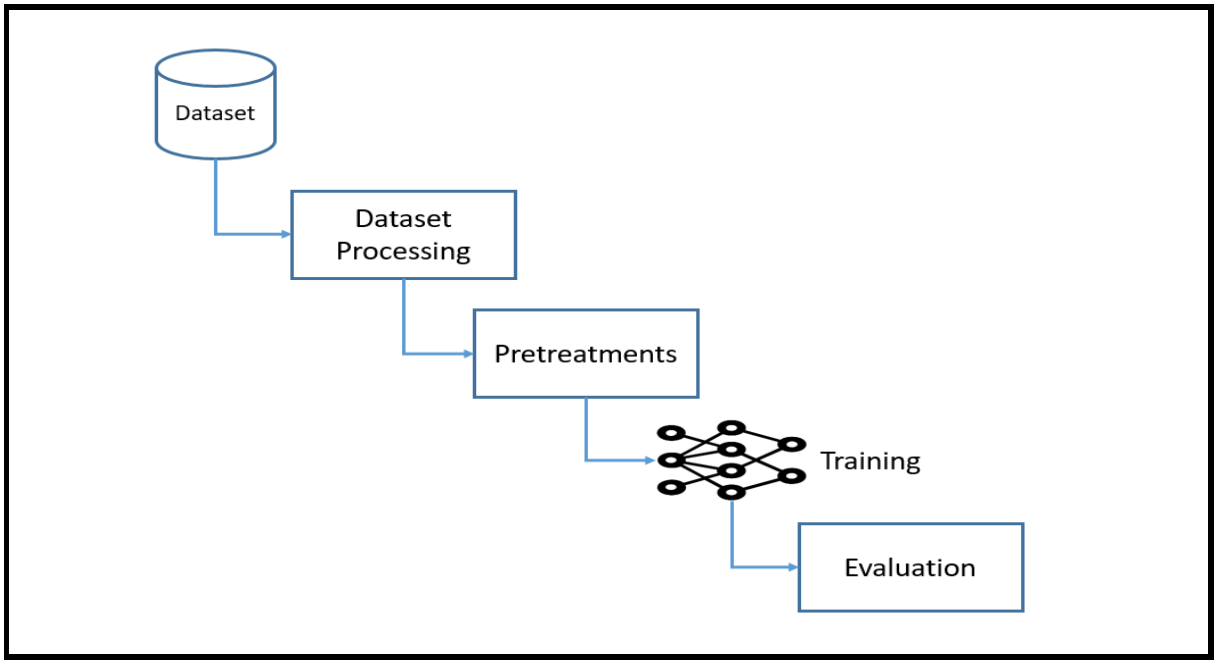


Figure 8.2: Pipeline of our Errors Detection Approach [107]

1. Dataset Processing: The first step involves assembling required RDF triples and organizing them into subject, object, and relation components. It is important to note that this lays the groundwork for model training with input.

2. Pretreatment: This includes deleting or modifying extraneous information from the datasets used to run experiments on and normalizing continuous attributes with values between 0 and 1 to avoid any preference of certain features over others by the classifier.

3. Data Splitting: The next step is dividing our datasets into two parts, training and validation sets, in order to assess how well our models have been trained.

4. Model Design: One of the requirements for designing our deep learning model is to create embedding layers for every input so that they are able to convert the

sequences of subject, object, and relation into dense vectors. These embeddings capture the semantic meaning of the triples. LSTM layers are used on these embedded sequences, which allow the model to learn both sequential dependencies and long-term dependencies in data. The outputs from LSTM layers are then concatenated to integrate patterns from subject, object, and relation. This integrated output is passed through a dense layer with a ReLU activation function for non-linear transformations and feature extraction purposes.

5. Output Layer: For binary classification, in the end, an output layer with a sigmoid activation function is used to forecast the correctness of triples.

Algorithm 8.1: Dataset Creation and Pre-processing for Errors Detection

Input : RDF triples dataset $\mathcal{D} = \{(S_i, P_i, O_i)\}$
Output: Numerical representations of entity pairs for training and testing

Step 1: Dataset Processing
Extract subjects S, predicates P, and objects O from $\mathcal{D}$.

Step 2: Pretreatment
foreach $(S_i, P_i, O_i) \in \mathcal{D}$ **do**
    Remove or modify extraneous information.
    Normalize continuous attributes to $[0, 1]$ to ensure balanced feature importance.

Step 3: Data Splitting
Split $\mathcal{D}$ into training set $\mathcal{D}_{train}$ and validation set $\mathcal{D}_{val}$.

Return: Return the processed dataset for model training.

8.4.2 Erroneous Triples Detection Approach

In this study, we're trying out a new deep learning technique named ETD-LSTM for (Erroneous Triples Detection using LSTM) to check if RDF triples are correct. We focus on three main things: the subject, the object, and their relationship. Our goal is to figure out if each triple is valid by looking at all the triples related to each subject and object.

Here's how we do it: We start by creating a synthetic dataset with sequences for the subject, object, and relationship and labeling each one as true or false. We then split this dataset into two parts: one for training our model and one for testing it.

For the model, we convert these sequences into dense vectors using embedding layers, which helps the model grasp the meaning of the triples. Next, we use Long Short-Term Memory (LSTM) networks to process these vectors. LSTMs are great at handling sequences and remembering important details, which is key for our task.

After processing with LSTMs, we combine the results from the subject, object, and relationship. This combined data goes through a dense layer with a ReLU activation function to extract useful features. Finally, we use a sigmoid function to determine if each triple is valid or not. Figure 8.3 shows how everything fits together.

We train our model using part of the DBpedia dataset and fine-tune it with the Adam optimizer and binary cross-entropy loss function. We then check how well it performs on a validation set and continue training for a set number of epochs.

To see if our model works, we use it to predict the validity of new samples, helping us determine if each triple is likely correct.

Overall, our approach uses deep learning to automatically validate RDF triples. By learning from patterns in the data, our model provides a useful tool for checking RDF triples in different scenarios.

8.4.3 Our Deep Learning Model

Our deep learning model, illustrated in Figure 8.3, uses LSTM layers to determine whether RDF triples are correct. It works with three inputs: the subject, object, and relation. We first turn these inputs into dense vectors using an embedding layer, which gives us fixed-size vectors based on our vocabulary and embedding settings.

Next, the model uses LSTM layers to work with these vectors. LSTMs are great at handling sequences and picking up patterns, so they help the model understand how the subject, object, and relation relate to each other. The way the LSTM layers are set up influences how well the model learns and remembers these connections.

After processing with LSTMs, we combine the results for the subject, object, and relation. This combined data is passed through a dense layer with a ReLU activation function. ReLU helps the model learn more complex patterns by adding non-linearity.

In the end, we use an output layer with a single neuron and a sigmoid function to give us a probability between 0 and 1. A value close to 1 means the triple is likely correct, while a value close to 0 means it's probably not.

We train the model with a labeled dataset and adjust it using the Adam optimizer and binary cross-entropy loss. This helps the model learn the patterns in RDF triples and make accurate predictions about their validity.

8.5 Experiments and Evaluation

In this part, we'll review how we set up our experiment. We tested our model by comparing it with another model from existing research to see how it performs. We'll also discuss

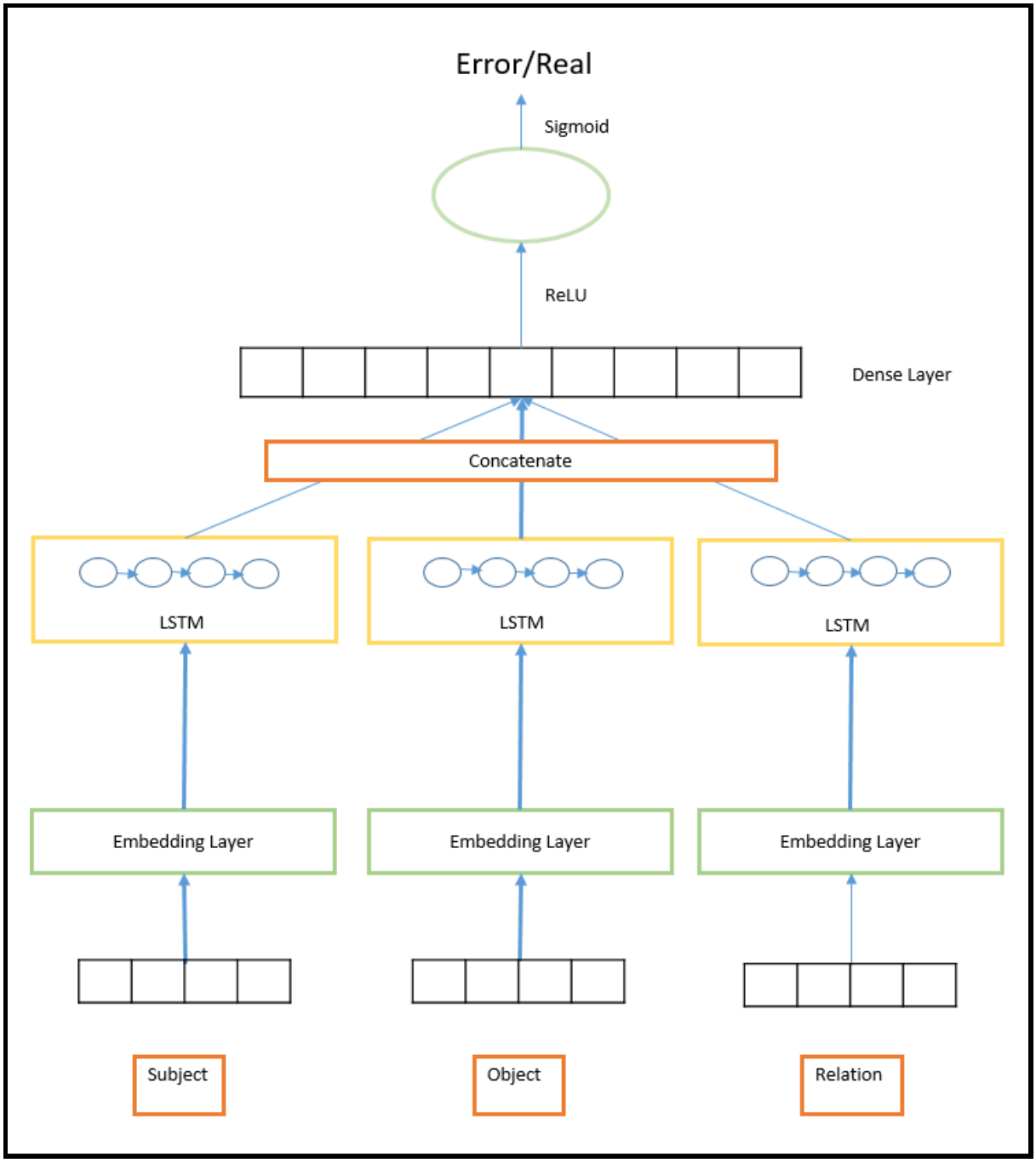


Figure 8.3: Architecture of our Deep Neural Network for Erroneous Triples Detection in Linked Data [107]

Table 8.1: Hyper Parameters of our Errors Detection Model [107]

Hyper parameter	**Value**
Max Sequence Length (inputs)	10
Number of Samples	60000
Embedding Dimension	50
LSTM Units	64
Dense Layer Units	64
Output Activation Function	Sigmoid
Batch Size	32
Number of Epochs	100
Optimizer	Adam
Loss Function	Binary Crossentropy

the results we found.

8.5.1 Experiments

We trained our model using part of the DBpedia dataset on a T4 GPU from Google Colaboratory. Due to some technical limits, we used 54,000 tuples for training and kept 6,000 tuples for testing. You can find the model's hyperparameters in Table 8.1.

We used TensorFlow and Keras to create our model. It breaks down the subject, relation, and object separately, each with its own set of layers. After combining the results, we use a dense layer with 64 units and a ReLU activation and finish with a single neuron to decide the final outcome. We trained it using the Adam optimizer and binary cross-entropy loss for 100 epochs with a batch size of 32.

Our model is designed to identify erroneous triples within DBpedia, distinguishing between correct and incorrect factual statements. Below, we provide one valid (true) and one erroneous (false) example generated by our model.

For a correct fact, our model validates the following triple:

```
<http://dbpedia.org/resource/Isaac_Newton>
   <http://dbpedia.org/ontology/birthPlace>
   <http://dbpedia.org/resource/Woolsthorpe,_Lincolnshire> .
```

This triple is valid since Isaac Newton was indeed born in Woolsthorpe, Lincolnshire.

Conversely, our model correctly flags the following erroneous triple:

```
<http://dbpedia.org/resource/Barack_Obama>
   <http://dbpedia.org/ontology/birthPlace>
   <http://dbpedia.org/resource/Kenya> . (False Prediction)
```

This statement is incorrect because Barack Obama was born in Honolulu, Hawaii, not in Kenya. Our model successfully detects such factual inconsistencies, improving the reliability of knowledge graphs by filtering out erroneous triples.

8.5.2 Evaluation

In this part, we go over how we tested our model. We used the DBpedia dataset to see how our model stacked up against another top model we found. This gave us a clear idea of how our model performs in real-world scenarios and highlighted what it does well and where it might need some tweaks. By evaluating our model's performance, we could see what it does well and where it needs improvement. This process is essential for assessing how effectively our model handles real-world data and for making informed decisions about its usefulness and impact.

8.5.2.1 Results

We tested our model thoroughly using a part of the DBpedia dataset, and you can see the results in Table 8.2 and Figure 8.4. We also compared our model to the one from [94]. The comparison results are shown in Table 8.3.

For this test, we looked at 200 triples with the predicate: http://dbpedia.org/property/author. Here's how our model did:

- True Positives (TP): 77 correct classifications.
- False Negatives (FN): 37 instances.
- False Positives (FP): 22 instances.
- True Negatives (TN): 64 correct classifications.

Our model had a precision score of 0.78, which is a bit better than the 0.76 score of the [94] model. It also had a much higher recall score of 0.68 compared to 0.31 for the [94] model. These results show that our model performs quite well, especially in precision and recall, compared to the other model.

8.5.2.2 Discussion

Debattista et al. [94] came up with a method to find incorrect RDF statements in Linked Data to improve data quality. They tried to fix problems like incomplete and messy data, especially where properties aren't clearly defined. But their method has some key issues:

Table 8.2: Experimentation Results on DBpedia of our Errors Detection Model [107]

Measure	Value
TP	2291
FN	757
FP	1209
TN	1743
Precision	0.65
Recall	0.75
F1 Score	0.70

Table 8.3: Evaluation of our Errors Detection Model with a state-of-art Model on DBpedia [107]

Measure	Our Model	State-of-art Model [94]
TP	77	/
FN	37	/
FP	22	/
TN	64	/
Precision	**0.78**	0.76
Recall	**0.68**	0.31
F1 Score	0.73	/

- It does not support datasets containing blank nodes, which are used in RDF to represent resources without explicitly naming them. This limits the flexibility of the method, as blank nodes can play a crucial role in expressing relationships between data that may not have a global identifier, especially in cases where anonymity or partial information is required.

- It doesn't fully use typed annotations, so it might miss some important details. It also struggles with hierarchical relationships, which are pretty common in Linked Data. Plus, there could be biases in how samples are selected, and there's some uncertainty about outliers and the accuracy of the results.

Our method overcomes these problems with several advantages:

- We use a new deep learning model with LSTM networks that are great at capturing complex relationships.

- Our model creates detailed representations of subjects, objects, and relations, capturing their meanings better.

- It's designed to handle large datasets, making it practical for real-world use.

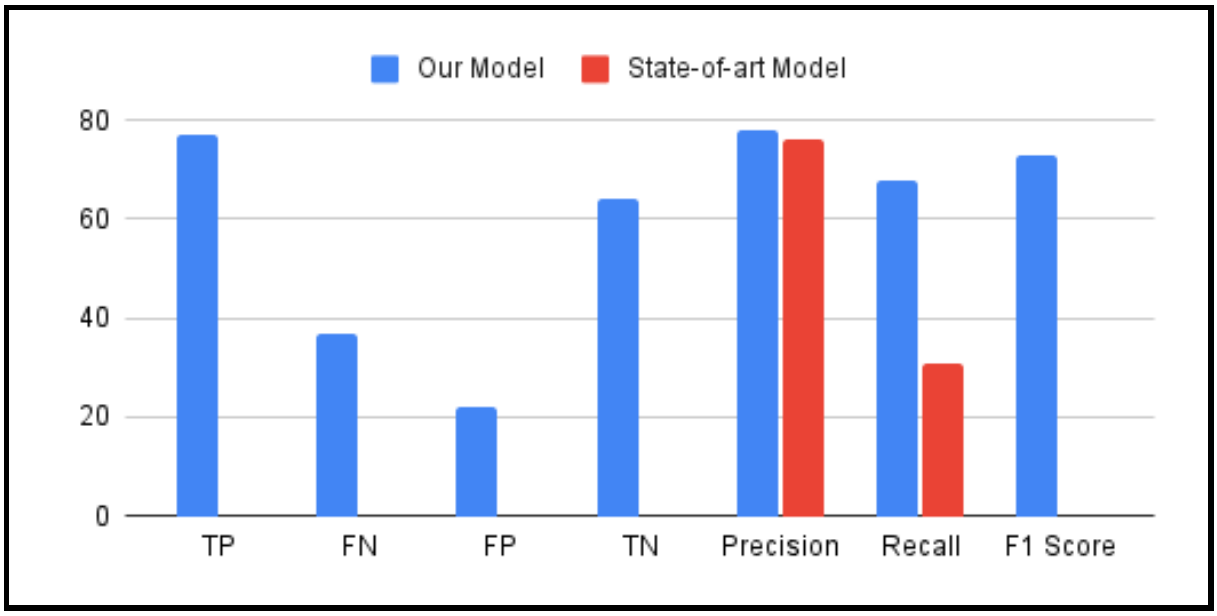


Figure 8.4: Histogram of Evaluation of our Errors Detection Model with a state-of-art Model on DBpedia [107]

- We also consider semantic links between entities, which adds more depth to our analysis.

Overall, our method is a more effective solution for improving data quality in Linked Data compared to the earlier approach.

8.6 Conclusion

In this work, we've introduced a new deep learning model for spotting errors in RDF triples. Our model uses LSTM networks to capture long-term patterns and dependencies within the triple sequences. By turning subjects, objects, and relations into dense vectors, it learns their meanings and boosts error detection accuracy.

Our tests on a validation set show that this model is very accurate at finding incorrect RDF triples. This improvement in data quality is crucial for applications that depend on linked data, as it helps maintain the integrity and accuracy of the knowledge graph.

Unlike traditional methods that struggle with high data dimensionality and manual intervention, our approach employs embedding layers to learn dense representations of entities and relations, enhancing the accuracy of anomaly detection. Additionally, we incorporate semantic link analysis to improve the identification of inconsistencies between entities, addressing a key limitation of existing models that overlook deeper relationships within the data.

To tackle scalability challenges, our solution is designed to efficiently handle large datasets, overcoming the limitations of graph-based and statistical methods that fail to scale effectively. Furthermore, by reducing dependence on manual corrections, we enhance automation and efficiency in the error detection process. Finally, while many machine learning models risk amplifying errors due to incomplete or low-quality input data, our approach mitigates this risk by leveraging context-aware representations of RDF triples, ensuring more robust and accurate error detection.

In the next chapter, we aim to embed our model into current systems that manage different types of data connections, such as type and sameAs links. This integration will assist in identifying and fixing errors in generated triples, which will enhance the overall reliability of these systems. In our future work, we plan to apply our model to a variety of linked datasets to assess its effectiveness in different contexts.

Chapter 9

Case Study: Applying Our Deep Learning Contributions for Incompleteness Resolution and Error Detection in UniProt

9.1 Introduction

Linked data provides a powerful method for structuring and interlinking data from diverse sources, facilitating the integration of information across various domains. Despite its advantages, linked data is frequently plagued by several sources of uncertainty. These uncertainties include incomplete information, incorrect relationships, and erroneous data entries, which can significantly impact the reliability and usability of the data. Addressing these issues is critical for enhancing the quality and accuracy of linked data.

This chapter presents a comprehensive framework designed to address uncertainties in linked data, specifically applied to the UniProt dataset. UniProt, a prominent resource in the field of protein sequence and functional information, offers a wealth of RDF triples that are crucial for biological research and applications. However, like other linked data sources, UniProt is not immune to uncertainties such as missing types, incomplete links, and erroneous triples.

This chapter provides a comprehensive overview of our work and its context. It begins by presenting the methodology used for conducting the case study, followed by an exploration of how our contributions align with the W3C ontology framework, emphasizing their relevance and integration. The discussion then shifts to the UniProt dataset, focusing on the critical need to address uncertainty to ensure data reliability. The developed

framework for managing uncertainties in linked data is outlined, including the methodology used. The chapter also covers the experiments conducted, such as the training and testing of our models, and provides an evaluation of the results, offering insights into the performance and effectiveness of the framework. Finally, the chapter concludes with a summary of the key findings and contributions.

9.2 Methodology for Conducting the Case Study

We propose our four key contributions to tackle these challenges:

1. ED-MTD Missing Type Detection: This contribution focuses on identifying types that are absent from the dataset. Accurately detecting and filling these missing types is vital for ensuring the completeness of the data.

2. LinkED-S2S for Missing Link Detection: Here, we aim to uncover relationships between entities that should be present but are currently missing. Identifying these gaps helps in enriching the dataset and improving its relational accuracy.

3. SASNN for SameAs Link Detection: This contribution involves identifying equivalent entities across the dataset. Correctly determining sameAs links is essential for integrating data from multiple entities and resolving entity duplication.

4. ETD-LSTM for Erroneous Triple Detection: This part of the framework focuses on spotting incorrect triples within the data. Filtering out erroneous triples is crucial for maintaining the integrity and reliability of the dataset.

The primary goal of this study is to integrate these contributions into a unified framework. This framework will first use the Missing Type Detection, Missing Link Detection, and sameAs Link Detection contributions to generate additional triples, addressing the problem of data incompleteness. Subsequently, the Erroneous Triple Detection will be applied to these generated triples to filter out errors, thus refining the dataset and enhancing its overall quality.

9.3 Uncertainty in W3C Ontology

The Uncertainty Ontology by W3C is a pivotal framework designed to manage and mitigate various types of uncertainties inherent in linked data. Uncertainty in linked data can stem from multiple sources, including incomplete information, ambiguous entity relationships, and errors in data entries. Recognizing the critical nature of addressing these

uncertainties, our contributions have been geared towards leveraging deep learning models to enhance the reliability and robustness of linked data.

Uncertainty in Linked Data can be broadly classified into multiple types, including, but not limited to: Incompleteness and Errors. In the context of our work, we focus specifically on addressing these two types of uncertainty, as they play a significant role in impacting the quality and completeness of Linked Data.

- Incompleteness: This occurs when certain data points or relationships are missing within the dataset. In the context of linked data, incompleteness can significantly hamper the dataset's utility and comprehensiveness.
- Errors: These are inaccuracies or incorrect data points that can lead to faulty conclusions and analyses. Errors can be due to incorrect data entries, outdated information, or misclassified entities.

The W3C Uncertainty Ontology provides a structured approach to identifying and managing these uncertainties, thereby ensuring that the linked data remains a reliable resource for various applications.

Our contributions are specifically focused on resolving these uncertainties using advanced deep learning models. By integrating our solutions within the W3C Uncertainty Ontology framework, as illustrated in Figure 9.1, we aim to enhance the quality and reliability of linked data significantly.

We address uncertainty through the following contributions:

1. Incompleteness
    - Links Detection: To identify and establish missing links within the dataset. Using deep learning models, we analyze the dataset to detect potential connections between entities that are not explicitly linked. This process helps uncover hidden relationships and enhance the dataset's completeness.
    - Types Detection: To detect and assign missing types to entities within the dataset. Deep learning techniques are employed to scrutinize the dataset for entities that lack proper classification. By assigning appropriate types to these entities, we ensure that each entity is accurately represented, thereby improving the dataset's semantic richness.
    - SameAs Links Detection: To identify semantically similar entities within a dataset. Our deep learning model detects and links similar entities, effectively enriching the dataset. This not only consolidates information but also provides a more interconnected and comprehensive data landscape.

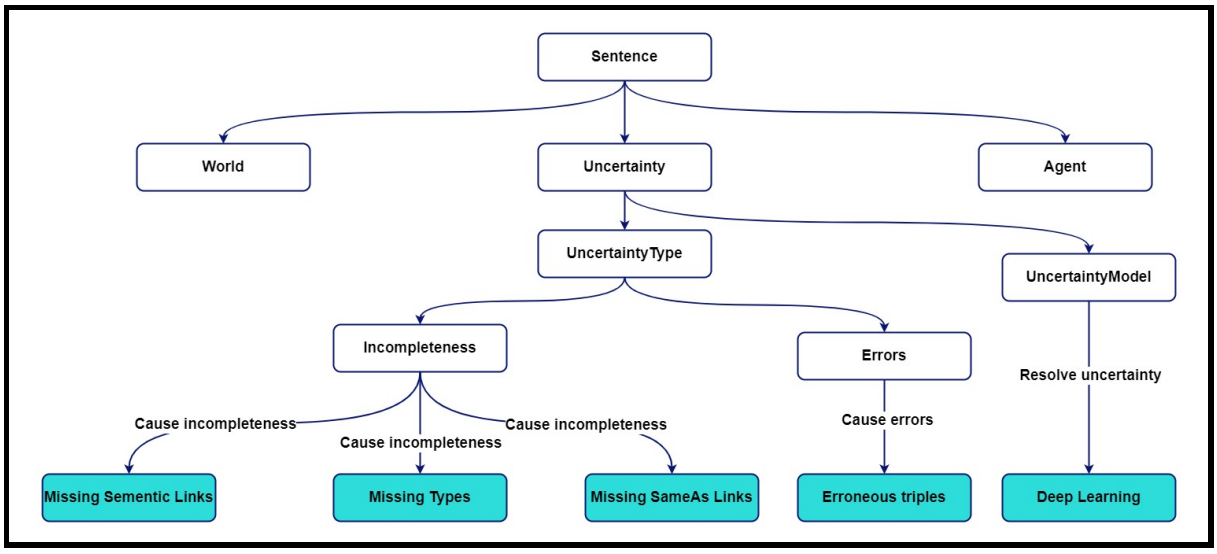


Figure 9.1: Uncertainty Ontology for Resolving Incompleteness and Detecting Errors

2. Errors

   - Errors Detection: To identify and rectify incorrect data entries within the dataset. Deep learning models are designed to differentiate between correct and false triples, ensuring the dataset's integrity and reliability.

The integration of our deep learning-based solutions with the W3C Uncertainty Ontology framework provides a robust mechanism for managing uncertainties in linked data. Our contributions align with the W3C's objectives of creating a structured and reliable data environment. By addressing both incompleteness and errors, we ensure that the linked data not only becomes more accurate but also more useful for various analytical and practical applications.

Our deep learning contributions to the W3C Uncertainty Ontology represent a significant advancement in the context of linked data. By effectively addressing uncertainties, we pave the way for more reliable and comprehensive data, thereby enhancing the overall utility of the W3C Ontology framework.

9.4 Uncertainty in the UniProt Dataset

UniProt, or The Universal Protein Resource, is a vast and invaluable database dedicated to protein sequences and annotations. It is a collaborative effort between the European Bioinformatics Institute (EMBL-EBI), the Swiss Institute of Bioinformatics (SIB), and

the Protein Information Resource (PIR). This partnership brings together expertise from these institutions to manage and enhance a wide array of protein-related data.

UniProt comprises several key components:

- UniProtKB (UniProt Knowledgebase): This is the central hub of protein information, including manually curated entries (Swiss-Prot) and automatically annotated entries (TrEMBL).
- UniRef (UniProt Reference Clusters): This database provides clustered sets of sequences to facilitate sequence similarity searches.
- UniParc (UniProt Archive): This archive maintains a comprehensive collection of protein sequences from various sources, tracking changes over time.

The development and maintenance of UniProt involve ongoing efforts from these institutions. EMBL-EBI and SIB developed Swiss-Prot and TrEMBL, while PIR created PIR-PSD (the Protein Sequence Database). Initially, these datasets operated separately with different focuses on protein sequence coverage and annotation. However, they have since been integrated to provide a more unified and comprehensive resource.

9.4.1 Overview of UniProt Dataset

UniProt offers detailed RDF triples representing protein sequences, functions, and relationships. It serves as a cornerstone for bioinformatics research, providing essential data for various applications ranging from fundamental biological research to clinical studies. The accuracy and completeness of this data are crucial for its users, including researchers, healthcare professionals, and data scientists.

UniProt is widely used in practical applications, such as drug discovery, where researchers explore protein targets for new medications, and genetic research, where it aids in understanding the functions of genes and their associated proteins. Additionally, it supports protein function prediction, disease-related gene mapping, and enzyme activity studies, helping to identify biomarkers and therapeutic targets for diseases.

9.4.2 Impact of Uncertainty in UniProt

Uncertainty within the UniProt dataset can have significant consequences:

- Incomplete Information: Missing or inaccurate annotations related to protein functions can create gaps in our understanding of biological processes and protein roles.

This can affect functional studies and hinder the development of therapeutic strategies.

- Incorrect Relationships: Errors in the connections between proteins or other biological entities can misrepresent interactions and functional associations. Such inaccuracies can distort network analyses and lead to misleading conclusions.
- Misleading Data: Incorrect or inconsistent triples can impact downstream analyses and applications, potentially leading to flawed results and interpretations. This compromises the validity of research findings and the efficacy of data-driven applications.

Addressing these uncertainties is essential for maintaining UniProt's reliability. By identifying and correcting missing or incorrect information, we aim to enhance the dataset's overall quality and support more accurate and meaningful research outcomes.

9.5 Our Framework for Uncertainty Handling Approach

Handling uncertainty in linked data, particularly in comprehensive datasets like UniProt, is critical for maintaining data integrity and supporting accurate research outcomes. Our proposed framework (illustrated in Figure 9.2) addresses these uncertainties through a systematic approach, leveraging deep learning models to manage different aspects of data quality.

The UniProt dataset is structured as RDF triples, each triple consisting of a subject, predicate, and object. This format allows for rich semantic representations of biological data, including various types of information such as protein sequences, functions, and interactions.

Our framework is designed to tackle four primary sources of uncertainty in the UniProt dataset:

1. Missing Type Detection: Identifies types that should be present but are missing from the dataset.
2. Missing Link Detection: Uncovers relationships between entities that are currently missing.
3. SameAs Link Detection: Determines entities within the UniProt dataset that are equivalent to each other.

4. Erroneous Triple Detection: Filters out incorrect triples from those generated by the first three components.

This approach combines these components into a cohesive system that not only addresses data incompleteness but also ensures the accuracy of the generated triples.

9.5.1 Framework Overview

Our framework integrates four deep learning models, each addressing a specific aspect of uncertainty:

- An encoder-decoder model with an attention mechanism is used to predict and identify missing types within the dataset.
- A sequence-to-sequence model, enhanced with an attention mechanism, is applied to detect missing relationships between entities.
- A Siamese neural network is leveraged to find equivalent entities in the UniProt dataset.
- A binary classification model using Long Short-Term Memory networks is employed to detect and filter out erroneous triples in the generated data.

The integration of these models allows for a comprehensive approach to managing and improving the quality of linked data.

9.5.2 Methodology

The methodology for implementing our framework involves several key steps. The process is illustrated in Algorithm 9.1:

1. Data Creation and Preprocessing: We begin by selecting subsets of the UniProt dataset. This includes creating training and testing datasets for each model. Data preprocessing involves cleaning the data, handling missing values, and formatting the data to be compatible with our deep learning models.
    - Data Creation: Read data from the UniProt dataset in the form of RDF triples.
    - Preprocessing: transform RDF triples into numeric vectors suitable for deep learning models; build training and testing datasets by identifying inputs and outputs for each model.

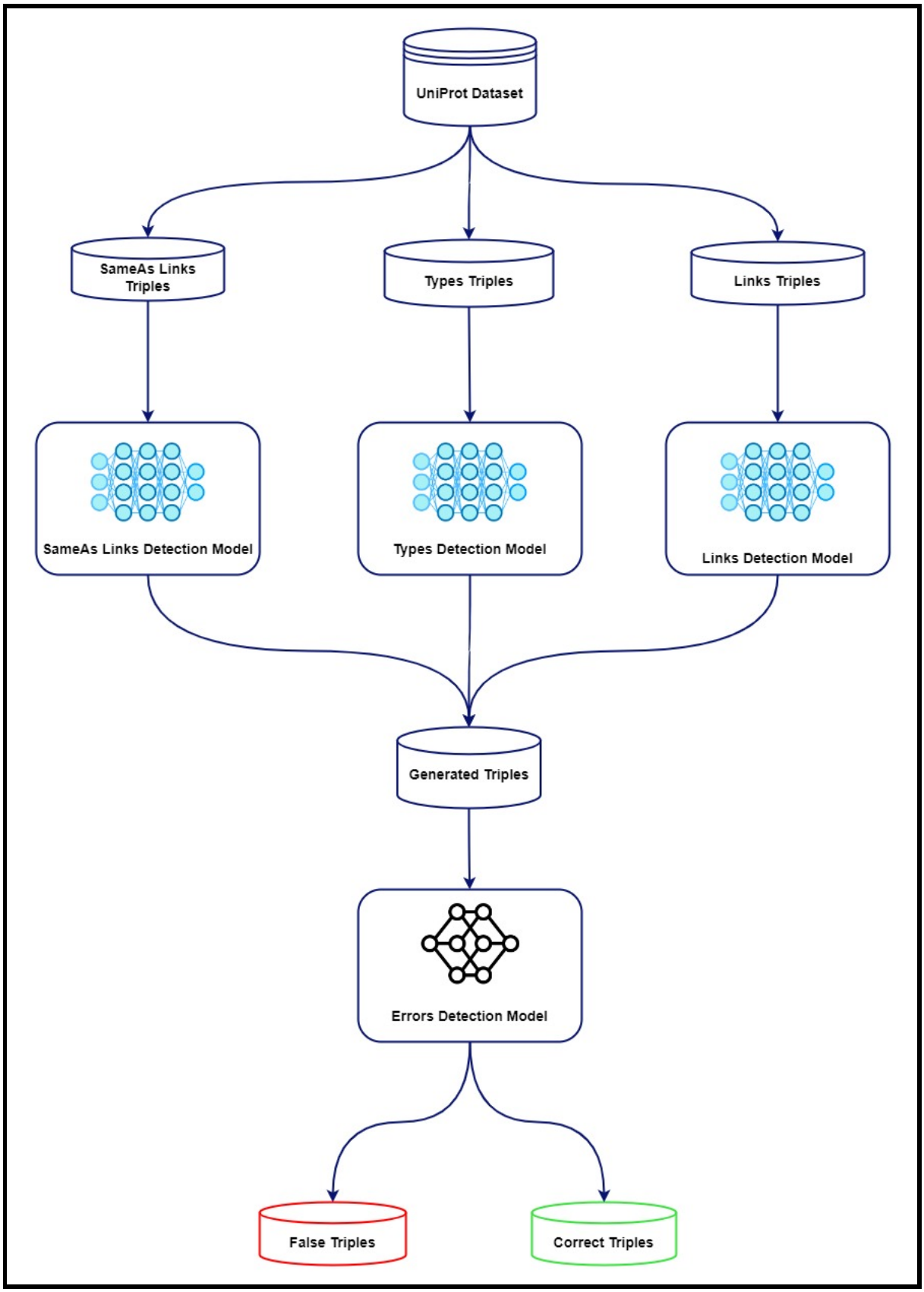


Figure 9.2: Framework Architecture

2. Generating Missing Triples:

   - The encoder-decoder model with attention is trained to predict missing types based on the existing dataset.
   - The sequence-to-sequence model, equipped with attention, identifies and generates missing relationships between entities.
   - The Siamese neural network model detects and generates sameAs links to recognize equivalent entities within the dataset.

3. Filtering Erroneous Triples:

   - The LSTM-based binary classification model is employed to evaluate and filter the triples generated by the previous models. This step ensures that the triples are accurate and reliable by identifying and removing erroneous entries.

Each component of the framework is designed to complement the others, resulting in a robust system that addresses the various aspects of uncertainty in linked data. By systematically generating missing triples and filtering out errors, our approach aims to enhance the quality and completeness of the UniProt dataset.

Algorithm 9.1: Dataset Processing and Triple Generation for our Framework

Data: UniProt RDF dataset
Result: Processed dataset, generated missing triples, and filtered triples
Step 1: Data Creation and Preprocessing
Read RDF triples from the UniProt dataset
Clean the data and handle missing values
Convert RDF triples into numerical vectors
Split the dataset into training and testing sets
Step 2: Generating Missing Triples
Train an encoder-decoder model with attention to predict missing types
Train a sequence-to-sequence model with attention to infer missing relationships
Train a Siamese neural network to detect sameAs links and recognize equivalent entities
Step 3: Filtering Erroneous Triples
Use an LSTM-based binary classification model to evaluate generated triples
Remove erroneous or unreliable triples
return Processed dataset with reliable triples

9.6 Experiments

Our experiments were conducted using the T4 GPU provided by Google Colaboratory. We trained and evaluated our deep learning models on three subsets of the UniProt dataset, each representing different biological categories. The models were trained on the following subsets:

- UniProtKB unreviewed Archaea (Hadarchaeota).
- UniProtKB reviewed Bacteria (Campylobacterota).
- UniProtKB reviewed Eukaryote (Amoebozoa).

The training and testing data were divided as follows:

1. Types Detection Model:
    - Subset 1: 50000 triples (70% training, 30% testing).
    - Subset 2: 50000 triples (70% training, 30% testing)
    - Subset 3: 50000 triples (70% training, 30% testing)
2. Links Detection Model:
    - Subset 1: 100000 triples (70% training, 30% testing)
    - Subset 2: 100000 triples (70% training, 30% testing)
    - Subset 3: 100000 triples (70% training, 30% testing)
3. SameAs Links Detection Model:
    - Subset 1: 30000 triples (70% training, 30% testing)
    - Subset 2: 30000 triples (70% training, 30% testing)
    - Subset 3: 30000 triples (70% training, 30% testing)

All consolidated generated triples from the testing phase were fed into the Erroneous Triple Detection Model, with a total of 100000 triples. The dataset for this model was also split into 70% for training and 30% for testing.

The training and testing data used for each model and subset are illustrated in Table 9.1:

Our approach leverages a set of intelligent models to improve the structuring and reliability of UniProt data. It begins with link detection, where our model identifies implicit relationships between entities. For example, it can generate a link indicating that one protein interacts with another:

Table 9.1: Training and Testing Data for Each Model [108]

Model	**Subset**	**Training (70%)**	**Testing (30%)**
Types Detection ED-MTD	Subset 1	35000	15000
	Subset 2	35000	15000
	Subset 3	35000	15000
Links Detection LinkED-S2S	Subset 1	70000	30000
	Subset 2	70000	30000
	Subset 3	70000	30000
sameAs Links Detection SASNN	Subset 1	21000	9000
	Subset 2	21000	9000
	Subset 3	21000	9000
Erroneous Triple Detection ETD-LSTM	All subsets	70000	30000

```
<http://purl.uniprot.org/uniprot/P12345>
  <http://purl.uniprot.org/core/interactsWith>
  <http://purl.uniprot.org/uniprot/Q67890> .
```

This triple enriches the knowledge graph by capturing biologically relevant interactions.

Next, we apply type detection, which assigns semantic categories to entities. Our model analyzes a protein's properties to determine its correct classification:

```
<http://purl.uniprot.org/uniprot/P12345>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://purl.uniprot.org/core/Enzyme> .
```

Here, protein P12345 is correctly classified as an enzyme, making it easier to use in bioinformatics analyses.

Data structuring does not stop there. We also integrate a sameAs link detection model, which identifies duplicate entities within UniProt and aligns them. For example, our model detects that P12345 and P12345-1 are actually different variants of the same protein and generates the following link:

```
<http://purl.uniprot.org/uniprot/P12345>
   <http://www.w3.org/2002/07/owl#sameAs>
   <http://purl.uniprot.org/uniprot/P12345-1> .
```

Conversely, it can reject incorrect alignments, such as linking P12345 and Q67890, which are two distinct proteins:

```
<http://purl.uniprot.org/uniprot/P12345>
   <http://www.w3.org/2002/07/owl#sameAs>
   <http://purl.uniprot.org/uniprot/Q67890> . (False Prediction)
```

Finally, to ensure the quality of the results generated by our models, we apply error detection, which filters out incorrect triples before their final integration. For instance, a valid triple indicating an interaction between P12345 and Q67890 is retained:

```
<http://purl.uniprot.org/uniprot/P12345>
   <http://purl.uniprot.org/core/interactsWith>
   <http://purl.uniprot.org/uniprot/,> .
```

On the other hand, an erroneous relation with a non-existent or incorrect entity, such as Z99999, is detected and removed:

```
<http://purl.uniprot.org/uniprot/P12345>
   <http://purl.uniprot.org/core/interactsWith>
   <http://purl.uniprot.org/uniprot/P52658> . (False Prediction)
```

By following this approach, we enhance data quality by enriching the UniProt knowledge graph with meaningful links, accurately classifying entities, aligning duplicates, and filtering out errors. Our pipeline ensures better exploitation of biological knowledge and improves the reliability of generated information.

9.7 Results and Discussion

The performance metrics for the three deep learning models, Types Detection, Links Detection, and sameAs Links Detection, are presented in Table 9.2, Figure 9.3 (before error detection) and Figure 9.4 (after error detection). We calculated the recall, precision, and F-measure for the test data and then evaluated the enhanced results after applying the Erroneous Triple Detection Model.

Table 9.2: Performance Metrics for Test Data Before and After Error Detection [108]

Model	**Subset**	**Recall**	**Precision**	**F-Measure**
Types Detection ED-MTD	Subset 1	0.78	0.80	0.79
	Subset 2	0.75	0.77	0.76
	Subset 3	0.77	0.79	0.78
Links Detection LinkED-S2S	Subset 1	0.82	0.85	0.83
	Subset 2	0.80	0.83	0.81
	Subset 3	0.79	0.81	0.80
sameAs Links Detection SASNN	Subset 1	0.92	0.94	0.93
	Subset 2	0.89	0.91	0.90
	Subset 3	0.86	0.88	0.87
Enhanced Results (Post-Error Detection) ETD-LSTM	**All subsets**	**0.90**	**0.92**	**0.91**

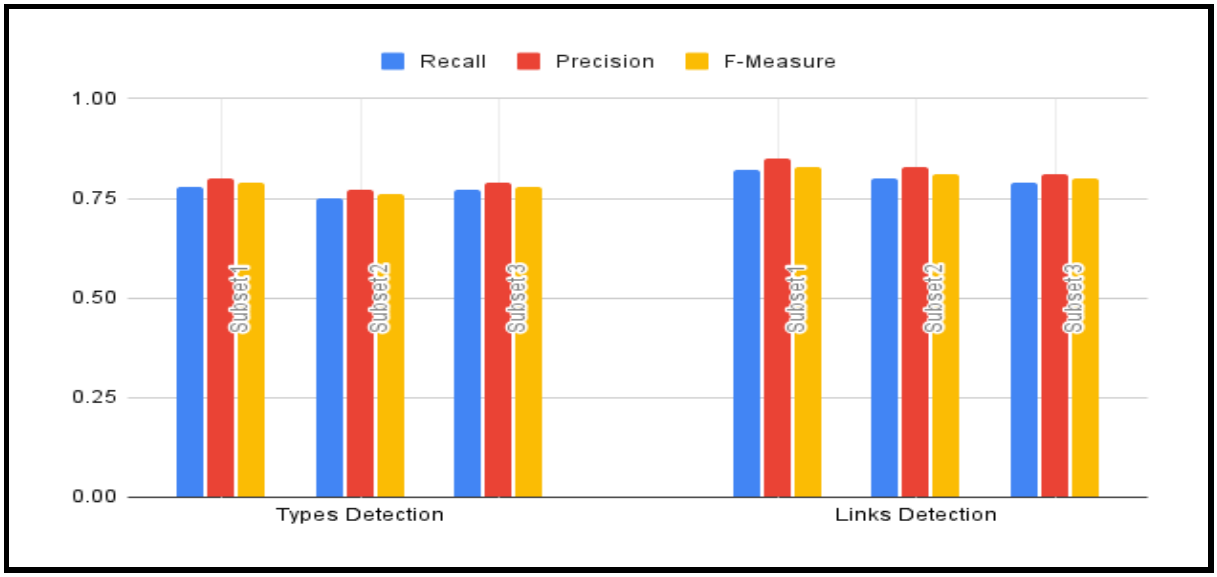


Figure 9.3: Histogram of Performance Metrics for Test Data Before Error Detection [108]

Our approach significantly advances the handling of uncertainty in linked data, specifically within the UniProt dataset. The deep learning models employed—encoder-decoder with attention for missing types detection, sequence-to-sequence with attention for link detection, Siamese neural network for sameAs link detection, and LSTM-based binary classification for error detection—demonstrate robust performance in addressing various types of data uncertainties.

- Importance of Contributions: Each model targets a specific aspect of data uncer-

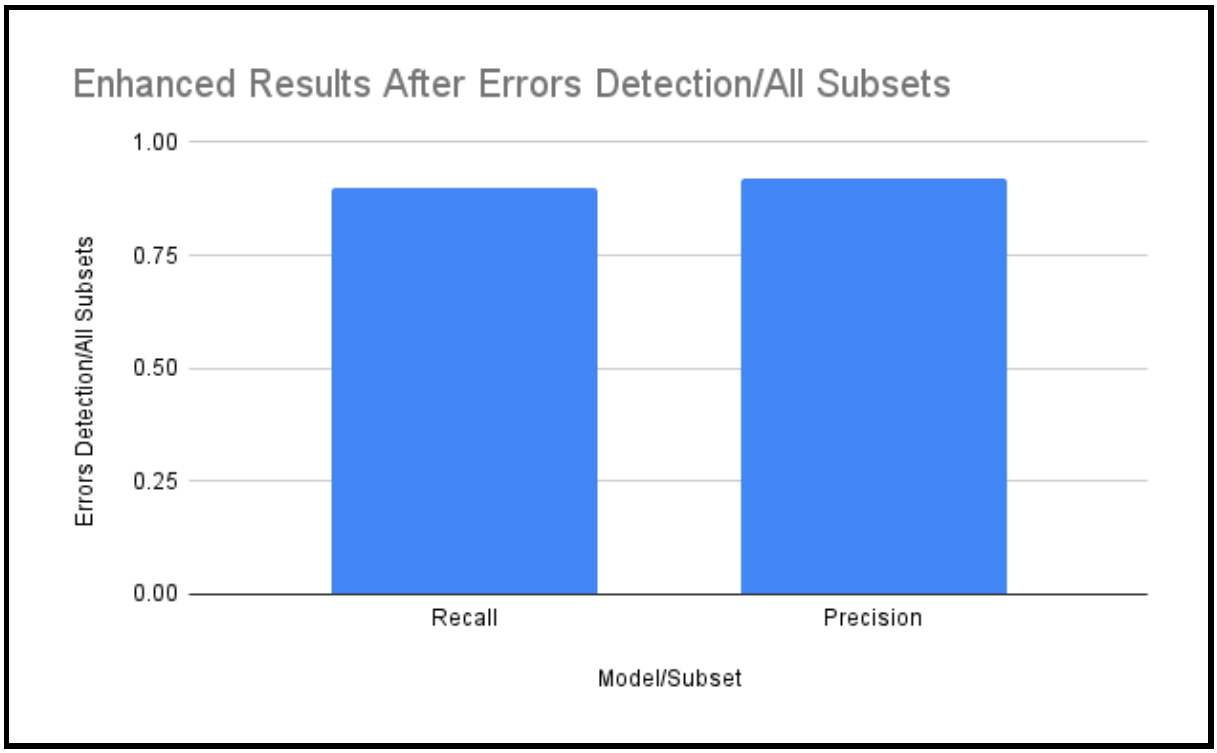


Figure 9.4: Enhanced Results After Errors Detection for All Subsets [108]

tainty. By integrating these models into a unified framework, we address the completeness and accuracy of linked data. The models efficiently identify missing types, relationships, and equivalent entities while filtering out erroneous triples, thereby enhancing the overall data quality.

- Deep Learning Techniques: The use of advanced deep learning techniques, such as attention mechanisms and LSTM networks, allows for the precise handling of complex data patterns and relationships. These techniques contribute to high recall, precision, and F-measure scores, highlighting their effectiveness in managing uncertainty.

- Model Quality: The significant improvement in performance metrics after applying the error detection model underscores the effectiveness of our approach. The error detection model's ability to refine the generated triples ensures that the final dataset is both complete and accurate, thereby providing valuable insights and reliable data for research applications.

Our framework's success in managing uncertainty demonstrates the potential of deep learning models in enhancing the quality of linked data, making it a valuable tool for future data management and research endeavors.

9.8 Conclusion

This chapter presented a unified framework for managing uncertainty in linked data, combining missing type detection, missing link detection, sameAs link detection, and erroneous triple detection. The approach effectively addressed data incompleteness and error detection.

The proposed framework advances the field by integrating multiple uncertainty-handling methods into a single solution. This approach enhances the quality and reliability of linked data, particularly in datasets like UniProt.

Addressing uncertainty in linked data is crucial for maintaining data quality and usability. The framework developed in this study offers a promising approach to managing uncertainties and improving the integrity of linked data.

General Conclusion

In the context of Semantic Web and the increasing complexity of Linked Data, ensuring data quality and completeness has become a pressing challenge. As organizations and applications increasingly rely on interconnected datasets, the need for effective methods to handle data incompleteness and errors has never been greater. The advancements in deep learning offer promising solutions to these challenges, enabling more accurate and comprehensive data management.

This thesis addresses critical challenges in Linked Data management by proposing and evaluating advanced deep learning techniques to handle data incompleteness and errors. The contributions outlined in this work significantly enhance the quality and usability of Linked Data through innovative methodologies and practical applications.

Our first major contribution is the development of LinkED-S2S, an encoder-decoder model with an attention mechanism designed for detecting missing links between RDF resources. This model offers a comprehensive solution for link discovery by uncovering hidden relationships and addressing data incompleteness across various datasets. The empirical results demonstrate its effectiveness compared to existing methods, providing a valuable tool for enriching Linked Data.

The second contribution involves predicting missing types for RDF entities using an encoder-decoder network with an attention mechanism. This approach improves dataset completeness by accurately inferring types based on associated predicates and objects. The application of this method to the DBpedia dataset highlights its potential to enhance type prediction and data quality.

Our third contribution focuses on sameAs link detection through a Siamese neural network model. This model improves the identification and alignment of similar entities, addressing the inherent heterogeneity and incompleteness in Linked Data. The results indicate significant improvements in link discovery accuracy and entity representation, advancing the state-of-the-art in sameAs link detection.

The fourth contribution introduces an advanced deep learning approach for error detection in Linked Data using LSTM networks. By capturing long-term dependencies and

semantic meanings within RDF triples, this model provides a scalable solution for identifying and removing erroneous triples. The approach successfully enhances dataset quality and reliability.

To validate these contributions, a case study on the UniProt dataset was conducted. This case study integrated the proposed methods for link detection, type prediction, sameAs link detection, and error correction, demonstrating their collective effectiveness in generating and validating triples. The results underscore the practical applicability of the proposed solutions in improving Linked Data management.

For future work, we plan to test our models on larger datasets to assess their scalability and performance. Additionally, we aim to explore the integration of natural language processing (NLP) techniques to further enhance the capabilities of our models, particularly in dealing with unstructured data and improving the accuracy of semantic link detection.

Moreover, we intend to incorporate various types of uncertainty, such as imprecision and contradiction, into our uncertainty management platform. This will involve developing mechanisms to handle these uncertainties effectively and to refine the overall performance of our models.

We also plan to develop practical applications that leverage the contributions proposed in this thesis, such as prediction tasks, recommendation systems, and knowledge graph enrichment. For instance, in the medical domain, addressing uncertainty is crucial. This includes focusing on managing errors, dealing with incomplete data, and predicting links between disparate pieces of information. By integrating these aspects into real-world applications, we can significantly improve decision-making processes and recommendations, particularly in environments where information is dynamic and continuously evolving.

The real-time handling of uncertainty aspects is becoming increasingly important, especially in fields requiring rapid decision-making based on constantly changing data. Enhancing our models' capabilities to detect and correct errors and predict relationships among data can greatly improve the quality of decisions and recommendations, ensuring that they are robust and reliable even in fast-paced scenarios.

Overall, this thesis makes significant strides in addressing the challenges of Linked Data, offering robust solutions to improve data completeness and accuracy. The proposed methodologies and their successful application to real-world datasets pave the way for further advancements in Linked Data processing and management.

Bibliography

[1] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. In *Linking the World's Information: Essays on Tim Berners-Lee's Invention of the World Wide Web*, pages 115–143. 2023.

[2] Fabien Gandon, Olivier Corby, and Catherine Faron-Zucker. *Le web sémantique: Comment lier les données et les schémas sur le web?* Dunod, 2012.

[3] Kenneth J Laskey and Kathryn B Laskey. Uncertainty reasoning for the world wide web: Report on the urw3-xg incubator group. *URSW*, 8:108–116, 2008.

[4] Dave Reynolds. Position paper: Uncertainty reasoning for linked data. In *Workshop*, volume 14. Citeseer, 2014.

[5] Mustafa Al-Bakri, Manuel Atencia, Jérôme David, Steffen Lalande, and Marie-Christine Rousset. Uncertainty-sensitive reasoning for inferring sameas facts in linked data. In *22nd european conference on artificial intelligence (ECAI)*, pages 698–706. IOS press, 2016.

[6] Heiko Paulheim and Christian Bizer. Type inference on noisy rdf data. In *International semantic web conference*, pages 510–525. Springer, 2013.

[7] Russa Biswas, Mehwish Alam, and Harald Sack. Madlink: Attentive multihop and entity descriptions for link prediction in knowledge graphs, 2021.

[8] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[9] Ivana Balažević, Carl Allen, and Timothy M Hospedales. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*, pages 553–565. Springer, 2019.

[10] Oussama Hamel and Messaouda Fareh. Deep sequence to sequence semantic embedding with attention for entity linking in context of incomplete linked data. *Engineering Applications of Artificial Intelligence*, 134:108689, 2024.

[11] Gerald Töpper, Magnus Knuth, and Harald Sack. Dbpedia ontology enrichment for inconsistency detection. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 33–40, 2012.

[12] Jürgen Umbrich, Katja Hose, Marcel Karnstedt, Andreas Harth, and Axel Polleres. Comparing data summaries for processing live queries over linked data. *World Wide Web*, 14(5):495–544, 2011.

[13] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The world-wide web. *Communications of the ACM*, 37(8):76–82, 1994.

[14] Boubker Sbihi. Web 2+: Vers une nouvelle version du web 2.0. *Journal of Information and Communication Technologies*, 35(2):12–24, 2009.

[15] James Hendler, Ora Lassila, and Tim Berners-Lee. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[16] Chin-I Lee, Tse-Chih Hsia, Hsiang-Chih Hsu, and Jing-Ya Lin. Ontology-based tourism recommendation system. In *2017 4th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 376–379. IEEE, 2017.

[17] Mark Needleman. The w3c semantic web activity. *Serials Review*, 29(1):63–64, 2003.

[18] Mussab Zneika. *Querying semantic web/linked data graphs using summarization.* PhD thesis, Université de Cergy Pontoise, 2019.

[19] Sid-Ali Chikh. L'exploitation des vocabulaires dans un portail web sémantique. le cas du portail thématique plan4learning de l'iipe-unesco.

[20] World Wide Web Consortium et al. Rdf 1.1 primer. 2014.

[21] Eric Prud'hommeaux, Andy Seaborne, et al. Sparql query language for rdf. w3c recommendation (2008), 2017.

[22] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[23] Michel Klein. Xml, rdf, and relatives. *IEEE Intelligent Systems*, 16(2):26–28, 2001.

[24] OWL Working Group et al. Owl 2 web ontology language document overview: W3c recommendation 27 october 2009. 2009.

[25] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI global, 2011.

[26] Tim Berners-Lee. Linked data - design issues. http://www.w3.org/DesignIssues/LinkedData.html, 2006. Accessed: 2022-05-29.

[27] Oussama Hamel and Messaouda Fareh. Missing types prediction in linked data using deep neural network with attention mechanism: Case study on dbpedia and uniprot datasets. In *Special Sessions in the Advances in Information Systems and Technologies Track of the Conference on Computer Science and Intelligence Systems*, pages 212–231. Springer, 2022.

[28] Colin Wilcox, Soufiene Djahel, and Vasileios Giagos. Identifying the main causes of medical data incompleteness in the smart healthcare era. In *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6. IEEE, 2021.

[29] Ann-Kristin Kock-Schoppenhauer, Christian Kamann, Hannes Ulrich, Petra Duhm-Harbeck, and Josef Ingenerf. Linked data applications through ontology based data access in clinical research. In *Informatics for Health: Connected Citizen-Led Wellness and Population Health*, pages 131–135. IOS Press, 2017.

[30] Zahid Raza, Khalid Mahmood, and Nosheen Fatima Warraich. Application of linked data technologies in digital libraries: a review of literature. *Library Hi Tech News*, 36(3):9–12, 2019.

[31] Marie-Christine Rousset. Reasoning on web data semantics. In *Seminar in College de France*, 2012.

[32] Didier Dubois, Henri Prade, and Agnes Rico. Representing qualitative capacities as families of possibility measures. *International Journal of Approximate Reasoning*, 58:3–24, 2015.

[33] Thierry Denœux, Didier Dubois, and Henri Prade. Representations of uncertainty in artificial intelligence: Probability and possibility. *A Guided Tour of Artificial In-*

telligence Research: Volume I: Knowledge Representation, Reasoning and Learning, pages 69–117, 2020.

[34] Deyi Li and Yi Du. *Artificial intelligence with uncertainty*. CRC press, 2017.

[35] Hamid Mcheick and Atif Farid Mohammad. The evident use of evidence theory in big data analytics using cloud computing. In *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6. IEEE, 2014.

[36] Daniel P Thunnissen. Uncertainty classification for the design and development of complex systems. In *Proceedings of the 3rd Annual Predictive Methods Conference, Veros Software, Santa Ana, CA*, pages 1–16, 2003.

[37] Joseph Aguilar-Martin. La logique floue et ses applications industrielles. *Quaderni*, 25(1):75–86, 1995.

[38] Piero P. Bonissone and Richard M. Tong. Reasoning with uncertainty in expert systems. *International journal of man-machine studies*, 22(3):241–250, 1985.

[39] Amihai Motro and Philippe Smets. *Uncertainty management in information systems: from needs to solutions*. Springer Science & Business Media, 1996.

[40] Jing Xiao. *Gestion des incertitudes dans le processus de développement de systèmes complexes*. PhD thesis, Institut National Polytechnique de Toulouse-INPT, 2009.

[41] Didier Dubois. Uncertainty theories: a unified view. In *8th International Workshop on Reliable Engineering Computing (REC 2018)*, 2018.

[42] Lotfi Asker Zadeh, George J Klir, and Bo Yuan. *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers*, volume 6. World scientific, 1996.

[43] Harry Halpin and Patrick J Hayes. When owl: sameas isn't the same: An analysis of identity links on the semantic web. *LDOW*, 628, 2010.

[44] Carlo Batini, Monica Scannapieco, Anisa Rula, Andrea Maurino, and Carlo Batini. Data quality issues in linked open data. *Data and information quality: Dimensions, principles and techniques*, pages 87–112, 2016.

[45] Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann. Crowdsourcing linked data quality assessment. In *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II 12*, pages 260–276. Springer, 2013.

[46] Anisa Rula, Amrapali Zaveri, et al. Methodology for assessment of linked data quality. In *CEUR Workshop Proceedings*, volume 1215. CEUR-WS, 2014.

[47] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

[48] E Heer. The age of intelligent machines: By raymond kurzweil. mit press, cambridge, mass., 1990, 1993.

[49] Fernando Santos Osório. *INSS: un système hybride neuro-symbolique pour l'apprentissage automatique constructif*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 1998.

[50] Herbert A Simon. Why should machines learn? In *Machine learning*, pages 25–37. Elsevier, 1983.

[51] Yung-Yao Chen, Yu-Hsiu Lin, Chia-Ching Kung, Ming-Han Chung, and I-Hsuan Yen. Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes. *Sensors*, 19(9):2047, 2019.

[52] Yanjie Liang, Zhiyong Gao, Jianmin Gao, Rongxi Wang, Qianqian Liu, and Yahui Cheng. A new method for multivariable nonlinear coupling relations analysis in complex electromechanical system. *Applied soft computing*, 94:106457, 2020.

[53] David E Rumelhart, Paul Smolensky, James L McClelland, and G Hinton. Sequential thought processes in pdp models. *Parallel distributed processing: explorations in the microstructures of cognition*, 2:3–57, 1986.

[54] Sneha Padhiar, Manan Patel, Kalpit Patel, and Rishi Shah. Machine learning algorithms for the detection of email spam based on python implementation. In *International Conference on Innovations and Advances in Cognitive Systems*, pages 44–52. Springer, 2024.

[55] Rui Ren, JieChao Cheng, Hao Shi, Lei Lei, Wuming Zhou, Guofeng Wang, and Congwu Li. Failure characterization based on lstm networks for bluegene/l system logs. In *Intelligent Computing and Block Chain: First BenchCouncil International Federated Conferences, FICC 2020, Qingdao, China, October 30–November 3, 2020, Revised Selected Papers 1*, pages 123–133. Springer, 2021.

[56] Konstantinos Stergiou and Theodoros E Karakasidis. Application of deep learning and chaos theory for load forecasting in greece. *Neural Computing and Applications*, 33(23):16713–16731, 2021.

[57] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[58] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[59] Mohammad Khajehzadeh, Suraparb Keawsawasvong, Viroon Kamchoom, Shi Chao, and Alimorad Khajehzadeh. Developing an effective optimized machine learning approaches for settlement prediction of shallow foundation. *Heliyon*, 2024.

[60] Wanping Li, Bixuan Wang, Jingcun Liu, Guogang Zhang, and Jianhua Wang. Igbt aging monitoring and remaining lifetime prediction based on long short-term memory (lstm) networks. *Microelectronics Reliability*, 114:113902, 2020.

[61] Xiaosheng Peng, Hongyu Wang, Jianxun Lang, Wenze Li, Qiyou Xu, Zuowei Zhang, Tao Cai, Shanxu Duan, Fangjie Liu, and Chaoshun Li. Ealstm-qr: Interval wind-power prediction model based on numerical weather prediction and deep learning. *Energy*, 220:119692, 2021.

[62] Sandeep Kumar and Arun Solanki. An abstractive text summarization technique using transformer model with self-attention mechanism. *Neural Computing and Applications*, 35(25):18603–18622, 2023.

[63] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[64] Jian-wei LIU, Jun-wen LIU, and Xiong-lin LUO. Research progress in attention mechanism in deep learning. *Chinese Journal of Engineering*, 43(11):1499–1511, 2021.

[65] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[66] Tomáš Kliegr and Ondřej Zamazal. Lhd 2.0: A text mining approach to typing entities in knowledge graphs. *Journal of Web Semantics*, 39:47–61, 2016.

[67] Russa Biswas, Radina Sofronova, Mehwish Alam, and Harald Sack. Entity type prediction in knowledge graphs using embeddings. *arXiv preprint arXiv:2004.13702*, 2020.

[68] Molood Barati, Quan Bai, and Qing Liu. An entropy-based class assignment detection approach for rdf data. In *Pacific rim international conference on artificial intelligence*, pages 412–420. Springer, 2018.

[69] Yaroslav Nechaev, Francesco Corcoglioniti, and Claudio Giuliano. Type prediction combining linked open data and social media. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1033–1042, 2018.

[70] Russa Biswas, Rima Türker, Farshad Bakhshandegan Moghaddam, Maria Koutraki, and Harald Sack. Wikipedia infobox type prediction using embeddings. In *DL4KGS@ ESWC*, pages 46–55, 2018.

[71] Nandana Mihindukulasooriya and Mariano Rico. Type prediction of rdf knowledge graphs using binary classifiers with structural data. In *International Conference on Web Engineering*, pages 279–287. Springer, 2018.

[72] Xiang Zhang, Erjing Lin, and Siyao Pi. Predicting object types in linked data by text classification. In *2017 Fifth International Conference on Advanced Cloud and Big Data (CBD)*, pages 391–396. IEEE, 2017.

[73] Liya Kondratyeva, Irina Alekseenko, Igor Chernov, and Eugene Sverdlov. Data incompleteness may form a hard-to-overcome barrier to decoding life's mechanism. *Biology*, 11(8):1208, 2022.

[74] Li Ding, Joshua Shinavier, Zhenning Shangguan, and Deborah L McGuinness. Sameas networks and beyond: analyzing deployment status and implications of owl: sameas in linked data. In *The Semantic Web–ISWC 2010: 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I 9*, pages 145–160. Springer, 2010.

[75] Joe Raad, Nathalie Pernelle, and Fatiha Saïs. Detection of contextual identity links in a knowledge base. In *Proceedings of the knowledge capture conference*, pages 1–8, 2017.

[76] Pavel Shvaiko and Jérôme Euzenat. Ontology matching: state of the art and future challenges. *IEEE Transactions on knowledge and data engineering*, 25(1):158–176, 2011.

[77] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F Cruz, and Francisco M Couto. The agreementmakerlight ontology matching system. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences: Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings*, pages 527–541. Springer, 2013.

[78] Fatiha Saïs. *Knowledge Graph Refinement: Link Detection, Link Invalidation, Key Discovery and Data Enrichment.* PhD thesis, Université Paris Sud, 2019.

[79] Fabian M Suchanek, Serge Abiteboul, and Pierre Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *arXiv preprint arXiv:1111.7164*, 2011.

[80] Marie Destandau and Jean-Daniel Fekete. The missing path: Analysing incompleteness in knowledge graphs. *Information Visualization*, 20(1):66–82, 2021.

[81] Fariz Darari, Werner Nutt, Giuseppe Pirrò, and Simon Razniewski. Completeness management for rdf data sources. *Semantic Web*, 9(6):745–774, 2018.

[82] Avicenna Wisesa, Fariz Darari, Adila Krisnadhi, Werner Nutt, and Simon Razniewski. Wikidata completeness profiling using prowd. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 123–130, 2019.

[83] Radityo Eko Prasojo, Fariz Darari, Simon Razniewski, and Werner Nutt. Managing and consuming completeness information for wikidata using cool-wd. *COLD@ ISWC*, 1666, 2016.

[84] John P McCrae and Paul Buitelaar. Linking datasets using semantic textual similarity. *Cybernetics and information technologies*, 18(1):109–123, 2018.

[85] Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, Kleanthi Georgala, Mofeed Mohamed Hassan, Kevin Dreßler, Klaus Lyko, Daniel Obraczka, and Tommaso Soru. Limes: A framework for link discovery on the semantic web. *KI-Künstliche Intelligenz*, 35(3):413–423, 2021.

[86] Gabriela Oliveira Mota Da Silva, Paulo Roberto De Souza, and Frederico Araújo Durão. Hsld: a hybrid similarity measure for linked data resources. *International Journal of Metadata, Semantics and Ontologies*, 14(1):16–25, 2020.

[87] Armando Barbosa, Ig I Bittencourt, Sean W Siqueira, Diego Dermeval, and Nicholas JT Cruz. A context-independent ontological linked data alignment ap-

proach to instance matching. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 18(1):1–29, 2022.

[88] Guangyuan Piao and John G Breslin. Measuring semantic distance for linked open data-enabled recommender systems. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 315–320, 2016.

[89] Claudia d'Amato, Pierpaolo Masella, and Nicola Fanizzi. An approach based on semantic similarity to explaining link predictions on knowledge graphs. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 170–177, 2021.

[90] Manel Achichi, Zohra Bellahsene, Mohamed Ben Ellefi, and Konstantin Todorov. Linking and disambiguating entities across heterogeneous rdf graphs. *Journal of Web Semantics*, 55:108–121, 2019.

[91] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121*, 2017.

[92] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[93] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[94] Jeremy Debattista, Christoph Lange, and Sören Auer. A preliminary investigation towards improving linked data quality using distance-based outlier detection. In *Semantic Technology: 6th Joint International Conference, JIST 2016, Singapore, Singapore, November 2-4, 2016, Revised Selected Papers 6*, pages 116–124. Springer, 2016.

[95] André Melo and Heiko Paulheim. Detection of relation assertion errors in knowledge graphs. In *Proceedings of the Knowledge Capture Conference*, pages 1–8, 2017.

[96] Huiying Li, Yuanyuan Li, Feifei Xu, and Xinyu Zhong. Probabilistic error detecting in numerical linked data. In *International Conference on Data Management in Cloud, Grid and P2P Systems*, pages 61–75. Springer, 2015.

[97] Heiko Paulheim. Identifying wrong links between datasets by multi-dimensional outlier detection. In *WoDOOM*, pages 27–38. Citeseer, 2014.

[98] Shuangyan Liu, Mathieu d'Aquin, and Enrico Motta. Measuring accuracy of triples in knowledge graphs. In *Language, Data, and Knowledge: First International Conference, LDK 2017, Galway, Ireland, June 19-20, 2017, Proceedings 1*, pages 343–357. Springer, 2017.

[99] Mariano Rico, Nandana Mihindukulasooriya, Dimitris Kontokostas, Heiko Paulheim, Sebastian Hellmann, and Asunción Gómez-Pérez. Predicting incorrect mappings: a data-driven approach applied to dbpedia. In *Proceedings of the 33rd annual ACM symposium on applied computing*, pages 323–330, 2018.

[100] Dominik Wienand and Heiko Paulheim. Detecting incorrect numerical data in dbpedia. In *The Semantic Web: Trends and Challenges: 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings 11*, pages 504–518. Springer, 2014.

[101] Lihua Zhao, Rumana Ferdous Munne, Natthawut Kertkeidkachorn, and Ryutaro Ichise. Missing rdf triples detection and correction in knowledge graphs. In *Semantic Technology: 7th Joint International Conference, JIST 2017, Gold Coast, QLD, Australia, November 10-12, 2017, Proceedings 7*, pages 164–180. Springer, 2017.

[102] Petar Ristoski and Heiko Paulheim. Semantic web in data mining and knowledge discovery: A comprehensive survey. *Journal of Web Semantics*, 36:1–22, 2016.

[103] Oussama Hamel and Messaouda Fareh. Encoder-decoder neural network with attention mechanism for types detection in linked data. In *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*, pages 733–739. IEEE, 2022.

[104] Feiwei Qin, Nannan Gao, Yong Peng, Zizhao Wu, Shuying Shen, and Artur Grudtsin. Fine-grained leukocyte classification with deep residual learning for microscopic images. *Computer methods and programs in biomedicine*, 162:243–252, 2018.

[105] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[106] Davide Chicco. Siamese neural networks: An overview. *Artificial neural networks*, pages 73–94, 2021.

[107] Oussama Hamel and Messaouda Fareh. Deep learning-based erroneous data detection in linked data context using lstm and embeddings. In *2024 International Conference on Advances in Electrical and Communication Technologies (ICAECOT)*, pages 1–6. IEEE, 2024.

[108] Oussama Hamel and Messaouda Fareh. A deep learning-based framework for handling incompleteness and detecting errors in linked data applied to the uniprot dataset. In *2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP)*, pages 1–8. IEEE, 2024.

[109] Bin Ding, Huimin Qian, and Jun Zhou. Activation functions and their characteristics in deep neural networks. In *2018 Chinese control and decision conference (CCDC)*, pages 1836–1841. IEEE, 2018.

[110] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.

[111] Jingcheng Du, Qingyu Chen, Yifan Peng, Yang Xiang, Cui Tao, and Zhiyong Lu. Ml-net: multi-label classification of biomedical texts with deep neural networks. *Journal of the American Medical Informatics Association*, 26(11):1279–1285, 2019.

[112] Nick Craswell. *Mean Reciprocal Rank*, pages 1703–1703. Springer US, Boston, MA, 2009.

Appendix A

Activation Functions

A.1 Binary Activation Function

The binary activation function uses a threshold value S to determine neuron activation. If Y exceeds S, the neuron is activated, which can be useful in binary classification tasks [109]. The graph of this function is shown in Figure A.1:

$$f(Y) = \begin{cases} 1 & \text{if } Y \geq S \\ 0 & \text{if } Y < S \end{cases}$$

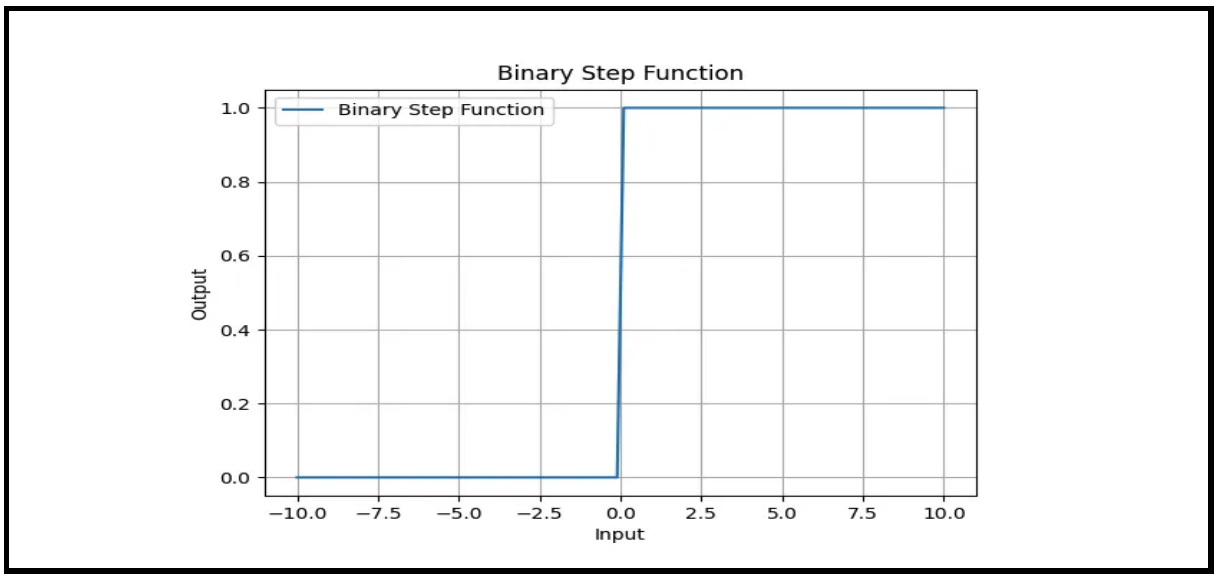


Figure A.1: Graphical Representation of the Binary Function

A.2 Linear Activation Function

The linear activation function is a simple, proportional function, ideal for scenarios where the relationship between input and output is straightforward [109]. Figure A.2 illustrates the linear activation function. The output is directly proportional to the input:

$$f(z) = z$$

While binary and linear functions are suited to specific, well-defined cases, most real-world data, especially Linked Data, requires more complex, non-linear functions to manage uncertainty and achieve accurate results.

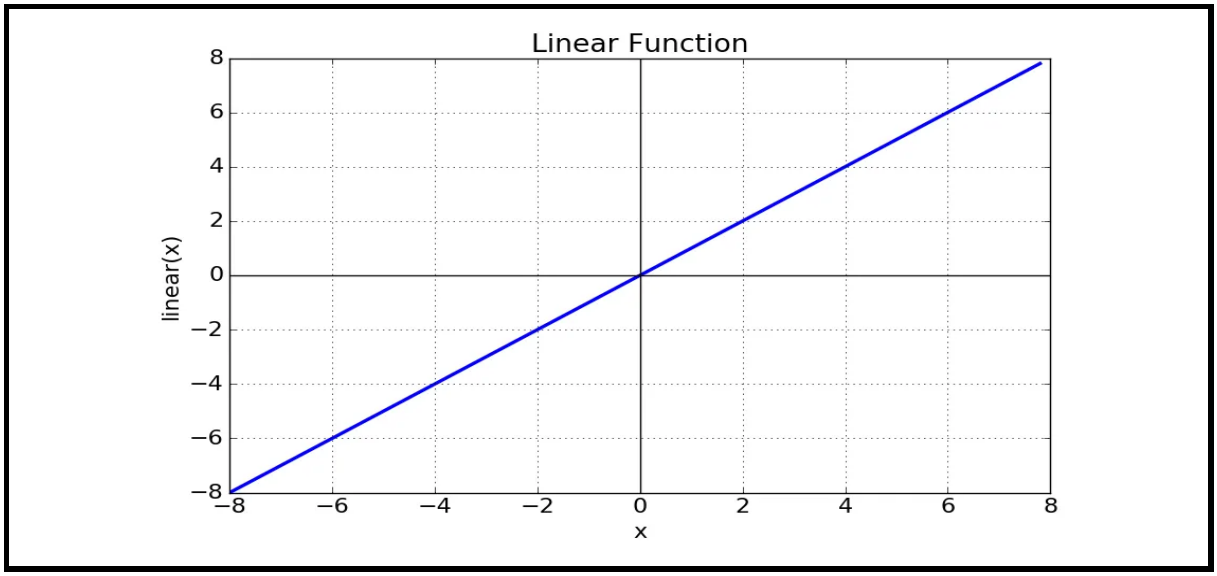


Figure A.2: Graphical Representation of the Linear Activation Function

A.3 Non-Linear Activation Functions

Non-linear activation functions are essential for neural networks to handle non-linearly separable data, which is common in complex datasets. These functions introduce non-linearity into the model, enabling the network to learn from intricate patterns and relationships. Several non-linear activation functions are widely recognized and utilized in the literature:

- Sigmoid Function [109]: The sigmoid function is frequently used in models requiring output in the range [0, 1], making it suitable for tasks such as probability prediction. The function is defined as:

$$g(z) = \frac{1}{1 + e^{-z}}, \quad z \in \mathbb{R}$$

It smoothly maps any input to a value between 0 and 1, as shown in Figure A.3. However, it is prone to the vanishing gradient problem, where gradients become very small during backpropagation, slowing down the learning process.

- Tanh Function (Hyperbolic Tangent) [109]: The tanh function, which is a scaled version of the sigmoid function, maps input values to a range between -1 and 1, providing stronger gradients for negative values. It is expressed as:

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Figure A.4 illustrates this function. While it centers the output around zero, making optimization easier, it also suffers from the vanishing gradient problem.

- ReLU Function (Rectified Linear Unit) [109]: The ReLU function is currently the most popular activation function in deep learning due to its simplicity and efficiency. It is defined as:

$$f(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$

As depicted in Figure A.5, ReLU introduces sparsity by setting negative values to zero and accelerates convergence in training deep networks. However, it can lead to exploding gradients and is not well-suited for recurrent neural networks due to these large gradient values.

- Softmax Function [109]: The softmax function is used in the output layer of neural networks for multi-class classification problems. It converts a vector of values into a probability distribution, where each value is between 0 and 1, and their sum equals 1. The softmax function is given by:

$$S(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Here, z_j represents the j-th element of the input vector, and K is the number of classes in the classification task. This function is particularly useful for determining the most probable class among multiple possibilities.

The advancement of these activation functions has significantly impacted the evolution of neural networks, particularly in the transition from traditional models to Deep Learning, where networks can learn more complex and hierarchical representations of data.

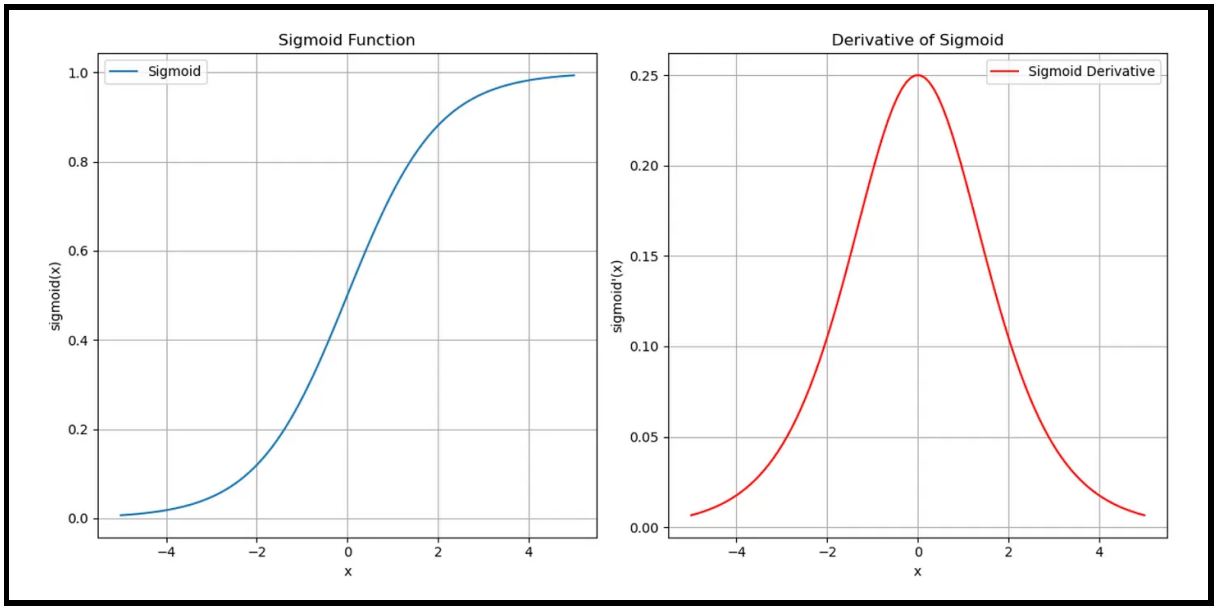


Figure A.3: Graphical Representation of the Sigmoid Function

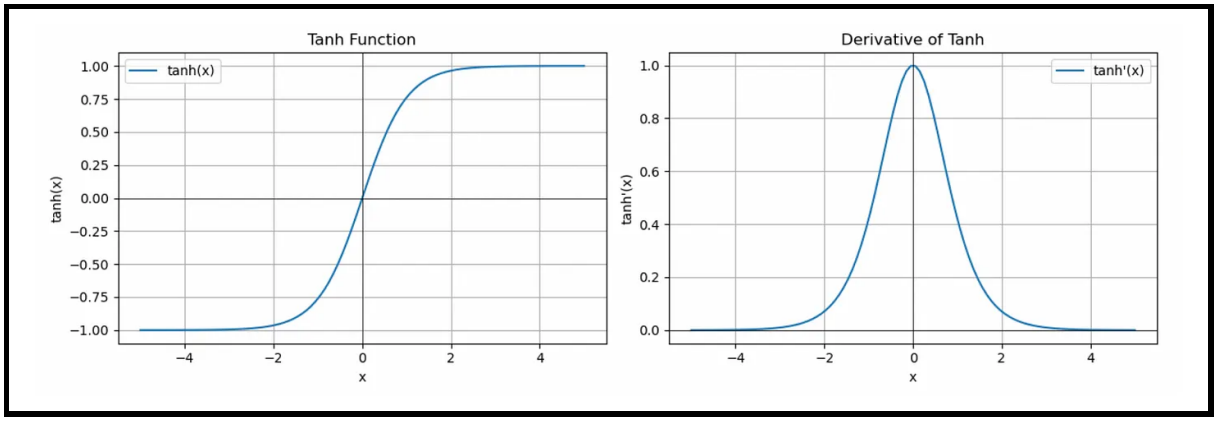


Figure A.4: Graphical Representation of the Tanh Function

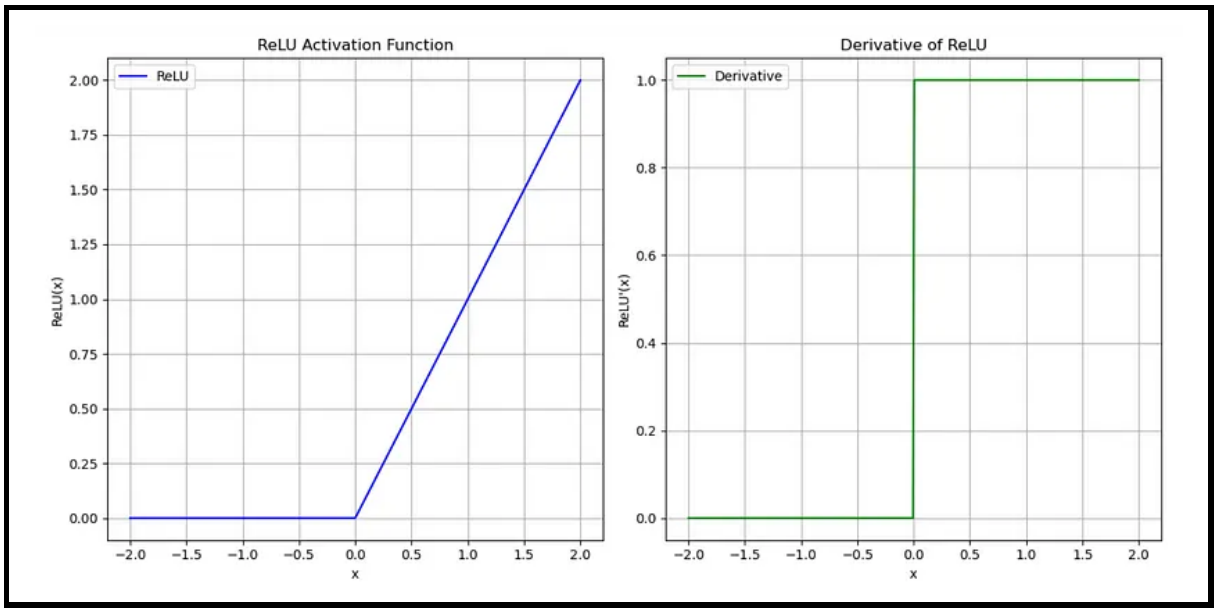


Figure A.5: Graphical Representation of the ReLU Function

Appendix B

Datasets and Metrics

B.1 Datasets

In the context of Linked Data, various datasets play a crucial role in providing structured and interlinked information across different domains. These datasets offer a foundation for knowledge representation and can significantly enhance the accuracy and richness of information available through semantic web applications. Below is an overview of notable RDF datasets used in Linked Data:

- DBpedia[1] is a prominent dataset that converts Wikipedia's content into RDF, making it accessible in a structured format. DBpedia connects with other datasets, such as Geonames, allowing applications to integrate and extract comprehensive knowledge from multiple sources. This interconnectedness improves the quality and depth of user experiences by leveraging diverse datasets.

- YAGO3-10 is a benchmark dataset designed for knowledge base completion, comprising entities interconnected by at least ten distinct relations. This dataset is instrumental for evaluating knowledge base completion algorithms, providing a robust framework for assessing the effectiveness of various models.

- WN18 includes 18 relations derived from WordNet, covering approximately 41,000 synsets. An issue identified with this dataset is the presence of many test triplets also found in the training set, either directly or through inverse relations. To mitigate this problem, the updated dataset, WN18RR, has been introduced, addressing the limitations of the original dataset.

[1] https://www.DBpedia.org/

- FB15k encompasses entity pairs and knowledge base relation triples from Freebase. The dataset was later refined to create FB15k-237 to address the problem of inverse relation leakage, where information from training data inadvertently influences testing and validation. This adjustment ensures a more accurate evaluation of link prediction models by eliminating such leakage.

These benchmark datasets are accessible for research and application development at the following link[2].

B.2 Metrics

In evaluating model performance, particularly in binary classification and information retrieval, various metrics are employed to gauge effectiveness. These metrics provide insights into the model's ability to make accurate predictions and retrieve relevant information. Below is an overview of key metrics used in these domains:

- Precision [110,111] quantifies the accuracy of positive predictions made by a model. It is defined as the proportion of true positive predictions among all positive predictions. The formula for precision is given by:

$$\text{Precision} = \frac{1}{p}\sum_{i=1}^{p}\left(\frac{TP}{TP+FP}\right) \tag{B.1}$$

  where TP denotes true positives, FP denotes false positives, and p represents the number of positive predictions.

- Recall [110, 111], also known as sensitivity or the true positive rate, measures the proportion of true positive predictions out of all actual positive instances in the dataset. It reflects the model's ability to identify all positive instances. The formula for recall is:

$$\text{Recall} = \frac{1}{p}\sum_{i=1}^{p}\left(\frac{TP}{TP+FN}\right) \tag{B.2}$$

  where FN denotes false negatives and p represents the total number of positive instances.

[2]https://codeload.github.com/villmow/datasets_knowledge_embedding/zip/refs/heads/master

- F-score, or F1-score [110, 111], is the harmonic mean of precision and recall, providing a balanced measure that incorporates both metrics. It is particularly useful when there is a need to balance precision and recall. The formula for the F-score is:

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{B.3}$$

  These metrics are essential for evaluating binary classification models, assessing their ability to accurately classify positive instances (precision), capture all relevant positive instances (recall), and balance the two (F-score).

- Mean Reciprocal Rank (MRR) [112] is a metric used to evaluate the effectiveness of search systems. It computes the inverse of the rank at which the first relevant document is found, averaged across multiple queries. The formula for MRR is:

$$\text{MRR} = \frac{1}{Q} \sum_{i=1}^{Q} \frac{1}{\text{rank}_i} \tag{B.4}$$

  where rank_i is the position of the relevant document for the i-th query, and Q represents the number of queries.

- Hits@k measures the fraction of true entities that appear within the top k positions of the ranked results, providing insight into the effectiveness of the ranking system in retrieving relevant items.