Geospatial Link Discovery With Human In The Loop

Abdullah Fathi Ahmed Ahmed

August 12, 2024

Paderborn University

Faculty for Computer Science, Electrical Engineering and Mathematics
Department of Computer Science
Data Science Research Group (DICE)

PhD DISSERTATION

Geospatial Link Discovery With Human In The Loop

Abdullah Fathi Ahmed Ahmed

1. Reviewer	Prof. Dr. Axel-Cyrille Ngonga Ngomo Department of Computer Science Paderborn University
2. Reviewer	Prof. Dr. Alsayed Algergawy Department of Computer Science University of Passau
Supervisors	Prof. Dr. Axel-Cyrille Ngonga Ngomo and Dr. Mohamed Ahmed Sherif

August 12, 2024

Abdullah Fathi Ahmed Ahmed
Geospatial Link Discovery With Human In The Loop
PhD DISSERTATION, August 12, 2024
Reviewers: Prof. Dr. Axel-Cyrille Ngonga Ngomo and Prof. Dr. Alsayed Algergawy
Supervisors: Prof. Dr. Axel-Cyrille Ngonga Ngomo and Dr. Mohamed Ahmed Sherif

Paderborn University
Data Science Research Group (DICE)
Department of Computer Science
Faculty for Computer Science, Electrical Engineering and Mathematics
Warburger Straße 100
33098 Paderborn

Abstract

The increased availability of heterogeneous Knowledge Graphs (KGs) has unlocked numerous opportunities in data-driven applications, such as information retrieval, Natural Language Processing (NLP), and geospatial analysis. However, the potential of these KGs remains largely untapped due to their limited interlinking. This thesis addresses critical research gaps in the realm of Link Discovery (LD) over KGs particularly focusing on geospatial KGs. Three key research gaps are identified: scalability (**Research Gap 1**), the absence of holistic models for Linked Open Data (LOD) (**Research Gap 2**), and the need for explainability in LD with the human in the loop (**Research Gap 3**).

To address **Research Gap 1**, we propose a novel algorithm called *LineSimp*. In the *LineSimp* , we discover the topological relations between geospatial Resource Description Framework (RDF) resources using a simplified version of such geospatial RDF resources. Our findings show an average loss of 15% in F-measure on the original data, but a significant speedup of up to $67\times$ when applied to simplified geometries. The *Intersection Matrix Approach* offers a scalable solution in the LD over geospatial KGs with a speed increase of up to 35% in computation time. In addition, we introduce COBALT. COBALT combines content measures with R-tree indexing. Content measures are based on the area, diagonal, and distance of the Minimum Bounding Boxes (MBBs) of polygons, which speeds up the process while allowing for further precision.

To address **Research Gap 2**, we introduce NELLIE, a modular pipeline architecture. NELLIE not only identifies relevant KGs for linkage but also successfully integrates them, contributing to a more unified and complete (KG). Our approach shows a measurable impact on link prediction and KG completeness, marking a milestone for 24/7 linking and holistic LOD models. When tackling **Research Gap 3**, we offer innovative solutions to improve explainability, including the conversion of complex Link Specifications (LSs) into natural language. Our template-based and neural-based verbalization methods significantly improve user comprehension without sacrificing the richness of the underlying LSs.

In summary, this thesis makes groundbreaking contributions to the domain of LD over geospatial KGs. It introduces scalable, effective, and explainable methodologies that have extensive applications in real-time systems, Knowledge Graph (KG) integration, and the broader realm of explainable AI.

Abstract (German language)

Die rasche Ausweitung heterogener Wissensgraphen (KGs) hat zahlreiche Möglichkeiten in datengetriebenen Bereichen wie der Informationssuche, Natural Language Processing (NLP), und geografischen Analyse eröffnet. Das Potenzial dieser KGs bleibt jedoch aufgrund der begrenzten Vernetzung zwischen ihnen weitgehend ungenutzt. Diese Dissertation befasst sich mit kritischen Forschungslücken im Bereich von Link Discovery (LD) über KGs, insbesondere mit einem Fokus auf geografische KGs. Drei Hauptforschungslücken werden identifiziert: Skalierbarkeitsprobleme (**Forschungslücke 1**), das Fehlen ganzheitlicher Modelle für Linked Open Data (LOD) (**Forschungslücke 2**) und die Notwendigkeit für Erklärbarkeit in der LD mit dem Menschen in der Schleife (**Forschungslücke 3**).

Für **Forschungslücke 1** schlagen wir neue Algorithmen vor, insbesondere den *Line Simplification Approach*. Im *Line Simplification Approach* entdecken wir den topologischen Beziehungen zwischen geografischen RDF-Ressourcen durch eine vereinfachte Version dieser geografischen RDF-Ressourcen. Unsere Ergebnisse zeigen einen durchschnittlichen Verlust von 15% beim F-Maß auf den Originaldaten, aber eine signifikante Beschleunigung um bis zu 67×, wenn sie auf vereinfachte Geometrien angewendet werden. Der *Intersection Matrix Approach* bietet eine skalierbare Lösung in LD über geografischen KGs mit einer Geschwindigkeitssteigerung von bis zu 35% bei der Berechnungszeit. Darüber hinaus führen wir Cobalt ein. Cobalt kombiniert die die Inhaltsähnlichkeitsmessungen mit der R-Baum-Indexierung. die Inhaltsähnlichkeitsmessungen basieren auf der Fläche, Diagonale und Entfernung der Minimum Bounding Boxes (MBBs) von Polygonen, was den Prozess beschleunigt, jedoch weitere Präzision ermöglicht.

Um **Forschungslücke 2** zu beheben, führen wir Nellie ein, eine modulare Pipeline-Architektur. Nellie identifiziert nicht nur relevante KGs für die Verlinkung, sondern integriert sie auch erfolgreich, was zu einem einheitlicheren und vollständigeren KG beiträgt. Unser Ansatz zeigt eine messbare Auswirkung auf die Linkvorhersage und die Vollständigkeit des KG, und markiert einen Meilenstein für 24/7-Verlinkung und ganzheitliche LOD-Modelle. Bei der Bewältigung von **Forschungslücke 3** bieten

wir innovative Lösungen zur Verbesserung der Erklärbarkeit, einschließlich der Umwandlung komplexer Link Specifications (LSs) in natürliche Sprache. Unsere templatebasierten und neuralbasierten Verbalisierungsmethoden verbessern das Benutzerverständnis erheblich, ohne die Reichhaltigkeit der zugrundeliegenden LSs zu opfern.

Zusammenfassend leistet diese Dissertation bahnbrechende Beiträge zum Bereich LD über geografische KGs. Sie schlägt skalierbare, effektive und erklärbar Methodologien vor, die umfangreiche Anwendungen in Echtzeitsystemen, der Integration von KGs und dem breiteren Bereich der erklärbar KI haben.

Acknowledgements

First and foremost, I extend my heartfelt appreciation to each individual who has been a cornerstone in the foundation of my PhD journey; your support has been the *sine qua non* of my achievements.

I reserve special commendation for my esteemed supervisor, Prof. Dr. Axel-Cyrille Ngonga Ngomo, and my dedicated advisor, Dr. Mohamed Ahmed Sherif. Their mentorship has transcended the conventional bounds of academic supervision, availing me not only a platform to grow professionally but also the latitude to tailor my research in alignment with my own intellectual curiosities. Their unwavering belief in my capabilities and consistent encouragement have been instrumental since the inception of my doctoral quest. My indebtedness for their sage guidance is profound and everlasting.

My sincere thanks are extended to my second reviewer, Prof. Dr.-Ing. Alsayed Algergawy, and to the distinguished members of my thesis committee, Dr. Mohamed Sherif, Prof. Dr.-Ing. Roman Dumitrescu, and Prof. Dr. Stefan Sauer. Their perceptive critiques and constructive observations were invaluable in refining my research and elevating it to the academic standards it has achieved. I am profoundly grateful to Dr. Pamela Heidi Douglas for her exceptional proofreading skills and insightful suggestions. Her native expertise has been instrumental in refining this work to its current standard, enhancing its readability and scholarly precision.

I am truly fortunate to have been part of the DICE research group, a vibrant community teeming with intellectual enthusiasm and collaboration. My experience in Paderborn has been rendered memorable by the solidarity, mutual support, and friendships I have forged there.

Turning to the bedrock of my existence, my family, I wish to express my deepest affection and gratitude. To my father Prof. Dr. Fathi, whose work ethic and resilience have always served as a guiding light; to my mother, whose unconditional love and wisdom have been my constant source of strength; and to my brothers Dr. Okba, Dr. Otba, and Dr. Qutaibah, who have been both confidants and pillars of support; I owe each of you an incalculable debt of gratitude. Your unwavering faith in me and your emotional sustenance have shaped not just this thesis but the person I am today.

I am also indebted to Dr. Ali Eldahie, whose unwavering support has been a source of strength and encouragement for me. In sum, the success of my doctoral journey is a tribute to the collective efforts, enduring support, and boundless love from each of you. My deepest and most heartfelt thanks to all.

Contents

Selected Publications

1. Abdullah Fathi Ahmed et al. “NELLIE: Never-Ending Linking for Linked Open Data”. In: *IEEE Access* 11 (2023), pp. 84957–84973

2. Abdullah Fathi Ahmed et al. “Explainable Integration of Knowledge Graphs Using Large Language Models”. In: *Natural Language Processing and Information Systems*. Ed. by Elisabeth Métais et al. Cham: Springer Nature Switzerland, 2023, pp. 124–139

3. Alexander Becker et al. “COBALT: A Content-Based Similarity Approach for Link Discovery over Geospatial Knowledge Graphs”. In: *SEMANTiCS*. 2023

4. Abdullah Fathi Ahmed et al. “Multilingual Verbalization and Summarization for Explainable Link Discovery”. In: *Data and Knowledge Engineering* 133 (2021), p. 101874

5. Mohamed Ahmed Sherif et al. “IngridKG: A FAIR Knowledge Graph of Graffiti”. In: *Scientific Data* (2023)

6. Abdullah Fathi Ahmed et al. “LSVS: Link Specification Verbalization and Summarization”. In: *International Conference on Applications of Natural Language to Information Systems*. Springer. 2019, pp. 66–78

7. Abdullah Fathi Ahmed et al. “Do your Resources Sound Similar? On the Impact of Using Phonetic Similarity in Link Discovery”. In: *K-CAP 2019: Knowledge Capture Conference*. 2019

8. Abdullah Fathi Ahmed et al. “Radon2: a buffered-intersection matrix computing approach to accelerate link discovery over geo-spatial rdf knowledge bases”. In: *Ontology Matching: OM-2018: Proceedings of the ISWC Workshop*. 2018, p. 197

9. Abdullah Fathi Ahmed et al. “On the Effect of Geometries Simplification on Geospatial Link Discovery”. In: *Procedia Computer Science* 137 (2018). Proceedings of the 14th International Conference on Semantic Systems 10th – 13th of September 2018 Vienna, Austria, pp. 139–150

List of Acronomy

LD	Link Discovery
RDF	Resource Description Framework
LOD	Linked Open Data
KGs	Knowledge Graphs
KG	Knowledge Graph
NLP	Natural Language Processing
SW	Semantic Web
LLM	Large Language Model
RDFS	Resource Description Framework Schema
TTL	Terse RDF Triple Language
OWL	Ontology Web Language
RNN	Recurrent neural network
BERT	Bidirectional Encoder Representations from Transformers
NMV-LS	Neural Machine Verbalization-LS
LSTM	Long short-term memory
LS	Link Specification
LSs	Link Specifications
IM	Intersection Matrix
DE-9IM	Dimensionally Extended 9-Intersection Model
SPARQL	SPARQL Protocol and RDF Query Language SPARQL
IoT	Internet of Things
URI	Uniform Resource Identifier

HTTP	Hypertext Transfer Protocol
IRI	Internationalized Resource Identifier
LSQ	Linked SPARQL Protocol and RDF Query Language SPARQL (SPARQL) Queries
LGD	LinkedGeoData
WWW	World Wide Web
W3C	World Wide Web Consortium
NLG	Natural Language Generation
XML	Extensible Markup Language
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
NMT	Neural Machine Translation
KGE	Knowledge Graph Embedding
NUTS	Nomenclature of Territorial Units for Statistics
CLC	Corine Land Cover
MBB	Minimum Bounding Box
MBBs	Minimum Bounding Boxes
WKT	Well-known Text Format for Geometries
EEA	European Environment Agency
GRU	Gated Recurrent Unit
CNN	Convolutional Neural Network
NMT	Neural Machine Translation
T5	Text-To-Text Transfer Transformer

Part I

Preliminaries

Introduction | 1

> *The key to growth is the introduction of higher dimensions of consciousness into our awareness.*
>
> — **Lao Tzu**
> (Chinese Philosopher)

The digital age has ushered in an era of unprecedented access to information. As of April 2023, 64.6% of the world's population had access to the World Wide Web (WWW)[1], which had grown to encompass at least 4.47 billion indexed pages[2]. This vast digital landscape, known as Web 2.0 or the Document Web, is interconnected through links, serving as a primary information source for billions of people. Search engines like *Google*, for example, process an average of 99 thousand search requests per second, translating to over 8.5 billion searches per day[3].

However, the Document Web has its limitations. The relationships within it are not typed, making it challenging to understand the significance of these connections. Moreover, traditional retrieval techniques often return entire web pages instead of specific, relevant information from across multiple pages. To address the drawbacks of the Document Web, Berners-Lee et al.[4] created the Semantic Web (SW) [BHL01], which is an extension of the Document Web and the existing WWW. The SW enables data to be presented in the form of a globally connected database that smart agents can access and process. Thus, the data must follow established formats and protocols and must be available and machine-understandable for agents (i.e., machines) to process them quickly.

The World Wide Web Consortium (W3C) provides the framework for SW standardization, including syntax, formal descriptions of terminology and concepts, and vocabularies. For example, the Resource Description Framework (RDF)[5] standard

[1] https://www.statista.com/statistics/617136/digital-population-worldwide/
[2] https://www.worldwidewebsize.com/ Accessed, 31 July, 2023
[3] https://www.oberlo.com/blog/google-search-statistics (Accessed, 31 July, 2023)
[4] https://en.wikipedia.org/wiki/Tim_Berners-Lee
[5] https://www.w3.org/RDF/

allows the representation of facts as triples (see an example about the former president of US *Barack Obama* represented as a set of triples in the Listing 1 in Terse RDF Triple Language (TTL) format[6]).

The Web of Linked Data, or Linked Data, is the manifestation of the SW and has grown from 12 KGs [Ngo+14a] to more than 10,000 KGs[7]. Recent technological advancements in hardware development and network infrastructure have facilitated the accumulation of vast amounts of data in a wide range of disciplines, such as life sciences, social networks, monitoring industrial plants [Los+11], monitoring open SPARQL endpoints [Sal+15], implementing the Internet of Things (IoT), and cloud computing [Meh+15]. As a result, 8.85 million pieces of information about individuals, locations, species, and artistic works are included in the KGs, such as *DBpedia*, the structural representation of *Wikipedia* pages. More than 1.3 billion facts in the Linked SPARQL Queries (LSQ) dataset [Sal+15] describe more than 250 million query events at the open SPARQL endpoints. There are more than 20 billion triples that describe millions of geographical things in LinkedGeoData (LGD) [ALH09]. In addition to the ever-increasing number of published KGs, we can also see how the size of an individual KG increases with each new edition. For example, *DBpedia* has grown from 103 million triples (DBpedia 2.0), representing 1.95 million entities, to 10.094 billion triples, representing more than 8.85 million entities in 2022[8].

However, the LOD cloud[9] is made entirely of open data that is freely reusable and distributed under an open license. LOD is a fraction of the Web of Linked Data. Data must adhere to a set of guidelines known as the "*Linked Data Principles*" in order to be published as Linked Data.

These principles are as follows:

1. First Principle. Use the Uniform Resource Identifier (URI)s to name things.

[6] https://www.w3.org/TeamSubmission/turtle/

[7] http://lodstats.aksw.org/

[8] https://DBpedia.org/sparql (Accessed, 10 March, 2022)

[9] https://lod-cloud.net/

```
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

  dbr:Barack_Obama dbo:birthDate "2013-05-14"^^xsd:date .
  dbr:Barack_Obama dbo:birthPlace dbr:Hawaii .
```

Listing 1: A simple set of triples in TTL format.

2. Second Principle. Use the Hypertext Transfer Protocol (HTTP) URIs to find those names.

3. Third Principle. Use standard formats, such as RDF and SPARQL, so that if someone searches for a URI, it provides useful information.

4. Fourth Principle. Include links to other URIs so that people can discover more things.

To realize the vision of the Web of Linked Data, resources within one KG must be linked to resources in another KG within LOD. Thus, in this work, we focus on the fourth principle. Given a source RDF KG G_s and a target RDF KG G_t, our objective is to generate new RDF statements that connect the resources of G_s with those of G_t. Establishing these links between KGs is crucial for producing Linked Data. Moreover, generating links between entities within KGs is essential to gather the data needed to build a comprehensive model for LOD. Linked KGs play a pivotal role in various data-driven applications, including information retrieval, NLP, recommendation systems, search engines, conversational agents, e-commerce platforms, and drug discovery. The importance of linking KGs in life sciences is illustrated by the question in example: *What are the side effects of drugs used to treat Tuberculosis?*

To obtain comprehensive results using the biomedical section of the Linked Data Question Answering benchmark [Ung+14], question answering systems must utilize links between pharmaceuticals specified in *Diseasome*[10] and drugs detailed in *Sider*[11]. The query in Listing 2 demonstrates that the Ontology Web Language (OWL) property `owl:sameAs` is employed to establish the link between pertinent resources in these KGs (associated with `?s2` and `?s3`).

For the effective creation of accurate links across KGs, there are two primary requirements, as outlined by *Nentwig et al.* [Nen+17] for a linking framework (i.e., LD tools). These requirements are the following:

[10] https://pubmed.ncbi.nlm.nih.gov/22891498/

[11] http://sideeffects.embl.de/

```
SELECT DISTINCT ?x WHERE {
disease:1154 diseasome:possibleDrug ?s2 .
?s2 rdf:type drugbank:drugs .
?s3 owl:sameAs ?s2 .
?s3 sider:sideEffect ?x .
}
```

Listing 2: SPARQL query to answer "*What are the side effects of drugs used for Tuberculosis?*".

- **Efficiency**. A linking tool should be fast and scalable, especially for large KGs with millions of resources. A simplistic, non-scalable approach evaluates all potential resource pairings (the Cartesian product), leading to quadratic complexity. Thus, a primary efficiency goal is to reduce search space to prevent unnecessary resource pairings. *Nentwig et al.* [Nen+17] have highlighted several frameworks that prioritize the efficiency of a linking framework.

- **Effectiveness**. A linking tool should produce high-quality mappings based on standard metrics such as accuracy, recall, and F-measure. The results should be precise, meaning that the relationships generated by a specific framework must be accurate. Additionally, to ensure completeness, a linking tool should generate as many accurate links as possible. Only genuine resource connections should be established. *Nentwig et al.* [Nen+17] have reviewed numerous tools that focus on the effectiveness of a linking framework.

Having introduced the two main requirements for efficient LD frameworks, we now turn our attention to the specific challenges associated with LD over Geospatial KGs. While there is comprehensive literature on the scalability of general LD, there are limited insights when it comes to the difficulties of LD over Geospatial KG. This work emphasizes the improvement of the scalability of LD over Geospatial KG, given the limited literature on Geospatial LD at scale. The rapid expansion of linked geospatial data requires scalable methods to identify links between geospatial resources. Past research indicates that only 7.1% of the relationships between resources are geographically related [Ngo13]. This can be attributed to the following: I) The large volume of geospatial resources on LOD requires efficient methods to compute the relationships between these resources. For example, LGD [ALH09] comprises more than 20 billion triples that describe millions of geographical entities. II) The need to compute specific relationships, such as distance and topological connections between geospatial resources, involves managing the vector representation of geospatial data.

Geospatial resources are now crucial for many real-time applications, including emergency response, location-based services, and real-time traffic management [ZL05]. Establishing links between real-time geospatial KGs is challenging, especially to support immediate decision-making. Thus, ensuring the efficiency and scalability of the LD process is increasingly complex. In our research, we develop innovative algorithms to improve the scalability of LD over Geospatial KGs. We examine how simplifying geospatial representations affects the quality of discovered relations and the efficiency of different LD approaches when using these simplified geometries. We also study the impact of parallel computing on the topological relations between geospatial resources, as defined by Dimensionally Extended 9-Intersection Model (DE-9IM). Additionally, we formalize the problem of topological relation discovery for geospatial resources, with a focus on content measures and R-Tree indexing. These algorithms are presented in Part II of the thesis, following the preliminary concepts covered in Part I. Building upon our novel algorithms and methods to address the scalability of LD over Geospatial KGs, we further extend Part II by introducing NELLIE.

NELLIE is a modular pipeline architecture designed to gather the data necessary to create a holistic model for LOD. It comprises a series of modules, each addressing a distinct data augmentation task. NELLIE starts by identifying relevant KGs for integration. It then manages the KG data integration at both ontology and instance levels. Subsequently, NELLIE fuses related instances/classes to form a unified KG and then embeds this KG. Our primary aim with this proposed architecture is to gather the essential data for creating a comprehensive model for LOD.

Having discussed the challenges and advancements in enhancing the scalability of LD, especially over Geospatial KGs, another equally pressing concern emerges: the intricacy of LSs and their accessibility to non-experts. A LS a specification with two components: *similarity measures* m for comparing properties and *operators* op for combining similarities. Frameworks such as SILK [IJB11] and LIMES [NA11] have provided robust mechanisms for determining link conditions, but their intricate specifications often remain elusive for non-experts. Although the literature is rich in discussions about the scalability of general LD, it has not delved into the challenge of making LD transparent and comprehensible. It remains largely unaddressed. This brings us to the forefront of "Explainable LD." In this pioneering endeavor, our aim is to demystify the world of complexes of LSs. Our innovative approach introduces a fresh perspective to LD, placing a premium on explainability. Using NLP techniques and integrating human-in-the-loop, we aspire to transform these complex LSs into intuitive natural language texts. In Part III of this thesis, we will dive into this novel approach, offering a new lens through which the LD community can view and understand LSs.

1.1 Motivation

The landscape of heterogeneous KGs adhering to linked data principles is steadily expanding, as we can see in Figure 1.1. These KGs find extensive applications in data-driven domains, including information retrieval, NLP, recommendation systems, search engines, conversational agents, e-commerce solutions, and drug discovery. However, only a fraction of these KGs are currently interconnected. According to recent statistics[12] from LOD[13], out of 1564 KGs with 395.121 billion triples, only 2.72 billion links (0.07%) exist between them. This underscores the significant challenge of discovering links among these KGs, a crucial step towards realizing the LOD vision[14]. The scarcity of links among instances in KGs is primarily due to the labor-intensive nature of manual link establishment, especially in large KGs such as *DBpedia*[15], LGD[16], Bio2RDF[17], KEGG [Kan+16], and *Wikidata*[18]. This challenge is further amplified by the continuous growth in the number and size of published KGs. For example, *DBpedia* has expanded from 103 million triples (DBpedia 2.0), representing 1.95 million entities, to 10.094 billion triples, representing more than 8.85 million entities in 2022[19]. As the number of independent data providers increases, the simultaneous publication of KGs with identical information is likely to become more prevalent. For example, DBLP has been published by several entities[20], leading to redundant content in the Data Web. Furthermore, different KGs often provide varying perspectives on the same data. For example, DrugBank[21] KG mainly describes drug interactions, pharmacology, chemical structures, targets, and metabolism, while Sider[22] KG focuses on drug side effects.

In the context of LD within Geospatial KGs, scalability presents a significant challenge. This is due to the inherent complexity of LD and the specific challenges associated with LD over Geospatial KGs. These challenges include: I) The sheer volume of geospatial resources in LOD, such as LGD [ALH09], which contains more than 20 billion triples. This vast quantity necessitates the development of scalable techniques to compute links between these resources. II) The computation of certain relationships, such as distance and topological connections among geospatial resources, requires dealing with vector representations of geospatial data. This adds another layer of complexity to the LD process in Geospatial KGs. As a result of such massive data expansion as well as multifaceted data publishing, there is a growing demand for LD approaches at scale. Many frameworks have been developed to address different challenges of data augmentation. Before combining KGs, such sys-

[12] Accessed 10.03.2022 https://lod-cloud.net/#about, retrieved using https://github.com/lod-cloud/lod-cloud-draw/blob/master/scripts/count-data.py

[13] https://lod-cloud.net/

[14] https://www.w3.org/DesignIssues/LinkedData.html

[15] https://wiki.DBpedia.org/

[16] http://.org/About

[17] https://download.bio2rdf.org/release/4

[18] https://www.wikidata.org/

[19] Accessed in 10.03.2022 fromhttps://DBpedia.org/sparql

[20] http://datahub.io/dataset/fu-berlin-dblp and http://dblp.rkbexplorer.com/

[21] https://go.drugbank.com/

[22] http://sideeffects.embl.de/

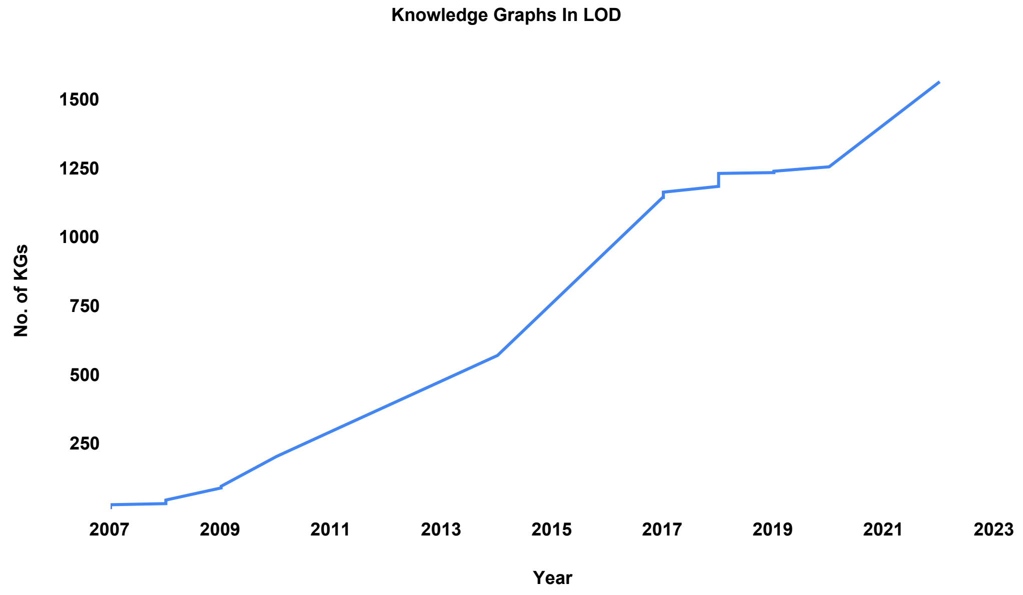

Fig. 1.1.: The landscape of heterogeneous KGs in LOD.

tems mainly identified semantically equivalent entities in different KGs or even the same KG, where they tried to achieve high efficiency and effectiveness in the linking process. For example, *LogMap* [JC11] and CODI [NMS10] use structural matching based on the ontology structure to discover links between ontologies. *Nentwig et al.* [Nen+17] list many data augmentation systems that have been developed in the last two decades. For example, LIMES [NA11; Ngo+21] and SILK [IJB11] apply matching strategies at the instance level to calculate property values. The authors of [Nen+17] address many challenges and aspects of the current LD frameworks. In a more recent survey [MT19], the authors have presented methods in the area of linked data integration, including LD and KG fusion. To fuse the data, the evaluation and fusion of the quality of the link data quality assessment and fusion SIEVE was proposed by [MMB12], which is integrated into the link data integration framework (LDIF) [Sch+11]. DEER [SNL15] is another data augmentation framework that can perform LD and fusion to produce enriched data.

In LD over Geospatial KGs, algorithms such as RADON [She+17], GIA.NT [Pap+21], and DORIC [Jin+21] have been developed. To address the challenges of efficiency, effectiveness, and explainability in LD, we present a set of novel approaches in this thesis that facilitate the following:

- Integration of enormous volumes of KGs within time or space restrictions: As the number and size of KGs continue to grow, integrating these large volumes of data becomes a significant challenge. This is particularly true when operating within constraints related to time or computational resources. Efficient integration techniques are required to handle these large-scale KGs.

- Techniques for connecting resources based on spatial links: While traditional LD methods focus on semantic and structural similarities, other types of link, such as spatial and phonetic links, have received less attention. Spatial links connect entities based on their geographical proximity or relatedness, which can be crucial in applications such as geospatial analysis, location-based services, and environmental studies.

- Explainable integration of KGs using natural language techniques: As linking KGs is based on complex LSs, there is a growing demand for explainability in KG integration. This involves not only generating accurate and reliable results, but also providing clear and understandable explanations of how these results are derived. Natural language techniques can play a crucial role in this regard. For example, rule-based natural language generation methods can transform complex data and relationships into human-readable texts. Sequence-to-sequence neural translation architectures can convert sequences of data into natural language descriptions. Large Language Model (LLM)s, such as the Text-To-Text Transfer Transformer (T5)[Raf+19] model, can generate comprehensive and coherent narratives from complex LSs used to link KGs. These techniques can significantly enhance the transparency, trustworthiness, and user-friendliness of KG integration.

- Proposing NELLIE, a pipeline architecture to build a chain of modules, in which each of our modules solves a challenge of data augmentation. NELLIE first addresses the problem of finding relevant KGs to integrate. Subsequently, NELLIE tackles the KG data integration task at both the ontology and instance levels. NELLIE then fuses the matched instances/classes to generate our fused KG. Finally, NELLIE performs the embedding of the KG of the resulting fused KG. Our end goal with this suggested architecture is to gather the data needed to build a holistic model for LOD linking, which, to our knowledge, does not exist. Our proposed architecture consists of three layers: the core layer, the application layer, and the publication layer. In this thesis, we pay more attention to the core layer, as it contains the main components and modules of our architecture. In particular, we address the following challenges in our paper: 1) KG matching (i.e., matching KGs based on their content); 2) KG linking, including ontology and instance matching; 3) KG fusion; and 4) KG embedding. Note that all these challenges are implemented in our core layer.

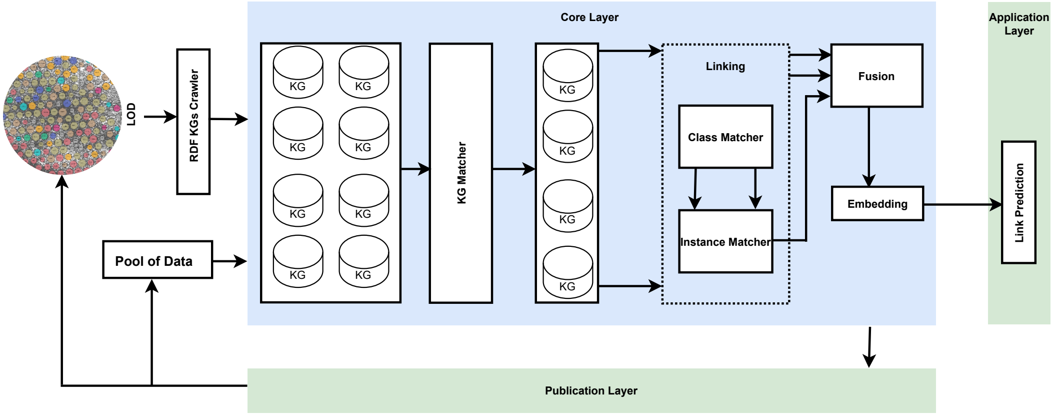

Fig. 1.2.: The modular pipeline architecture of NELLIE.

In the application layer, we address the link prediction challenge to evaluate the impact of our KG fusion on the link prediction task.

Figure 1.2 shows the architecture of NELLIE. In particular, we develop the key features of NELLIE as follows.

- Develop a modular pipeline architecture as a milestone for 24/7 linking that allows the collection of the data necessary to build a holistic model for LOD linking. Our modular architecture can be easily extended by adding more components and methods.
- Propose a two-phase linking strategy, starting with ontology matching and then instance matching.
- Carry out KG fusion to study the impact of KG fusion on link prediction and KG completion.

1.2 Research Gaps and Contributions

In this thesis, we tackle the main research gaps and challenges related to efficiency, effectiveness, and explainability in LD. Part I sets the stage by covering the essential preliminaries of the thesis. Building on this foundation, Part II addresses the research gaps denoted as Research Gap 1 and Research Gap 2. Subsequently, Part III delves into the challenges associated with Research Gap 3. We outline these research gaps as follows:

1.2.1 Research Gap 1: Geospatial Link Discovery over Knowledge Graphs at Scale

Research Gap 1 focuses on addressing the scalability of LD over geospatial KGs. This is an area that has been overlooked in the literature. In the context of Research Gap 1, our research is concentrated on the development of novel algorithms that meet the primary requirements of LD framework, such as scalability and effectiveness, with a special emphasis on scalability. Below are the solutions made to address Research Gap 1.

Research Question ($RQ_{1.1}$)

How does the simplification of geometries impact the performance, scalability, and accuracy of topological relations in geospatial LD approaches, and what is the associated computational cost of the simplification process?

To answer this research question, we investigate and formalize the challenges of LD for geospatial resources and the line simplification problem. We explore the impact of simplifying geospatial representations on the quality of discovered relations and the efficiency of various LD approaches when working with these simplified geometries. Our findings, detailed in Chapter 4, reveal an average loss of 15% in the F-measure of the original data but a significant speedup of up to $67\times$ when applied to the simplified data.

Research Question ($RQ_{1.2}$)

Does the parallel computation of the Intersection Matrix (IM) speed up the geospatial LD approach, and to what extent does the number of processors significantly influence IM computation?

We introduce RADON2 to tackle the challenge and address the research questions at hand ($RQ_{1.2}$). The approach taken by RADON2 involves calculating the IM for each geometry resource pair. These calculations are then stored, enabling efficient verification of topological relationships without the need to recalculate the IM each time such a relationship is queried. Our study explores the impact of parallel computing on these topological relationships between geospatial resources, as described by DE-9IM. Furthermore, we assess how the number of processors affects the computation time for IM. This topic is covered extensively in Chapter 5.

Research Question ($RQ_{1.3}$)

Does using content measures for the discovery of topological relationships in geospatial KGs result in efficiency gains in terms of reduced runtime, and what trade-off in accuracy, as measured by F-measure, occurs when employing these content measures? Moreover, how does the balance between accuracy and efficiency with simplified versions of the original polygons compare to using Minimum Bounding Box (MBB)s? Additionally, does the use of R-tree indexing impact runtime when applying content-based measures, and how does parallelization enhance performance, especially with large KGs such as Corine Land Cover (CLC)?

To answer these research questions, we introduce COBALT. COBALT combines content measures with R-tree indexing for the discovery of topological relationships in geospatial KGs. COBALT starts with a formalization of the problem of topological relation discovery for geospatial resources, emphasizing content measures and R-Tree indexing. COBALT then explores the efficiency and accuracy implications of using content-based measures for topological relation discovery. We also assess the computational costs of DE-9IM and the effects of parallel computing on topological relations and content-based measures. These findings are elaborated on in Chapter 6.

1.2.2 Research Gap 2: The Absence of Holistic Models for Linked Open Data

The proliferation of heterogeneous KGs adhering to linked data principles is evident. These KGs have become indispensable in data-driven applications, spanning from information retrieval, NLP, and recommendation systems, to search engines, conversational bots, e-commerce solutions, and drug discovery. Despite their ubiquity, there exists a void: the absence of holistic models for LOD linking. This contrasts with specific human languages, such as English, which benefit from LLMs, such as Bidirectional Encoder Representations from Transformers (BERT) [Dev+18].

Research Question ($RQ_{2.1}$)

To what extent is the construction of a modular pipeline architecture feasible as a milestone for 24/7 linking to link Knowledge Graph (KG)s needed for a holistic LOD model? Additionally, how significant is the impact of using the two-phase linking technique on LD in terms of enhancing efficiency and effectiveness?

To bridge this gap, we introduce NELLIE, a modular pipeline architecture crafted to sequentially address the challenges associated with enhancing data volume. The NELLIE pipeline begins by pinpointing relevant KGs for linkage. Then it delves into the intricate process of KG data integration, encompassing both ontology and instance levels. Once identified, the matching instances and classes are seamlessly fused by NELLIE, culminating in a unified KG. The final step involves NELLIE embedding this fused KG. NELLIE is a milestone for 24/7 linking to gather the data required to develop a holistic LOD model. Our modular architecture is readily extensible by adding more components and operations. We also propose a two-phase linking technique that begins with ontology matching and then moves on to instance matching. Furthermore, we provide a KG fusion method to investigate the influence of KG fusion on link prediction and KG completeness. These findings are elaborated on in Chapter 6.

1.2.3 Research Gap 3: Explainable Link Discovery

In the field of LD, declarative frameworks such as SILK [IJB11] and LIMES [NA11] employ complex LSs to define the conditions for linking two resources. For non-expert users, these complex LSs can be challenging to decipher, especially when using frameworks like LIMES. As we move towards explainable LD, the need for transparency and interpretability in LD becomes paramount. Our approach aims to verbalize these complex LSs, making them more accessible and understandable through NLP techniques.

Research Question ($RQ_{3.1}$)

Does the process of verbalizing LSs enhance user comprehension compared to the original LSs, and if so, how fluent and readable is the generated LSs verbalization? Furthermore, how accurately does the verbalization capture the essence of the underlying LSs, and what degree of information is potentially lost when applying our summarizing approach?

To solve this problem, we introduce a *template-based* approach to transform LSs into natural language, enhancing the explainability of the LD process. Inspired by the pipeline architecture of Natural Language Generation (NLG) systems, such as those presented by Reiter & Dale [RD00], our method aims to bridge the comprehension gap for users. Our evaluations suggest that our approach can generate comprehensive and user-friendly natural language descriptions. In addition, we also propose Multi-LSVS, a generic multilingual *template-based* approach that allows the verbalization of LSs in many languages, i.e., converts LSs into understandable natural language text. For example, our multilingual LSs verbalization framework verbalizes German and Spanish in addition to English. Our approach can generate complete and easily understandable natural language descriptions even by lay users. Furthermore, we introduce a neural-based LSs verbalization approach, trained using verbalization from our template-based method. We also highlight challenges that require further exploration and propose a multilingual method for concise LSs verbalization. Details are elaborated on in Chapter 8.

Research Question ($RQ_{3.2}$)

How does the complexity of LSs influence the performance of our Neural Machine Verbalization-LS (NMV-LS) when training standard encoder-decoder architectures? And does fine-tuning an LLM enhance the verbalization capabilities of our NMV-LS system, potentially helping it to generalize across different LSs for verbalization? Additionally, how does the use of human-annotated data compare to silver data in terms of the quality of the verbalization produced?

In our pursuit of explainable LD, we introduce NMV-LS, a language model-based approach for LSs verbalization. This method leverages a few-shot learning strategy by fine-tuning an LLM, such as T5. It is designed to verbalize a variety of LSs, reducing the need for crafting specific linguistic rules for each system. By generating examples in LIMES and fine-tuning a language model, our approach can verbalize LSs from other tools, like SILK, with minimal examples. This method is also adaptable to other languages. More insights can be found in Chapter 9.

1.3 Thesis Outline

This thesis is structured into three distinct parts, collectively comprising ten chapters. Part I includes three foundational chapters: Chapter 1 sets the stage by elucidating the motivation, research topics, hypotheses, and overarching contributions of the thesis. The subsequent Chapter 2 delineates the basic notation and formalization that underpin the thesis. Chapter 3 offers a comprehensive review of the state-of-the-art, contextualizing our proposed approaches within the broader academic landscape.

Part II spans four chapters, each systematically segmented into Motivation, Approach, and Evaluation sections. This segment of the thesis underscores our efforts to improve the effectiveness and efficiency of LD. Specifically, Chapter 4 explores LD over geospatial KGs using line simplification algorithms. Chapter 5 discusses LD over geospatial KG employing buffered IM. Chapter 6 introduces the COBALT approach, leveraging content measure similarity for efficiency. Lastly, Chapter 7 chronicles the development of NELLIE, a system designed to build a holistic model for LOD linking, with its trilayered structure depicted in Figure 1.2. In Figure 1.3, we illustrate the structure of the thesis. Part III is dedicated to the realm of explainable LD.

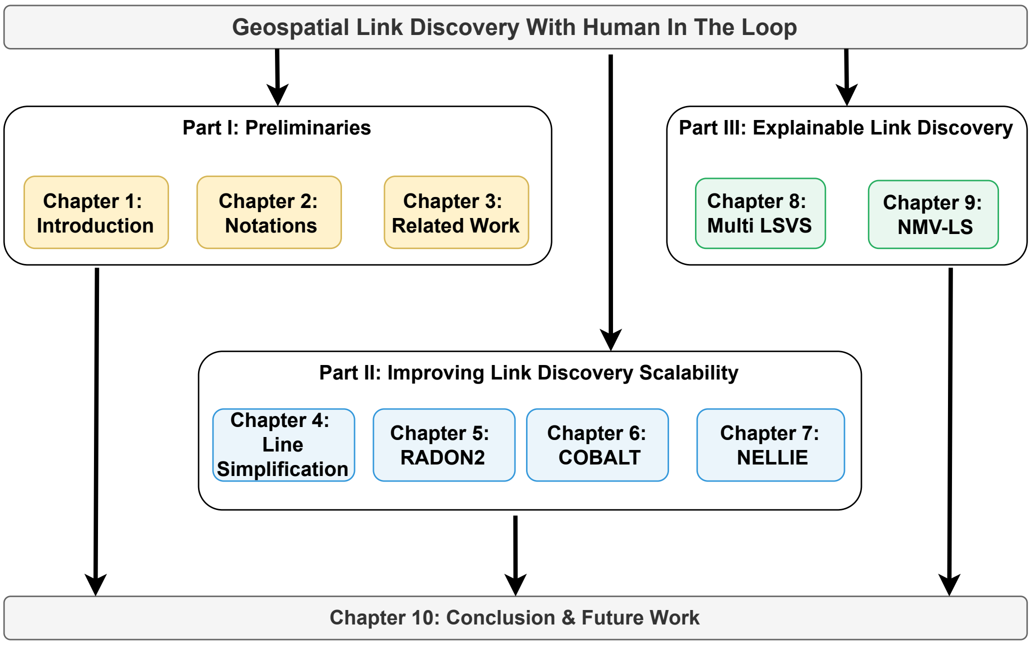
Geospatial Link Discovery With Human In The Loop
Part I: Preliminaries
Chapter 1: Introduction
Chapter 2: Notations
Chapter 3: Related Work
Part III: Explainable Link Discovery
Chapter 8: Multi LSVS
Chapter 9: NMV-LS
Part II: Improving Link Discovery Scalability
Chapter 4: Line Simplification
Chapter 5: RADON2
Chapter 6: COBALT
Chapter 7: NELLIE
Chapter 10: Conclusion & Future Work

Fig. 1.3.: Structure of the thesis.

Part III highlights the complexities inherent in LSs, which often necessitate expert interpretation. Chapter 8 pioneers a multilingual verbalization and summarizing approach for explainable LD, incorporating languages such as English, German, and Spanish. This chapter integrates a rule-based NLG architecture and presents an foray into a neural architecture: a bidirectional Recurrent neural network (RNN)-Long short-term memory (LSTM) 2-layer encoder-decoder model with an attention mechanism. Chapter 9 addresses the vocabulary challenges identified earlier by introducing the NMV-LS approach. This two-stage strategy combines rule-based verbalization with the capabilities of a few-shot learning technique, specifically fine-tuning the T5 language model. In Chapter 10, we conclude our work in this thesis and outline future work.

Notation 2

2.1 Linked Data

The Web of Linked Data serves as a prominent exemplar within the SW, illustrating the guiding principles of a Web of Data. These "best practices", often termed the Linked Data principles, were formulated by Berners-Lee et al.[1] SW [BHL01].

Adherence to these guidelines is essential for data publication. The principles are as follows:

- First Principle. Assign URIs to identify entities.
- Second Principle. Utilize HTTP URIs for resource retrieval.
- Third Principle. Adopt standard formats like RDF and SPARQL to provide meaningful information when a URI is accessed.
- Fourth Principle. Embed links to other URIs to facilitate further discovery.

The first three Linked Data principles emphasize the need to uniquely identify data "resources" on the Web using an HTTP URI [HB11]. In the context of Web architecture, "resource" denotes items of interest within a specific domain, encompassing both tangible entities and abstract concepts. A typical HTTP URI is structured as follows:

$$[scheme:][//authority][path][?query][\#fragment]. \tag{2.1}$$

The *authority* component is further broken down into:

$$authority = [userinfo@]host[:port]. \tag{2.2}$$

For instance, the Wikipedia article segment on "Fire and Blood" from "A Song of Ice and Fire" has the following URI:

[1] https://en.wikipedia.org/wiki/Tim_Berners-Lee

Example 2.1 *https://en.wikipedia.org/wiki/A_Song_of_Ice_and_Fire#Fire_&_Blood*

HTTP URIs are favored for two main reasons: (1) their simplicity ensures global uniqueness for any domain owner, and (2) they facilitate linking to descriptive information about the identified entity [HB11]. For a client to retrieve the URI using the HTTP protocol and obtain a resource description, the HTTP URI must be dereferenceable [Ngo+14a]. It is crucial to recognize that distinct URIs are employed to differentiate between an actual or abstract entity and its descriptive document. Depending on the end user, be it human or machine, the description of a resource can be rendered in either HyperText Markup Language (HTML) or RDF.

The W3C has defined the RDF [Kly04] data model to represent knowledge on the Web in a standardized way. In RDF, information is depicted as statements or RDF triples, each comprising a subject, a predicate, and an object, analogous to the components of a basic sentence.

2.2 Knowledge Graphs

An RDF KG G is a set of triples $(s, p, o) \in (\mathcal{R} \cup \mathcal{B}) \times \mathcal{P} \times (\mathcal{R} \cup \mathcal{L} \cup \mathcal{B})$, where $\mathcal{R}$ is the set of all resources, $\mathcal{B}$ is the set of all blank nodes, $\mathcal{P}$ is the set of all predicates and $\mathcal{L}$ the set of all literals. A resource is designated by the subject or the object of a triple, which may be either an Internationalized Resource Identifier (IRI) or an unidentified object (blank node). A UNICODE-compatible generalization of URIs is an IRI. An IRI may be used to identify the predicate of an RDF triple, which indicates the nature of the relationship between the subject and the object. The object of the triple can also be a blank node, an IRI literal, or an RDF represented by a UNICODE string.

Some languages are developed on top of RDF and enhance it with more expressive semantics to represent more complicated information using RDF triples. The Resource Description Framework Schema (RDFS), one of these languages, is a collection of classes that make use of the RDF data model and have several specific attributes. For the purpose of structuring RDF resources, it offers RDF vocabularies as a common language. Web Ontology Language is a different expressive representation language founded on formal logic OWL. OWL makes it possible to apply logic to the information and provides access to knowledge that is only implicitly modeled [HKR09]. An example of an OWL term is `http://www.w3.org/2002/07/owl#sameAs`. For simplicity reasons, it can be rewritten as `owl:sameAs` , where the namespace `http://www.w3.org/2002/07/owl#` is substituted with the prefix `owl`. Both RDFS and OWL follow W3C standardization. However, we require methods for serialization to publish an RDF KG on the Web. Since the RDF model is a data model and not a data format, several serializations that adhere to W3C standards have been developed, including RDF/Extensible Markup Language (XML) [BM04], Turtle [Bec+14], N-Triples, and RDF/JavaScript Object Notation (JSON),(See Figure 2.1).

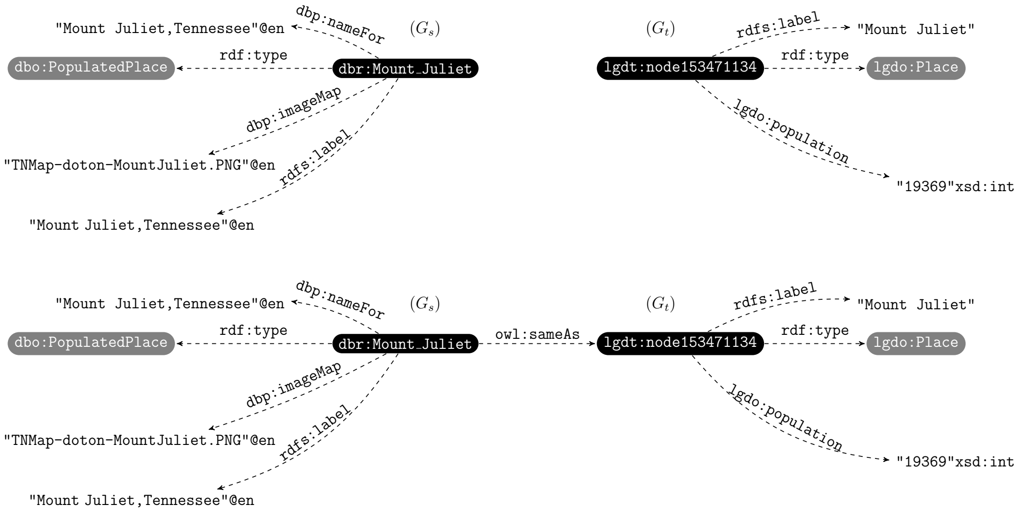

Fig. 2.1.: Example of RDF triples as KGs that describe the DBpedia resource `dbr:Mount_Juliet` with the LinkedGeoData resource `lgdt:node153471134`. We use `owl:sameAs` to link the DBpedia resource `dbr:Mount_Juliet` with the LinkedGeoData resource `lgdt:node153471134`.

2.3 Ontology Matching

Let O_s and O_t denote the source and target ontologies, represented by the sets $\{c_1^s, c_2^s, \ldots, c_M^s\}$ for O_s and $\{c_1^t, c_2^t, \ldots, c_N^t\}$ for O_t. Ontology alignment is a multifaceted task that involves identifying a variety of relationships between pairs of concepts from these sets. Such relationships can range from straightforward to complex, encompassing transformations, as discussed by Thiéblin et al. [Thi+20], to inference mechanisms outlined by Zhou [Zho+18]. Nevertheless, the focal point of this research is to isolate and identify specific pairs (c_i, c_j) within $O_s \times O_t$ for which a predefined relationship $r(c_i, c_j)$ is valid [JC11]. An example of such a relationship is the equivalence between the classes 'City' and 'Town', formally represented as `owl:equivalentClass(City, Town)`.

2.4 Link Discovery

Over time, numerous strategies have emerged to address the problem of identifying connections between resources. Initially, the common approach was to manually identify pairs (s, t), where $s \in G_s, t \in G_t$, that satisfy a certain relation R. However, the scale of LOD has expanded dramatically, from an initial size of 12 KGs [Ngo+14c] to now exceeding 1564 KGs[2]. As such, manually identifying these links has become an impractical and time-intensive endeavor. Given this context, LD frameworks have gained importance. These frameworks aim to identify a set of links, denoted $M^* \subseteq G_s \times G_t$, where the measure of similarity $m(s, t)$ is at least as great as a predefined threshold θ. Formally, the objective is to calculate the mapping M^* so that $\{(s, t) \in G_s \times G_t : m(s, t) \geq \theta\ \}$. The concept also extends to distance metrics. Using a distance function σ, $\mathcal{M}^*$ can be defined as $(s, t) \in G_s \times G_t : \sigma(s, t) \leq \varpi$, where $\varpi \in [0, \infty)$ serves as a distance threshold. Both the distance and similarity functions can be used interchangeably by converting one into the other, using the relationships $\sigma(s, t) = \frac{1}{1+m(s,t)}$ and $\theta = \frac{1}{1+\varpi}$.

A *similarity function* m is termed *atomic* if it is based on a single similarity measure applied to a specific set of properties, denoted $p_s, p_t \in P$. For simplicity, this relationship is written as $m(s, t, p_s, p_t)$ or simply as $m(p_s, p_t)$. It is termed *complex* if it combines multiple similarity measures through a metric operator such as $\max$ or $\min$.

2.4.1 Link Specifications

Declarative LD frameworks define the conditions necessary to generate such links using LS. Several grammars have been used to describe LS in previous work [IJB11; SNL17; NL12]. In general, these grammars assume that an LS consists of two types of atomic components: *similarity measures* m, which allow comparison of property values of input resources; and *operators* op that can be used to combine these similarities with more complex specifications. Without loss of generality, we define a similarity measure m as a function $m : G_s \times G_t \rightarrow [0, 1]$. We use *mappings* $M \subseteq G_s \times G_t$ to store the results of the application of a similarity function to $G_s \times G_t$ or subsets thereof. We define a *filter* as a function $f(m, \theta)$. We call a specification (*atomic* LS) when it consists of exactly one filtering function. A complex specification (*complex* LS) can be obtained by combining two specifications, L_1 and L_2, through an *operator* op that allows the results of L_1 and L_2 to be merged. Here, we use the

[2] `https://lod-cloud.net/`

```
OR(jaccard(x.name,y.name)|0.42,trigrams(x.name,y.description)|0.61)
```

Listing 1: Running example.

operators $\sqcap$, $\sqcup$, and $\backslash$ as they are complete and frequently used to define LS [SNL17]. A graphical representation of the complex LS of our example from Listing 1 is given in Figure 2.2. We define the semantics $[[L]]_M$ of an LS L with respect to a mapping M as given in Table 2.1. The semantics are similar to those used in languages like SPARQL, i.e., they are defined extensionally through the mappings they generate. The mapping $[[L]]$ of an LS L with respect to $G_s \times G_t$ contains the links that will be generated by L.

Tab. 2.1.: LS syntax and semantics.

LS	$[[LS]]_M$
$f(m,\theta)$	$\{(s,t)\|(s,t) \in M \wedge m(s,t) \geq \theta\}$
$L_1 \sqcap L_2$	$\{(s,t)\|(s,t) \in [[L_1]]_M \wedge (s,t) \in [[L_2]]_M\}$
$L_1 \sqcup L_2$	$\{(s,t)\|(s,t) \in [[L_1]]_M \vee (s,t) \in [[L_2]]_M\}$
$L_1 \backslash L_2$	$\{(s,t)\|(s,t) \in [[L_1]]_M \wedge (s,t) \notin [[L_2]]_M\}$

An LS L is *subsumed* by L', denoted by $L \sqsubseteq L'$, if for all mappings M, we have $[[L]]_M \subseteq [[L']]_M$. Two LS are *equivalent*, denoted by $L \equiv L'$ iff $L \sqsubseteq L'$ and $L' \sqsubseteq L$. Subsumption ($\sqsubseteq$) is a partial order over $\mathcal{L}$.

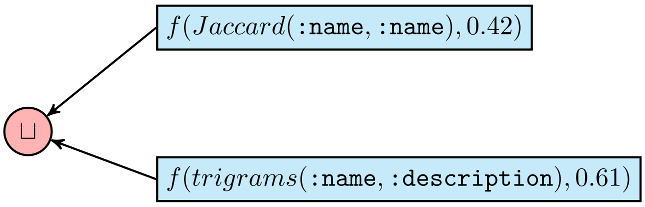
$f(Jaccard(\texttt{:name}, \texttt{:name}), 0.42)$
$\sqcup$
$f(trigrams(\texttt{:name}, \texttt{:description}), 0.61)$

Fig. 2.2.: An example of complex LSs. The filter nodes are rectangles, while the operator node is a circle.

2.4.2 Link Specifications Verbalization

Our definition of the realization function ζ relies on the formalization of LS declared in the previous section. Let $\mathcal{A}$ be the set of all (*atomic* LS) that can be combined in a (*complex* LS) L. Let C^S and C^T be two sets of constraints that specify the sets S and T, respectively. Let $\mathcal{M}$ be a set of similarity measures and $\mathcal{T}$ a set of thresholds. In general, a constraint C is a logical predicate. The restrictions in LS could state, for example, the `rdf:type` of the elements of the set they describe, that is, $C(x) \leftrightarrow x$ `rdf:type someClass`, or the characteristics that each element of the set must have, for example, $C(x) \leftrightarrow (\exists y : x \texttt{ someProperty } y)$. Each $s \in S$ must comply with each of the constraints $C_1^S \ldots C_m^S$, while each $t \in T$ must comply with each of the constraints $C_1^T \ldots C_k^T$. We call $z \in \mathcal{A} \cup C^S \cup C^T \cup \mathcal{M} \cup \mathcal{T}$ an *atom*. We define the realization function $\zeta : \mathcal{A} \cup C^S \cup C^T \cup \mathcal{M} \cup \mathcal{T} \rightarrow Language$, where $Language$ is our target language, which can be English, German, or Spanish. In turn, this realization function ζ maps each atom to a word or a sequence of words in our target language.

Tab. 2.2.: Dependencies used by LS verbalization.

Dependency	Explanation
`amod`	Represents the *adjectival modifier* dependency. For example, `amod(ROSE,WHITE)` stands for `white` rose.
`dobj`	Dependency between a verb and its *direct object*. For example, `dobj(EAT,APPLE)` expresses "to eat an/the apple".
`nn`	The *noun compound modifier* is used to modify a head noun by the means of another noun. For instance, `nn(FARMER,JOHN)` stands for `farmer` John.
`poss`	Expresses a possessive dependency between two lexical items. For example, `poss(JOHN,DOG)` expresses `John`'s dog.
`prep_X`	Stands for the preposition `X`, where `X` can be any preposition, such as, `via`, `of`, `in`, and `between`.
`subj`	Relation between *subject* and verb. For example, `subj(PLAY,JOHN)` expresses `John` plays.
`coord`	Stands for the relation between a conjunct (many conjuncts) and a given conjunction (in most cases **and** or **or**). For example, in the sentence **an** apple and a pear, `coord(PEAR, APPLE)` holds.

Formally, the goals of this chapter are twofold: First, construct the extension of ζ to the entire LS so that all *atoms* z can be mapped to their realization $\zeta(x)$. Second, manage how these atomic realizations can be combined. For the sake of simplicity, we denote the extension of ζ by the same label ζ. We adopt a rule-based approach to achieve this goal, where the rule extending ζ to the entire LS is expressed in a conjunctive manner. This means that for premises $P_1, \ldots, P_n$ and consequences $K_1, \ldots, K_m$ we write $P_1 \wedge \ldots \wedge P_n \Rightarrow K_1 \wedge \ldots \wedge K_m$. The premises and consequences are clarified by using an extension of the Stanford dependencies[3]. In particular,

[3] For a complete description of the vocabulary, see `http://nlp.stanford.edu/software/dependencies_manual.pdf`.

we build on the constructs explained in Table 2.2. For example, the dependency between a $verb$ and its $object$ is represented as $\texttt{dobj}(verb, object)$.

Neural Machine Verbalization

Given a source sentence $\mathbf{x}$ and a target sentence $\mathbf{y}$, verbalization is tasked with finding $\mathbf{y}$ that maximizes the conditional probability of $\mathbf{y}$ (i.e., $\arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$). In neural machine verbalization, an encoder-decoder model with a set of parameters is trained to maximize the conditional probability of sentence pairs using a parallel training dataset. Accordingly, a verbalization model that learned the conditional distribution can generate a corresponding verbalization of a given sentence by searching for the sentence that maximizes the conditional probability. This definition is inspired by the definition of Neural Machine Translation (NMT) [BCB15].

2.5 Knowledge Graph Fusion

Let G_s be a finite source KG and G_t be a finite target KG. The aim of KG fusion is to find a consolidated KG $G_{s \oplus t}$ that contains a fused version of the related entities from G_s and G_t. We assume that we have a mapping M_{merge} that contains a set of pairs of similar entities among G_s and G_t. A KG fusion approach fuses each pair of similar entities into a single entity by applying a fusion strategy operator $\oplus$. In this thesis, we apply an *additive fusion operator*.

2.6 Link Prediction

Given a subset of all true triples, the goal is to learn a *scoring function* ϕ that assigns a score $s = \phi(e_s, r, e_o) \in \mathbb{R}$, which shows if a triple is true. (e_s, r, e_o) with e_s is the subject entity, e_o is the object entity, and r is a relation. The ultimate goal of link prediction is to learn a *scoring function* ϕ to correctly score all missing triples. In the case of linear models such as TuckER, ComplEx, and DistMult , the scoring function is a specific form of tensor factorization; in non-linear models, the scoring function is a more complex (deep) neural network architecture. For a particular triple, a score is either positive if the model predicts a true fact or negative for a false one. Furthermore, the logistic sigmoid function is typically applied to the score to return a corresponding probability prediction $p = \sigma(s) \in [0, 1]$ to determine if a certain fact is true. Table 2.3 lists the scoring functions of three state-of-the-art link prediction models that we selected for our experiments. All three models are linear.

Tab. 2.3.: Scoring functions for three state-of-the-art link prediction models, together with the dimensionality of their relation parameters and major terms of their space complexity. $\overline{\mathbf{e}}_o \in \mathbb{C}^{d_e}$ is the complex conjugate of $\mathbf{e}_o$, $\underline{\mathbf{e}_s}, \underline{\mathbf{w}_r} \in \mathbb{R}^{d_w \times d_h}$ denotes a 2D reshaping of $\mathbf{e}_s$ and $\mathbf{w}_r$ respectively, $\mathbf{h}_{e_s}, \mathbf{t}_{e_s} \in \mathbb{R}^{d_e}$ are the head and tail entity embedding of entity e_s, and $\mathbf{w}_{r^{-1}} \in \mathbb{R}^{d_r}$ is the embedding of relation r^{-1} (which is the inverse of relation r). $\langle\cdot\rangle$ denotes the dot product and $\times_n$ denotes the tensor product along the n-th mode, and $\mathcal{W} \in \mathbb{R}^{d_e \times d_e \times d_r}$ is the core tensor of a Tucker decomposition.

Model	Scoring Function	Relation Parameters	Space Complexity
TuckER [BAH19]	$\mathcal{W} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_r \times_3 \mathbf{e}_o$	$\mathbf{w}_r \in \mathbb{R}^{d_r}$	$\mathcal{O}(n_e d_e + n_r d_r)$
DistMult [Yan+15]	$\langle \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \rangle$	$\mathbf{w}_r \in \mathbb{R}^{d_e}$	$\mathcal{O}(n_e d_e + n_r d_e)$
ComplEx [Tro+16]	$\mathrm{Re}(\langle \mathbf{e}_s, \mathbf{w}_r, \overline{\mathbf{e}}_o \rangle)$	$\mathbf{w}_r \in \mathbb{C}^{d_e}$	$\mathcal{O}(n_e d_e + n_r d_e)$

2.7 Well-Known Text Format for Geometries

Storing spatial objects in a standardized and readable format is a complex undertaking. A commonly used standard to represent spatial objects is Well-known Text Format for Geometries (WKT) [Inc11]. In the context of this thesis, the focus will be mainly on features of WKT related to polygons.

WKT offers robust solutions to the complex challenge of representing national territories by utilizing an array of data types, including point, linestring, polygon, and multipolygon. For instance, a country's territory may consist of multiple disconnected parts, such as Spain, which includes the mainland and several islands. Another challenge arises when a country's territory completely encircles portions of another country's territory. An example is the German municipality "Büsingen am Hochrhein", which is encircled by Switzerland. It is essential to have a format capable of storing such complex spatial data, including territories with "holes."

A point simply consists of an x-coordinate and a y-coordinate, represented as *point (1, 2)* for a point at coordinates (1, 2). A Linestring is composed of multiple points to represent a linear shape, such as *linestring (1 2, 3 4)* to describe a line from point (1, 2) to point (3, 4).

Polygons in WKT are defined using an exterior ring and one or more interior rings. The exterior ring, essentially a closed linestring, delineates the polygon's outer boundary. The interior rings specify areas within the exterior ring that do not belong to the polygon. For example, a square with vertices at points (0, 0) and (3, 3) would be represented as *polygon((0 0, 0 3, 3 3, 3 0, 0 0))*. If we want to exclude a smaller square with vertices at (1, 1) and (2, 2) from this polygon, the representation would be *polygon((0 0, 0 3, 3 3, 3 0, 0 0), (1 1, 1 2, 2 2, 2 1, 1 1))*.

In theory, it is possible to represent a territory of multiple disconnected islands within a single polygon by defining the exterior ring to encompass the entire world and excluding all areas outside the islands. However, to simplify such cases, WKT introduces the multipolygon data type. A multipolygon consists of multiple individual polygons, each with its own interior and exterior rings.

2.8 Topological Relations

In the field of spatial analysis, the topological relations between geometric entities play a critical role. Whether it is in Geographic Information Systems (GIS), urban planning, or environmental studies, a nuanced understanding of how spatial elements relate to each other is crucial. Two methods stand out for their effectiveness and widespread use in calculating these relations: DE-9IM and content measures. These computational models serve as foundational tools to assess and describe spatial relationships, each with its own unique set of benefits, limitations, and mathematical underpinnings. As we delve into the intricacies of spatial relations, these two approaches will serve as our guiding frameworks. These topological relations reflect the semantics of the English language [CSE94; CDV93] including `equals`, `within`, `contains`, `disjoint`, `touches`, `meets`, `covers`, `coveredBy`, `intersects`, `crosses`, and `overlaps` as shown in Figure 2.3.

2.8.1 The DE-9IM

The Dimensionally Extended 9-Intersection Model (DE-9IM)[CSE94] is a topological model and a standard used to describe the spatial relations of two geometries in two-dimensional space. Since the spatial relations expressed by DE-9IM are topological, they are invariant to rotation, translation, and scaling transformations [EF91]. The DE-9IM model is based on a 3×3 IM of the form:

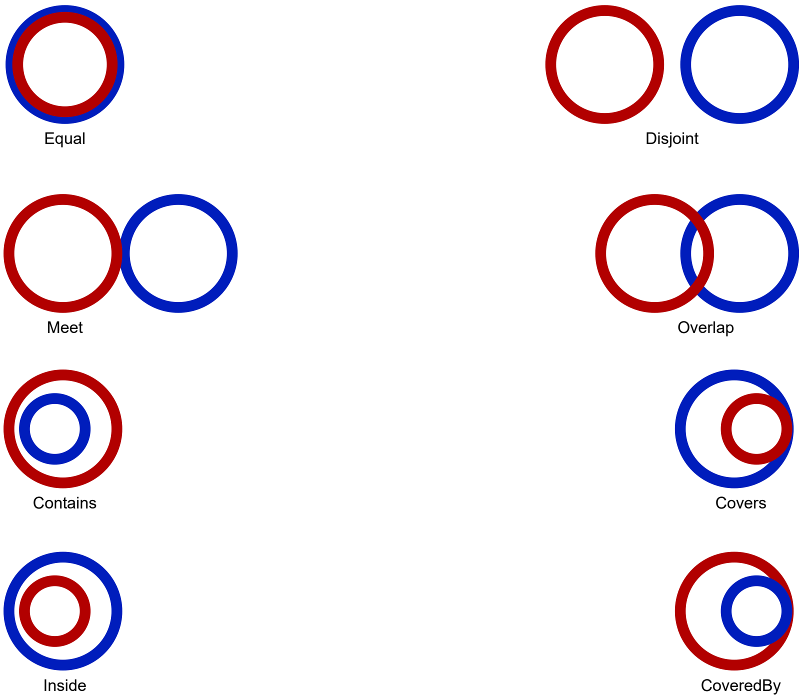

Fig. 2.3.: The content measure relations.

$$DE9IM(g_1, g_2) = \begin{bmatrix} dim(I(g_1) \cap I(g_2)) & dim(I(g_1) \cap B(g_2)) & dim(I(g_1) \cap E(g_2)) \\ dim(B(g_1) \cap I(g_2)) & dim(B(g_1) \cap B(g_2)) & dim(B(g_1) \cap E(g_2)) \\ dim(E(g_1) \cap I(g_2)) & dim(E(g_1) \cap B(g_2)) & dim(E(g_1) \cap E(g_2)) \end{bmatrix} \tag{2.3}$$

where dim is the maximum number of dimensions of the intersection $\cap$ of $interior(I)$, $boundary(B)$, or $exterior(E)$ of the two geometries g_1 and g_2. The domain of dim is $\{-1, 0, 1, 2\}$, where -1 indicates that there is no intersection, 0 represents an intersection that results in a set of one or more points, 1 indicates an intersection made up of lines, and 2 represents an intersection that results in an area. A simplified binary version of $dim(x)$ with the binary domain $\{true, false\}$ is obtained using the Boolean function $\beta(dim(I(g)) = false$ iff $dim(I(g)) = -1$ and $true$ otherwise. There is only a subset of topological relations available through DE-9IM that reflects the semantics of the English language [CSE94; CDV93] including `equals`, `within`, `contains`, `disjoint`, `touches`, `meets`, `covers`, `coveredBy`, `intersects`, `crosses`, and `overlaps`. The relations distinguished by the DE-9IM are shown in Figure 2.3.

2.8.2 Content Measures of Topological Relations

We use the content measures defined by Godoy et al. [GR04] to decide whether the relation $r(s,t)$ exists. Godoy et al. have implemented three content measures to determine whether a topological relation between two polygons exists by comparing the area, the diagonal or the area, and the diagonal of the polygon's minimum bounding boxes. The relations distinguished by the content measures are shown in Figure 2.3. We formally define the basic components necessary to compute content similarity.

Definition 1 *The* ***MBB*** *of a polygon P with $n \geq 3$ points $((x_1,y_1),\ldots,(x_n,y_n))$ is defined as the smallest rectangle that contains all the points of the polygon. Formally, MBB$(P) = \Big((X_{min},Y_{max}),(X_{max},Y_{min})\Big)$, where $X_{min} = \min(x_1,\ldots,x_n)$, $X_{max} = \max(x_1,\ldots,x_n)$, $Y_{min} = \min(y_1,\ldots,y_n)$ and $Y_{max} = \max(y_1,\ldots,y_n)$.*

Definition 2 *The* ***area*** *of an MBB M is defined as $area(M) = (X_{max} - X_{min}) \cdot (Y_{max} - Y_{min})$.*

Definition 3 *The* ***diagonal*** *of an MBB M is formally defined as $diagonal(M) = \sqrt{(X_{max} - X_{min})^2 + (Y_{max} - Y_{min})^2}$.*

Definition 4 *The MBB of the* ***union*** *of two MBBs A and B is defined as MBB$(A \cup B) = \Big((\min(X^A_{min},X^B_{min}),\max(Y^A_{min},Y^B_{min})),(\max(X^A_{max},X^B_{max}),\min(Y^A_{max},Y^B_{max}))\Big)$.*

Definition 5 *The* ***intersection*** *of two MBBs A and B is formally defined as MBB$(A \cap B) = \Big((\max(X^A_{min},X^B_{min}),\min(Y^A_{min},Y^B_{min})),(\min(X^A_{max},X^B_{max}),\max(Y^A_{max},Y^B_{max}))\Big)$.*

Definition 6 *To check if two MBBs A and B* ***overlap on their x-coordinates***

$$proj_{X(A,B)} = \begin{cases} \textit{false} & \textit{if } X^A_{max} < X^B_{min} \\ \textit{false} & \textit{if } X^A_{min} > X^B_{max} \\ \textit{true} & \textit{otherwise} \end{cases}$$

Definition 7 *To check if the two MBBs A and B* ***overlap on their y-coordinates***

$$proj_{Y(A,B)} = \begin{cases} \textit{false} & \textit{if } Y^A_{max} < Y^B_{min} \\ \textit{false} & \textit{if } Y^A_{min} > Y^B_{max} \\ \textit{true} & \textit{otherwise} \end{cases}$$

Definition 8 *To further analyze the relation of two MBBs A and B, we also need the* ***distance*** *between them.*

$$d(A,B) = \begin{cases} \Big(\min(|X^A_{min} - X^B_{max}|, |X^B_{min} - X^A_{max}|)\Big) \textit{ if } \neg \textit{proj}_X(A,B) \wedge \textit{proj}_Y(A,B) \\ \Big(\min(|Y^A_{min} - Y^B_{max}|, |Y^B_{min} - Y^A_{max}|)\Big) \textit{ if } \textit{proj}_X(A,B) \wedge \neg \textit{proj}_Y(A,B) \\ \sqrt{\begin{pmatrix} (\min(|X^A_{min} - X^B_{max}|, |X^B_{min} - X^A_{max}|))^2 \\ +(\min(|Y^A_{min} - Y^B_{max}|, |Y^B_{min} - Y^A_{max}|))^2 \end{pmatrix}} \textit{ if } \neg \textit{proj}_X(A,B) \wedge \neg \textit{proj}_Y(A,B) \\ \begin{pmatrix} -\min(|X^A_{min} - X^B_{min}|, |X^A_{max} - X^B_{max}|, \\ |Y^A_{min} - Y^B_{min}|, |Y^A_{max} - Y^B_{max}|) \end{pmatrix} \textit{ if } area(A \cup B) = \max(area(A), area(B)) \\ 0 \textit{ if } area(A \cup B) \neq max(area(A), area(B)) \wedge \textit{proj}_X(A,B) \wedge \textit{proj}_Y(A,B) \end{cases}$$

3 Related Work

> *A picture is worth a thousand words. An interface is worth a thousand pictures.*
>
> — **Ben Shneiderman**
> (Professor of Computer Science)

The Related Work chapter provides a comprehensive review of existing research that closely aligns with the key research gaps identified in this thesis concerning LD over KGs, with a particular focus on geospatial KGs. These research gaps include the challenges related to scalability, the need for more comprehensive models for LOD, and the increasing importance of explainability in LD. In the realm of scalability (**Research Gap 1**), we explore contributions from seminal works such as ORCHID, RADON, GIA.nt, Strabon, Geo-L, and MaskLink, each providing distinct perspectives and solutions to the scalability challenges inherent to geospatial LD. For **Research Gap 2**, which focuses on the absence of holistic models for LOD, covering works in data fusion, data integration and KG embedding. These works delve into issues such as data completeness and conciseness, as well as advanced techniques for link prediction within KGs. The chapter also covers works about the general LD landscape, including instance matching and ontology matching. Lastly, in addressing **Research Gap 3** concerning explainability in LD, the chapter reviews advancements in the field of Explainable Artificial Intelligence (XAI). It discusses how XAI has been applied to LD, focusing on various methods, such as template-based, NM-based, and LLM-based approaches.

By exploring these related works, the chapter aims to position the contributions of this thesis within the broader academic landscape. This allows us to highlight the thesis's unique focus on developing scalable, effective, and explainable methodologies specifically tailored for LD over geospatial KGs.

3.1 Link Discovery

LD in the realm of KGs and SW technologies has seen remarkable innovations. This process is crucial for interlinking resources between different RDF KGs, thereby facilitating data integration, query routing, and numerous other applications. The LD landscape can be broadly divided into two categories-instance matching and ontology matching- both of which utilize sophisticated frameworks and machine learning algorithms to address their respective challenges.

3.1.1 Instance Matching

Declarative LD frameworks build on complex LSs to declare the conditions necessary for linking resources between RDF KGs. For example, the SILK [IJB11] and LIMES [NA11] frameworks employ property-based methods for the computation of links between instances. LIMES aids its users to create LSs manually and execute them against source-target resources or by using various machine learning approaches such as WOMBAT [SNL17] and EAGLE [NL12]. LIMES [NA11] offers many different approximation techniques based on metric spaces to estimate the similarities between instances. SERIMI [Ara+11] is an automatic interlinking method that matches instances between a source KG and a target KG. SERIMI performs linking without prior knowledge of the data, domain, or schema of these knowledge bases. The authors of [Niu+12] introduce a semi-supervised learning algorithm to automatically discover data set–specific instance matching rules. SLINT [NIL12] uses an approach to schema-independent interlinking. In particular, SLINT starts by automatically selecting important RDF predicates using coverage and discriminability. It then uses weighted co-occurrence and adaptive filtering to carry out instance matching.

3.1.2 Ontology Matching

Knowledge integration is heavily dependent on ontology alignment, which has been the subject of extensive research in recent years. We list some of the state-of-the-art systems. LogMap [JC11] is a highly scalable ontology matching system that provides reasoning and diagnosis capabilities that are built into the framework. CODI [NMS10] is another ontology matching framework based on Markov logic. The system defines syntax and semantics and formalizes the ontology matching problem. The authors have introduced in their work [Che+21b] an ML extension that takes advantage of distant supervision and semantic embedding and may be applied to improve conventional ontology alignment systems. To put it simply, it first generates high-precision seed mappings using the original ontology alignment system and class disjointness constraints (as heuristic rules), and then uses these mappings to train a Siamese Neural Network (SiamNN) for predicting cross-ontology class mappings via semantic embeddings in OWL2Vec, an ontology-adapted language model [Che+21a].

In [IAK21], the authors have introduced VeeAlign, a deep learning-based model that employs a new dual attention technique to compute the contextualized representation of a notion, which is then used to find alignments. By doing this, this approach is not only able to take advantage of the syntactic and semantic data embedded in ontologies but is also, by design, versatile and scalable to different domains with little effort.

Recently, OntoConnect has been introduced in [Cha+21], which describes a domain-independent, non-human intervention ontology alignment method that uses graph embedding with negative sampling.

3.2 Geospatial Link Discovery over Knowledge Graphs

Geospatial LD over KGs is a specialized branch within the broader realm of LD, focusing on the discovery and understanding of topological relationships between geospatial entities. The area remains relatively underexplored, offering novel challenges and complexities due to the involvement of spatial data and relations [Ngo+14b]. Various methodologies and frameworks have emerged, aiming to optimize computations, reduce memory footprint, and enhance performance in geospatial LD tasks and in the field of spatial analysis, such as using line simplification techniques.

3.2.1 Orchid

The reduction-ratio-optimal approach Orchid [Ngo13] optimizes the computation of point sets based on the distance among geospatial entities. The main idea of Orchid is to apply space tiling for both source and target resources and to compare only resources within a given range. While the use of Orchid in [Ngo13] is based only on the *Hausdorff* metric, the work in [SN15] extends Orchid to other point-set distance metrics such as *mean* and *sum of minimums*.

3.2.2 Silk

Semros and Koubarakis proposed one of the earliest geospatial data link discovery approaches as an extension to Silk [SK16]. To avoid the high computational cost of computing the DE-9IM intersection matrix, they introduced a blocking technique. This method divides the Earth into multiple equally sized blocks and computes the MBB for each geometry. Each block is assigned a set containing the geometries that intersect the block's MBB. Geometries that do not share a block are considered disjoint, eliminating the need to compute the intersection matrix. By altering the size of the blocks, this approach offers a trade-off between avoiding computations and optimizing indexing time.

3.2.3 RADON

Radon [She+17], developed by Sherif et al., further optimizes the discovery of topological relationships using a three-step approach. The first step minimizes the index size by computing the estimated total hypervolume (eth) for both datasets and indexing the smaller dataset. Since both datasets must be loaded into memory, the space complexity of RADON increases. Due to the asymmetry of certain geospatial relations, RADON applies the inverse relation if datasets are swapped. The second step involves indexing the smaller eth dataset by dividing the space into hypercubes with dynamic height and length, determined by a heuristic granularity factor. Radon's adaptive grid sizes of Radon reduce the number of DE-9IM calculations, especially for geometries in close proximity. The third step utilizes a filter to compute the intersection matrix only if the bounding boxes of two geometries share a relation from the set $R = equals, intersects, contains, inside, covers, coveredBy$. The *disjoint* relation is treated as a special case in Radon and is computed by determining the *intersects* relation and excluding intersecting pairs.

3.2.4 GIA.nt

GIA.nt [Pap+21], introduced by Papadakis et al., adapts Radon's indexing method. Instead of computing the eth, GIA.nt indexes the first dataset using the same grid approach. The memory footprint is reduced by loading the target dataset geometries individually. For each target geometry, GIA.nt computes the MBB and identifies intersecting index tiles. Unlike Radon and Silk, GIA.nt computes all relations, except *disjoint*, simultaneously. It calculates the DE-9IM intersection matrix and adds related geometries to sets for each relation. Pairs that are absent from any set of relationships are considered *disjoint*.

3.2.5 DORIC

Jin et al. present DORIC [Jin+21], which uses space tiling and indexes the data set with the smaller eth, similar to RADON. After checking the bounding-box relations to rule out potential relations, DORIC adds further filters to reduce the DE-9IM calculations. By exploiting the transitivity of relations, DORIC reduces the number of required computations. For example, if geometries a and b are equal and a contains c, it can be inferred that b contains c. Furthermore, DORIC selectively computes parts of the intersection matrix relevant to specific relations.

3.2.6 Strabon

Strabon [KKK12] is an RDF store that supports geospatial queries. It stores geometries using the WKT standard and supports GeoSPARQL [BK11] queries. Strabon and GeoSPARQL use similar relations to the DE-9IM model, with *inside* and *covered by* combined into *within*, and *contains* and *covers* combined into *contains*. Internally, Strabon employs an R-tree-over-GiST system for spatial value storage and efficient spatial query handling.

3.2.7 Geo-L

Geo-L [ZK21], presented by Zinke-Wehlmann and Kirschenbaum, focuses on optimizing the entire lifecycle of geometric link discovery. Unlike other algorithms, Geo-L handles robustness and addresses incomplete and invalid geometries. Geo-L saves partially downloaded datasets and caches them in a database to prevent repeated downloads and reduce data transfer overhead.

3.2.8 MaskLink

MaskLink [San+18], proposed by Santipantakis et al., reduces *disjoint* relation computations. Like other approaches, it uses space tiling to partition the grid and assign source dataset geometries to intersecting tiles. Each tile is given a mask representing the area that does not intersect any previously assigned geometries. If a target geometry lies entirely within the mask, it must be disjoint from all source geometries in the tile, allowing further computations to be avoided.

3.2.9 Line Simplification

Line simplification techniques have been used in various fields, including computer vision, cartography, and computer graphics. [SAH18] introduced an approach to line simplification based on image processing, which is specifically designed for raster data. An area-reserving subdivision simplification algorithm is proposed in [Men18], where the algorithm presents a set of topology constraints to render map data on the screen. In Advanced Driving Assistance Systems (ADAS) [EHZ18], the *Douglas-Peucker* line simplification algorithm is used to fix the total number of vertices for the resultant polygon from static free space extraction, which is convenient for ADAS and automotive restrictions. Recently, *Douglas-Peucker* has been used to simplify the massive Asia-Pacific Data Center trajectories data from the *China Automatic Identification System* (AIS) for data-driven automatic maritime routing [Zha+18a].

3.2.10 Point-Sets Distance Measures

The input to a point-set distance measure is two sets of points. We denote $g_s = (s_1, \ldots, s_n)$ as a sequence of points to describe the source resource geometry g_s and $g_t = (t_1, \ldots, t_m)$ as a sequence of points to describe the target resource geometry g_t. We assume $(n >= m)$, where n resp. m stands for the number of distinct points in the geometry of g_s resp. g_t. A point p_i on the surface of the planet is completely described by two values: its latitude $lat(p_i) = \varphi_i$ and its longitude $lon(p_i) = \lambda_i$. We denote points p_i as pairs (φ_i, λ_i).

The state of the art of LD includes many measures to calculate the distance between the vector descriptions of RDF resources. We base our work in this paper on the LIMES implementation of the *Hausdorff, mean, min, link* and *sumOfMin* point-set distances. Next, we will introduce both the *Hausdorff* and the *mean* distance functions as two examples of such measures. A detailed review of state-of-the-art approaches to point-set distance measures for LD is available at [SN15].

Hausdorff Point-Set Distance

The *Hausdorff point-set distance* [HKR93] is defined as the maximum of the minimum pairwise distances between the two sets of points of source and target geometries. Formally,

$$D_{Hausdorff}(g_s, g_t) = \max_{s_i \in g_s} \left\{ \min_{t_j \in g_t} \left\{ \delta(s_i, t_j) \right\} \right\}, \tag{3.1}$$

where $\delta(s_i, t_j)$ is the minimum distance between two points, s_i and t_j. $\delta(s_i, t_j)$ can be accurately calculated based on the *great distance from the elliptic curve* [Bow84]; however, due to its high time complexity, most LD approaches depend on the *orthodromic distance* for computing $\delta(s_i, t_j)$. The orthodromic distance is formally defined as:

$$\delta(s_i, t_j) = R \cos^{-1} \sin(\varphi_{s_i}) \sin(\varphi_{t_j}) + \cos(\varphi_{s_i}) \cos(\varphi_{t_j}) \cos(\lambda_{s_i} - \lambda_{t_j}), \tag{3.2}$$

where $R = 6371\ km$ is the Earth's radius, assuming the planet to be a perfect sphere.

Mean Point-Set Distance

The *mean* distance function [DHS+73] is one of the most efficient distance measures for point sets with complexity $O(n)$. First, a mean point is computed for each of the source and target point sets. Then, the distance between the two mean points is computed using the orthodromic distance (see Equation 3.2). Formally, the mean distance between the two geometries g_s, g_t is defined as:

$$D_{mean}(g_s, g_t) = \delta \left(\frac{1}{n} \sum_{s_i \in g_s} s_i, \frac{1}{m} \sum_{t_j \in g_t} t_j \right), \tag{3.3}$$

where n and m are the sizes of g_s and g_t, respectively.

3.3 Holistic Models for Linked Open Data

We discuss works related to our work in the areas of data fusion, data integration, and KG embedding. These scholarly works explore aspects such as ensuring data conciseness and completeness, in addition to delving into sophisticated methods to execute link prediction within KGs. The efforts to build a holistic model for LOD are detailed in depth in our work [ASN23], which is the focus of Chapter 7.

3.3.1 Data Fusion

Data fusion is one of the key goals of data integration. Data fusion increases conciseness by fusing duplicate entries and merging common attributes. The work [BN09] defines the goals of data fusion to achieve more completeness and conciseness of data. The main challenges of data fusion are uncertainty due to conflicting data values. In [BN09], the authors discuss different ways of data fusion and present several methods. In the systematic survey [NVJ20], the authors introduce the challenges of KG fusion, discussing advanced techniques for handling KG fusion. The linked data quality assessment and fusion framework SIEVE [MMB12] is based on the linked data integration framework (LDIF) [Sch+11]. LDIF is an open-source framework that provides data translation and identity resolution while keeping track of data provenance.

3.3.2 Knowledge Graph Embedding

In recent years, dozens of Knowledge Graph Embedding (KGE) techniques have been developed to address tasks such as graph completion, question answering, and link prediction [Hua+19; Lin+15; NRP; NTK11; Tro+16]. For example, RESCAL [NTK11] computes a factorization in three directions of an adjacency tensor representing the input KG. The adjacency tensor is decomposed into a product of a core tensor and embedding matrices. HolE [NRP] uses circular correlation as its compositional operator.

On the other hand, TransE [Bor+13] is an energy-based KGE model in which a relation r between entities h and t corresponds to a translation of their embeddings, that is, $h + r \approx t$ provided that (h, r, t) exists in the KG. More details on KGE approaches and applications can be found in [Wan+17].

3.4 Explainable Link Discovery

Explainable artificial intelligence (XAI) has gained widespread recognition as an indispensable aspect of modern AI technology, particularly in settings that demand transparency and interpretability. This pertains especially to semantic data and its conversion into natural text. LD and LS systems, the focal points of our study, are no exception. A comprehensive survey by Barredo Arrieta et al. [Bar+20] highlights the emerging field of XAI, underscoring its various dimensions and critical significance. In declarative LD such as LIMES, EAGLE [NL12] has addressed the readability of LS, alongside accuracy and efficiency. However, the generated LS is still expressed in a declarative manner. Recently, the WOMBAT algorithm [SNL17] has implemented a machine learning algorithm for automatic LS finding by using generalization via an upward refinement operator. However, none of these works addressed the verbalization of LS in terms of natural language description.

3.4.1 Template-based Explainable Link Discovery

Despite substantial advances in Explainable Artificial Intelligence (XAI) [Bar+20], the task of converting LS into human-readable natural language has received less attention. Within this context, data-to-text systems [Rei07], a subclass of NLG, can produce textual output from various nonlinguistic inputs such as sensor data, logs, or LS. Rule-based NLG methods have proven their adaptability in numerous applications, ranging from the textual representation of graphical weather data to the summarizing of numerical databases and the simplification of complex medical information. In our initial work on explainable LD, we turned to rule-based NLG systems that adhere to the frameworks developed by Reiter & Dale [RD00] to translate LS into comprehensible English text. Our current research builds on this foundation and broadens the scope of rule-based NLG by extending it to a multilingual setting, incorporating not only English but also German and Spanish. This expansion improves both the interpretability and the applicability of LS across varied linguistic landscapes [ASN19a]. These efforts are detailed in depth in Chapter 8.

3.4.2 Neural Machine-based Explainable Link Discovery

Neural Machine Translation (NMT) has undergone a transformative evolution in recent years [KB13; BCB15; Cho+14b; SVL14a; Cho+14a]. These influential works frequently employ robust neural architectures such as Convolutional Neural Network (CNN) and biLSTM to tackle a variety of complex tasks. Such tasks range from the extraction of video features to complex text generation operations [Don+17]. In the context of our research, we find that our efforts align seamlessly with the overarching narrative of post hoc explainability. Our focus is on generating text-based explanations that are both practical and scientifically grounded, as corroborated by the existing literature [Bar+20]. A key application of NMT in our work is related to the translation of logical expressions, or LS, into human-readable natural language. For this, we employ a variety of NMT techniques, including standard sequence-to-sequence (*seq2seq*) models equipped with Gated Recurrent Unit (GRU) and both unidirectional and bidirectional LSTM networks. Furthermore, we incorporate transformer architectures to further enhance the user experience for those engaged with LD applications, where LS forms the foundational backbone. It should be noted that we leverage these state-of-the-art technologies not just as a means to enrich our dataset but also to dive deeper into the multifaceted challenges and potential solutions associated with making LD more explainable. These efforts are detailed exhaustively in our works [ASN19a; Ahm+21], which are the focus of Chapter 8.

3.4.3 Large Language Model-based Explainable Link Discovery

The emergence of transfer learning and pre-training methodologies has notably shifted the landscape of NLP, opening up new avenues for tackling complex problems. The use of (LLM) such as T5 [Raf+19] has been instrumental in this regard, demonstrating impressive results in a multitude of NLP tasks. These tasks range from sentiment analysis and question answering to more nuanced challenges, such as language translation and summarizing [Raf+19]. The prowess of LLMs is further augmented when integrated with sophisticated neural architectures, including, but not limited to, bidirectional Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) models equipped with attention mechanisms [Mou+19]. These advanced architectures enhance the model's capability to offer nuanced and in-depth explainability features. Within the context of our research, we leverage these cutting-edge technologies not only to enrich our dataset but also to provide a more comprehensive and insightful understanding of the challenges and solutions associated with the explainability of LD in our work [ASN19a; Ahm+21] that are explained in Chapter 9.

Part II

Improving Link Discovery Scalability

To enable the efficient development of proper links between KGs, a linking framework (i.e., LD tool) must support high effectiveness and efficiency. In this part, we focus on linking tools (for example, LIMES [NA11; Ngo+21]) that fall within the declarative representation paradigm. However, we pay more attention to the efficiency (i.e., scalability) challenge in LD over geospatial KGs.

- Effectiveness. A linking tool should provide high-quality mappings using established metrics such as accuracy, recall, and F-measure. As a result, the output must be precise; i.e. the relationships generated by a certain framework must be correct (precision). To be comprehensive, a link generator should generate as many links as possible. Finally, only links between resources that truly belong together should be established. *Nentwig et. al.* [Nen+17] have surveyed many tools considering the effectiveness of a linking framework.
- Efficiency. A linking tool should be rapid and scalable to large KGs with hundreds of thousands or millions of resources. A naive, non-scalable approach evaluates all potential resource pairings (Cartesian product), resulting in a quadratic complexity. As a result, one of the key efficiency goals is to reduce the search space so that superfluous resource pairs are avoided. *Nentwig et. al.* [Nen+17] have listed many tools dealing with the efficiency of a linking framework.

To address the efficiency challenge in LD, we present a set of novel approaches in this part of the thesis. These approaches facilitate the integration of enormous volumes of KGs while working within time or space restrictions. We also introduce techniques for connecting resources based on spatial links, which have received minimal attention in the literature.

4 LineSimp: A Line Simplification Approach For Link Discovery over Geospatial Knowledge Graphs

Preamble: This chapter is based on Ahmed et al. [ASN18b]. It studies the effect of simplifying the resource's geometries on the runtime and F-measure of LD approaches. In particular, we evaluate LD approaches for computing point-set distances, as well as the topological relations among RDF resources with geospatial representation. The results obtained on two different real datasets suggest that most geospatial LD approaches achieve a speedup of up to $67\times$ using simplification, while the average loss in their F-measure is less than 15%.

4.1 Motivation

With the increasing growth of Linked Data in geospatial resources over recent years comes the need to develop highly scalable approaches for discovering links among such resources. As pointed out in previous work [Ngo13], only 7.1% of the links between resources connect geospatial entities. This is due to two main factors: 1) the large number of resources with geospatial representation available on the LOD, which requires scalable algorithms for computing links between geospatial resources. For example, LGD[1] contains more than 20 billion triples that describe millions of geospatial entities [ALH09]. 2) The vector representation of geospatial resources demands the computation of particular relations, i.e., distance and topological relations between geospatial resources. For example, finding the nearby point of interest within a given radius.

[1] `http://linkedgeodata.org`

According to the Linked Data principles[2], the provision of links between KGs in RDF[3] is of central importance for numerous SW tasks. However, the LD process becomes more challenging, especially when dealing with geospatial resources in real-time applications, including structured machine learning [SNL17], question answering [Leh+12], and data fusion [SNL15]. In such a real-time application, the provision of explicit geospatial relations among resources is of central importance for achieving scalability.

Only a few state-of-the-art approaches have been developed for LD to deal with geospatial data represented in RDF. For example, [Ngo13] uses the *Hausdorff* distance to compute the distance between geospatial entities. A survey of 10 point-set distance measures for LD is provided in [SN15]. Based on the *MultiBlock*, SILK [SK16] computes topological relations according to the DE-9IM standard. Recently, RADON [She+17] has provided an indexing method combined with space tiling that enables efficient computation of topological relations between geospatial resources.

To the best of our knowledge, no previous work has studied the problem of discovery of geospatial relations among a simplified version of vector representations of geospatial resources. In this chapter, we study the effect of applying two line-simplification algorithms as a preprocessing step prior to the discovery of geospatial relations among such resources. In particular, we consider the effect of simplification upon both the efficiency of discovered relations (i.e., F-measure) and the scalability of the LD approaches (i.e., runtime). The contributions of this chapter are as follows:

1. We present and formalize the problem of LD for geospatial resources as well as the line simplification problem.

2. We study the effect of simplifying the geospatial representation of resources on the quality of discovered relations.

3. We study the speedup of various LD approaches when dealing with RDF resources with simplified geometries.

4. We present an evaluation of two line-simplification approaches for different LD approaches and show that while such approaches only lose, on average, 15% F-measure on the original data, they gain up to 67× speedup when applied to the simplified data.

[2] `https://www.w3.org/DesignIssues/LinkedData.html`

[3] Resource Description Framework, see `https://www.w3.org/RDF/`

The rest of this chapter is structured as follows. We begin by introducing our approach in Section 4.2. Then, in Section 4.3, we present our evaluation and results.

4.2 Approach

The methodology we adopt is focused on LD using the DE-9IM[CSE94] and Point-Sets Distance Measures with line simplification techniques. A formal definition of LD is available in Chapter 2, Section 2.4. Works that are related to this topic are discussed in Chapter 3, specifically in Sections 3.2 and 3.2.10.

4.2.1 Line Simplification

Line simplification (dubbed *curve simplification* in some literature) has been adopted in many fields, including computer vision, cartography, and computer graphics. The input to a line simplification algorithm is a polygonized curve with n vertices composed of line segments (also called a polyline in some contexts). The goal of a line simplification algorithm is to find an approximating polygonized curve with m vertices as output, where $m < n$. A closely related problem is to take a line with n vertices and approximate it within a defined error tolerance $\epsilon > 0$. In this work, we introduce only the *Douglas-Peucker* and *Visvalingam–Whyatt* algorithms as case studies due to their popularity. A detailed review of the line simplification algorithms can be found in this survey [HG97].

The Douglas-Peucker Algorithm

The *Douglas-Peucker* algorithm [DP73] is the most widely used high-quality curve simplification algorithm. It was invented independently by many authors. At each iteration, the *Douglas-Peucker* algorithm tries to approximate a sequence of points by a line segment from the first point to the last point. As shown in Algorithm 1, the algorithm starts with the two endpoints of the input polyline. Then, it finds the point with the farthest distance d from the line segment formed by the current start and endpoints. If d is below the simplification factor $dmax$, the approximation is accepted; otherwise, the algorithm is recursively applied to the two polylines before and after the chosen point. The *Douglas-Peucker* algorithm, though not optimal, has generally been invented to generate the highest subjective and objective quality approximations when compared with many other heuristic algorithms. Its best-case time cost is $\Omega(n)$, its worst-case cost is $O(mn)$, and its expected time cost is about $\Theta(n \log m)$. The worst-case behavior can be improved, with some sacrifice in the best-case behavior, using a $\Theta(n \log)$ algorithm employing convex hulls [HG97].

Algorithm 1: Douglas-Peucker-Algorithm

```
Result: return ResultList[]
function DouglasPeucker(PointList[], epsilon);
dmax = 0;
index = 0;
for i = 2 to (length(PointList) - 1) do
    d = PerpendicularDistance(PointList[i], Line(PointList[1], PointList[end]));
    if d > dmax then
        index = i ;
        dmax = d;
if dmax > epsilon then
    recResults1[] = DouglasPeucker(PointList[1...index], epsilon);
    recResults2[] = DouglasPeucker(PointList[index...end], epsilon);
    ResultList [] = recResults1[1...end-1] recResults2[1...end];
else
    ResultList[] = PointList[1], PointList[end];
```

The Visvalingam-Whyatt Algorithm

Visvalingam-Whyatt algorithm [VW93] (see Algorithm 2) uses the concept of *effective area* for progressive simplification of a line-by-point elimination. The basic idea behind this algorithm is to iteratively eliminate the fewer characteristic points, i.e., those which produce the least areal displacement from the current part-simplified line. The algorithm filters points on lines by a process of elimination rather than selection, while the *Dougherty-Peucker* Algorithm keeps the points on curves by selecting points rather than eliminating them. To remove points, *Visvalingam-Whyatt* iteratively computes the area of all triangles formed by each three successive points. If the area of the smallest triangle is smaller than a threshold (area-tolerance), then its middle point is deleted.

Algorithm 2: Visvalingam-Whyatt Algorithm

```
Result: $L$
Input line $L$ as a list of points, separate list $R$ of ranked points;
Compute the effective area of each point on the line;
Delete all points with zero area and store them in a separate list;
for do
    Find the point with least effective area and call it current point;
    Delete the current point from the original list $L$ and add it to the ranked list
      $R$ with its effective area;
    Recalculate the effective area of the two adjacent points;
    if Size of $L = 2$ then
        Terminate
    end
end
```

4.3 Evaluation

We have now prepared all of the ingredients needed for our study. We study the impact of line simplification algorithms on the main requirements (i.e., efficiency and runtime) of LD over RDF KGs containing geospatial entities. We evaluate the effect of simplifying geometries on the approaches used thus far in geospatial LD: point-set measures (e.g., Hausdorff and mean measures) and topological relations (e.g., `contains` and `overlaps` relations).

We aimed to answer four questions with our experimental evaluation:

Q_1 How much performance (i.e., F-measure) does each of the geospatial LD approaches lose when dealing with the simplified geometries vs. when dealing with the original ones?

Q_2 How well does each of the geospatial LD approaches scale (i.e., runtime and speedup) in the context of simplified geometries?

Q_3 Which is most affected relation and less affected relation by the simplification process?

Q_4 What is the runtime cost of simplification?

4.3.1 Experimental Setup

Hardware All of the experiments were carried out on the *OCuLUS* cluster running OpenJDK 64-Bit 1.8.0_161 on Ubuntu 16.04.3 LTS. *OCuLUS* is a high-performance machine located at the computer science institute on the main campus of *Paderborn University*. It consists of $9,920$ processor cores, 2.6 GHz *Intel Xeon "Sandy Bridge"*, with a main memory of capacity 45 TB. For our created jobs, we assigned 16 CPUs and 200 GB of RAM for each job with a termination time of 4 hours.

LIMES For our experiments, we selected the LD framework LIMES [NA11] to study the impact of the line simplification algorithms on the discovery of links between RDF resources with geospatial representation. We selected LIMES as it implements the time-efficient approach RADON [She+17] for the discovery of topological relations and also because it implements various point-set distance functions [SN15].

Datasets We evaluated our approach using two real-world datasets.
(1) Nomenclature of Territorial Units for Statistics (NUTS)[4] is manually curated by the *Eurostat group of the European Commission*. NUTS contains a detailed hierarchical description of whole European regions. (2) CLC is an activity of the European Environment Agency (EEA) that collects data regarding the land cover of European countries. CLC contains 44 sub-datasets ranging in size from 240 resources to $248,242$ resources.[5] We merged all CLC sub-datasets into one big dataset of $2,209,538$ resources that we dubbed CLC. As LIMES can only read geometries in the WKT format, we adopted the same preprocessing technique proposed by [She+17]. In particular, we preprocessed NUTS and CLC by converting `ngeo:posList` serialization into WKT, and lines larger than 64 KB were trimmed.

4.3.2 F-measure Analysis

To evaluate the F-measure, we conducted four sets of experiments as follows:

In *the first set of experiments*, we used the RADON approach with the same setting as in [She+17] to discover the relations `equals`, `intersects`, `contains`, `covers`, `coveredBy`, `touches`, `crosses`, and `overlaps`. We used the NUTS dataset as the source dataset and CLC as the target. We then tested the impact of the line

[4] Version 0.91 (`http://nuts.geovocab.org/data/0.91/`) is used in this paper

[5] For more details about CLC, see `https://datahub.io/dataset/corine-land-cover`

simplification algorithm of *Douglas-Peucker* [DP73] on RADON's performance (that is, the F-measure) when applied to the simplified data. To generate the simplified data, we applied the simplification factors of $0.05, 0.09, 0.10$, and 0.2. Given that RADON is complete [She+17] (i.e., RADON always achieved an F-measure of 1), we used the results generated by applying RADON to the original data set as our reference data set. Using such a reference dataset, we were able to calculate the F-measures presented in Table 4.1. Our results show a reverse correlation between the simplification factor and the F-measure. On average, RADON was able to achieve the 0.94 F-measure when applied against the simplified geometries. This answers Q_1 for the LD of the topological relations when applied to simplified geometries using the *Douglas-Peucker* algorithm.

Tab. 4.1.: F-measure results of applying RADON against geometries generated using the *Douglas-Peucker* line simplification algorithm. R/F= Relation/Factor, Avg.= Average.

R/F	0.05	0.09	0.10	0.20	Avg.
Equals	1.00	1.00	1.00	1.00	1.00 ± 0.00
Intersects	0.99	0.97	0.97	0.94	0.97 ± 0.02
Contains	0.99	0.97	0.97	0.93	0.97 ± 0.03
Within	0.99	0.97	0.97	0.93	0.97 ± 0.03
Covers	0.99	0.97	0.97	0.93	0.97 ± 0.03
Coveredby	0.99	0.97	0.97	0.93	0.97 ± 0.03
Crosses	1.00	1.00	1.00	1.00	1.00 ± 0.00
Touches	1.00	1.00	1.00	1.00	1.00 ± 0.00
Overlaps	0.80	0.52	0.47	0.28	0.52 ± 0.21
Avg.	0.97 ± 0.07	0.94 ± 0.16	0.93 ± 0.17	0.90 ± 0.23	0.94 ± 0.03

Using the same setting, we ran our *second set of experiments*, where we used the *Visvalingam-Whyatt* [VW95] algorithm to simplify the geometries. The results, as shown in Table 4.2, show that the selection of the simplification parameter is more critical to the *Visvalingam-Whyatt* algorithm. Using the smallest simplification factor of 0.005 leads to the best results with an average F-measure of 0.97. Also, the reverse correlation between the simplification factor and the F-measure still holds. Those results answer Q_1 for LD approaches for topological relations when applied to simplified geometries using the *Visvalingam-Whyatt* algorithm.

For the *third set of experiments*, we performed a deduplication task for the whole NUTS data set. That is, we set the NUTS data set as the source S and target T data sets. To measure how well each of the point distance measures performs, we first created a reference mapping $M = \{(n, n) \in NUTS\}$, then measured the distance between each of the geometries in $S \times T$. We then calculated the F-measure achieved within the experiment by comparing the pairs in M' (generated by applying the

Tab. 4.2.: F-measure results of applying RADON against geometries generated using the *Visvalingam-Whyatt* line simplification algorithm. R/F= Relation/Factor, Avg.= Average.

R/F	0.005	0.05	0.09	Avg.
Equals	1.00	1.00	1.00	1.00 ± 0.00
Intersects	0.86	0.01	0.00	0.29 ± 0.49
Contains	0.86	0.01	0.00	0.29 ± 0.49
Within	0.86	0.01	0.00	0.29 ± 0.49
Covers	0.86	0.01	0.00	0.29 ± 0.49
Coveredby	0.86	0.01	0.00	0.29 ± 0.49
Crosses	1.00	1.00	1.00	1.00 ± 0.00
Touches	1.00	1.00	1.00	1.00 ± 0.00
Overlaps	0.86	0.03	0.00	0.30 ± 0.49
Avg	0.94 ± 0.08	0.56 ± 0.52	0.56 ± 0.53	0.69 ± 0.22

point-set distances) with those in M. We used the implementations of the *Hausdorff, Mean, Min, Link Sum of minimums,* and *Surjection* from LIMES. The results of these experiments are listed in the last column of Table 4.3. We then used the *Douglas-Peucker* algorithm to simplify all the NUTS geometries with simplification factors $\{0.05, 0.9, 0.1, 0.2\}$. As shown in Table 4.3, the simplification factors of 0.1 and 0.2 achieved the best average F-measure of 0.82. One of the most interesting results of these experiments was that most of the measures of the set of points were able not only to achieve the same F-measure of the original data set when applied to the simplified data but also to outperform the F-measure in the original data in the cases of the *Hausdorff, Mean*, and *Min*.

Tab. 4.3.: Average F-measure results of applying a deduplication task on the NUTS dataset using the point-set distance measures implementations in LIMES. As input, we used both the original NUTS geometries (results are in the last column) and simplified geometries generated using the *Douglas-Peucker* line simplification algorithm. M/F = Measure/Factor, Avg.= Average,F_o= original F-measure

M/F	0.05	0.9	0.1	0.2	Avg.	F_o
Hausdorff	0.90	0.91	0.91	0.91	0.91	0.88
Mean	0.94	0.94	0.94	0.94	0.94	0.94
Min	0.14	0.16	0.16	0.21	0.18	0.13
Link	0.95	0.95	0.94	0.94	0.94	0.94
SumOfMin	0.95	0.95	0.94	0.94	0.94	0.94
Avg.	$0.77{\pm}0.36$	$0.78{\pm}0.35$	$0.78{\pm}0.35$	$0.79{\pm}0.32$		$0.77{\pm}0.36$

In the *fourth set of experiments*, we used the same setting as the last set of experiments except for the simplification algorithm. In this set of experiments, we generated a simplified version of the NUTS geometries using the *Visvalingam-Whyatt* algorithm with the simplification factors $\{0.005, 0.05, 0.1\}$. We then calculated the F-measure based on the following distance measurement functions *Hausdorff, Mean, Min, Link,* and *Sum of Minimums*. The results in Table 4.4 show the comparison between the F-measure obtained from a simplified version of the data and the F-measure (denoted $F_{original}$) obtained from an original version of the data. The results clearly show the sensitivity of the F-measure to changing the simplification factor. For example, when the simplification factor is equal to 0.005, the average of the F-measure is 0.77, while it drops to 0.26 with a simplification factor of 0.1.

Tab. 4.4.: F-measure results of applying a deduplication task on the NUTS dataset using the point-set distance measures implementations in LIMES. As input, we used both the original NUTS geometries (results are in the last column) and simplified geometries generated using the *Visvalingam-Whyatt* line simplification algorithm. M/F = Measure/Factor, Avg.= Average, F_o= original F-measure.

M/F	0.005	0.05	0.1	Avg.	F_o
Hausdorff	0.88	0.24	0.02	0.38 ± 0.45	0.88
Mean	0.94	0.94	0.92	0.93 ± 0.02	0.94
Min	0.13	0.13	0.13	0.13 ± 0.00	0.13
Link	0.94	0.14	0.01	0.37 ± 0.51	0.94
Sum of Min	0.94	0.94	0.24	0.71 ± 0.40	0.94
Avg.	0.77 ± 0.36	0.48 ± 0.42	0.26 ± 0.38		0.77 ± 0.36

4.3.3 Runtime Analysis

In order to answer Q_2, we evaluated the speedup gained by applying LD approaches to simplified geometries. We measured the runtime while performing the four sets of experiments mentioned earlier.

Figure 4.1 shows the runtime results for the first set of experiments, that is, we measured the runtime of applying RADON to discover topological relations when applied to the original datasets vs. when applied to the simplified datasets using the *Douglas-Peucker* algorithm. On average, RADON provided a $4.9\times$ speedup in its performance when applied to the original datasets. Moreover, there is a direct correlation between the speedup achieved and the simplification parameter. In particular, the lowest speedup of $3.7\times$ is achieved when applying the simplification factor of 0.05, and the speedup monotonically increases to $6.1\times$ when applying the simplification factor of 0.2.

For the second set of experiments, we also measured the runtimes when applying RADON against the original data sets vs. the simplified datasets using the *Visvalingam-Whyatt* algorithm. The results are shown in Figure 4.2. On average, RADON achieved

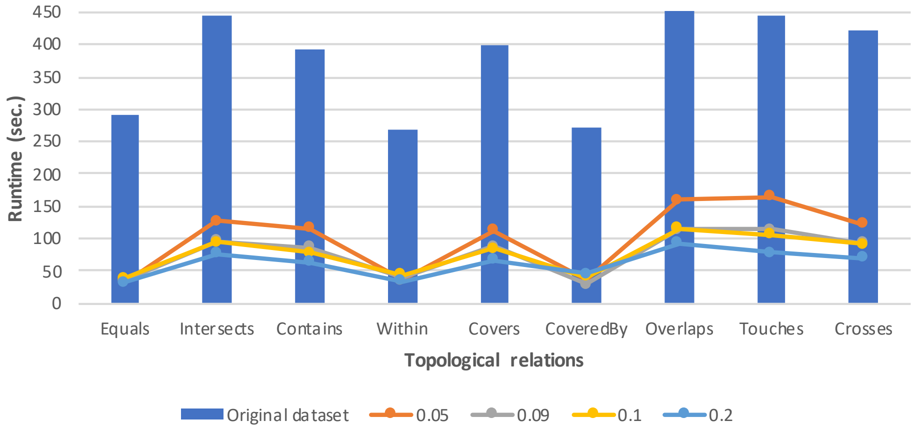

Fig. 4.1.: Runtimes of RADON's implementation of topological relations LD for original $NUTS \times CLC$ datasets vs. the runtimes of the simplified datasets using the *Douglas-Peucker* algorithm with simplification factors of $\{0.05, 0.09, 0.1, 0.2\}$.

$49.2\times$ speedup, with a maximum $67.3\times$ speedup in the case of a simplification factor of 0.09 but only $4.2\times$ speedup with a simplification factor of 0.005.

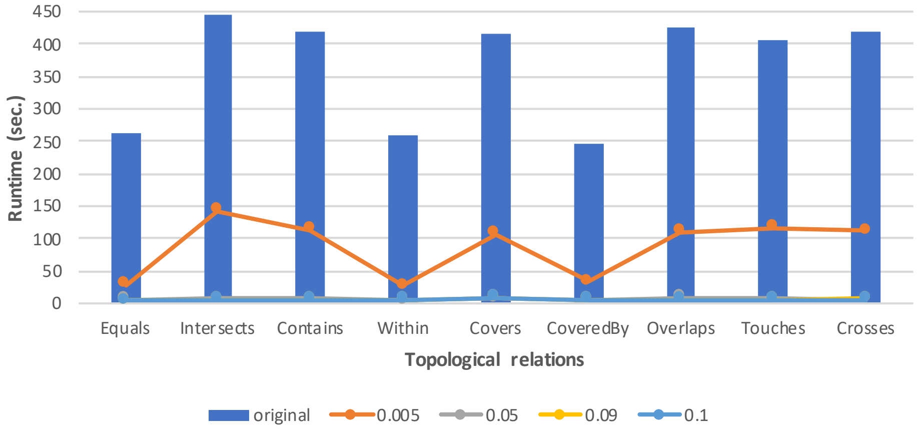

Fig. 4.2.: Runtimes of RADON's implementation of topological relations LD for original $NUTS \times CLC$ datasets vs. runtimes of the simplified datasets using the *Visvalingam-Whyatt* algorithm with simplification parameters $\{0.005, 0.05, 0.1\}$.

Using the same technique, we measured the runtime for the third set of experiments. The results are presented in Figure 4.3. The distance of the point-set achieved an average speedup of 9.2 when applied to the simplified geometries using the *Douglas-Peucker* algorithm (min. $= 2\times$, max. $= 19.8\times$).

Figure 4.4 shows the results of the run-times of the fourth set of experiments. Point-set measures achieved only an average speed increase of 2.1 when applied to simplified geometries using the *Visvalingam-Whyatt* algorithm (only 1.1 in the case of a simplification factor of 0.005).

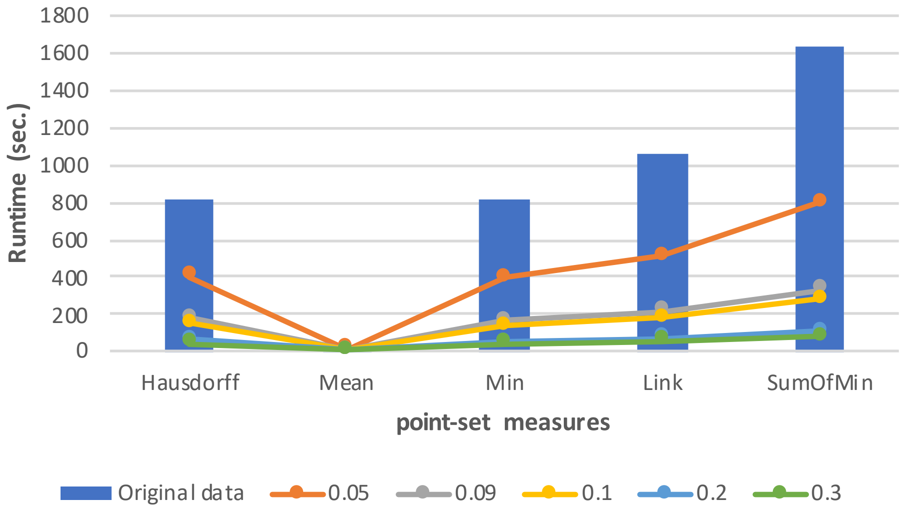

Fig. 4.3.: Runtimes of LIMES implementation of point-set measures LD deduplication for the original $NUTS$ dataset vs. the runtime of the simplified dataset using the *Douglas-Peucker* algorithm with simplification parameters $\{0.05, 0.09, 0.1, 0.2, 0.3\}$.

4.3.4 LD Relations Analysis

From all previous sets of experiments, we can now answer Q_3. In the case of the topological relations, the F-measure of the `overlap` relation is the most affected by the *Douglas-Peucker* simplification. This can be seen in Table 4.1. Also, the `equals`, `crosses`, and `touches` are not affected at all by any simplification (see Tables 4.1 and 4.2). In the case of point-set measures, the F-measure of the `min` measure is the most affected, while all the other relations not only achieve the F-measure of the original data but also outperform it in many cases (see Tables 4.3 and 4.4).

For the runtime of topological relations, the `equal` relation achieved the best speedup in the case of the *Douglas-Peucker* simplification (see Figure 4.1), while the `coverdBy` had the best speedup when using the *Visvalingam-Whyatt* algorithm (see Figure 4.2). For the runtime of the point-set relations, the `mean` relation achieves the shortest runtime even without any simplification, while the `sunOfMin` relation has the least speedup (see Figures 4.3 and 4.4).

4.3.5 Simplification Runtime Analysis

To answer Q_4, we measured the runtime cost of applying both the *Douglas-Peucker* and *Visvalingam-Whyatt* line simplification algorithms. We calculated the average simplification time needed while performing the first and second sets of experiments

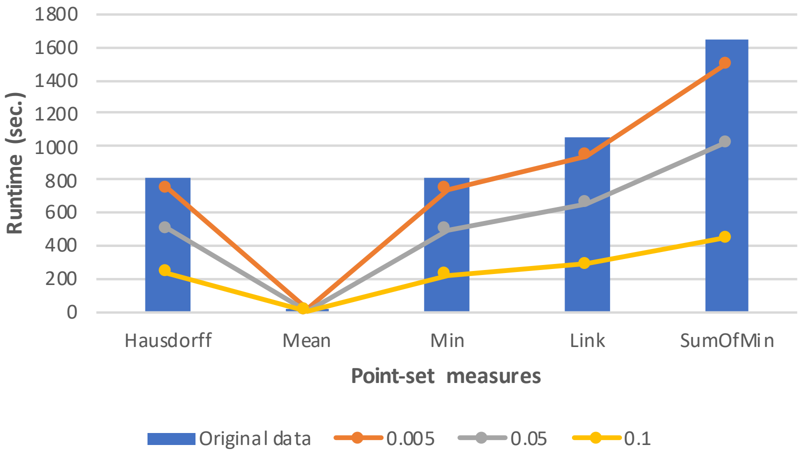

Fig. 4.4.: Runtimes of LIMES implementation of point-set measures LD deduplication for the original $NUTS$ dataset vs. the runtime of the simplified dataset using the *Visvalingam-Whyatt* algorithm with simplification parameters $\{0.005, 0.05, 0.1\}$.

together with the average time to run RADON for each of the topological relations versus the time needed to run RADON against the original data sets. The results are detailed in Figure 4.6 and Figure 4.5.

In case we want to discover all the topological relations at once, Figure 4.5 shows the total runtime needed for simplification on the left. Note that the simplification process is done only once for all topological relations. The total time to run RADON for all relations in the simplified data is plotted next in the figure. Next comes the total time of running RADON in addition to the simplification time for all relations. Finally, the time to run RADON in the original data is plotted. As we can see, as we perform the simplification process once and use the simplified data to extract all the relations, RADON is able to achieve on average $2.4\times$ speedup.

Figure 4.6 shows the average runtime needed for running RADON for only one relation on the original data vs. running it on a simplified one. As we can see, the simplification time is, on average, greater than the average time for a single RADON topological relation discovery task (see the first and last columns in Figure 4.6). This clearly shows that using simplification for the discovery of a single relation is suboptimal.

A complete answer for Q_4 would be that the more relations that need to be discovered, the faster the gains will be obtained from the use of simplification. Moreover, the simplification runtime cost would be very high once a single relation discovery is required, i.e., we recommend not using any simplification for a single relation discovery.

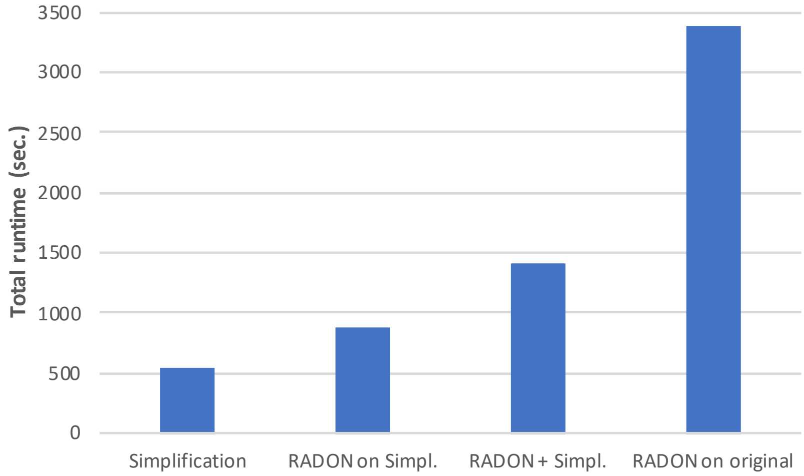

Fig. 4.5.: Total runtime for all topological relations. Runtimes of RADON on the original data vs. simplified data using the *Douglas-Peucker* algorithm.

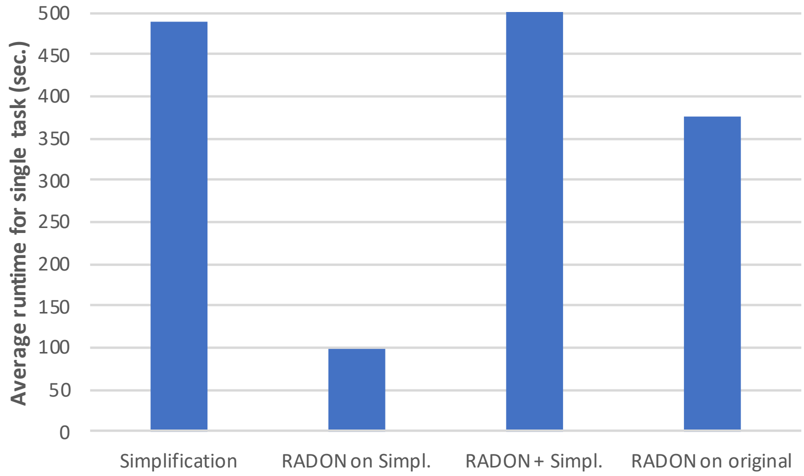

Fig. 4.6.: Average runtime of a single topological relation. Runtimes of RADON on the original data vs. simplified data using the *Douglas-Peucker* algorithm.

5 RADON2: An Intersection Matrix Approach For Link Discovery over Geospatial Knowledge Graphs

Preamble: This chapter is based on Ahmed et al. [ASN18c], in which RADON2 is presented. RADON2 addresses the LD problem with geospatial KGs, as it is at the essence of the SW, where a KG such as LGD consists of more than 20 billion facts [ALH09]. Reasoning on these considerable amounts of geospatial KGs lacks efficient methods to calculate the links between the resources contained in these KGs. RADON2 is the extension of the RADON algorithm, where we compute all the topological relations of DE-9IMin parallel to accelerate LD between geospatial resources. Our evaluation shows that RADON2 outperforms the state of the art significantly with up to 35% improvement in runtime.

5.1 Motivation

In Chapter 4, we began our examination of scalability issues in LD over geospatial KGs by investigating the role of line simplification algorithms. Specifically, we studied how these algorithms can serve as preprocessing steps to enhance the discovery of geospatial relations among simplified versions of vector representations of geospatial resources. We considered the impact of these algorithms on both the efficiency of the discovered relations, measured by the F-measure and the runtime scalability of LD approaches.

In contrast, the present chapter (Chapter 5) shifts the focus toward exploring the untapped potential of IM for the same LD tasks. More specifically, we study the impact of IM and its parallelization on the discovery of topological relationships between geospatial entities. We aim to understand how leveraging parallel IM computation can enhance the scalability of LD approaches, particularly in terms of runtime.

Furthermore, we introduce an extension of the RADON algorithm, originally presented in [She+17]. This algorithm uses an indexing method combined with space tiling to allow efficient computation of topological relations between geospatial resources, based on the DE-9IM standard. By diversifying our focus between line simplification and IM across different chapters, we aim to offer a more rounded perspective on improving the scalability of LD in geospatial KGs.

The contributions of this chapter are as follows:

1. We study the effect of parallel computing of all topological relations between the geospatial resources introduced in DE-9IM at once.

2. We study the impact of the number of processors on the runtime of IM computing.

The rest of this chapter is structured as follows. We begin by introducing our approach in Section 5.2. Then, in Section 5.3, we present our evaluation and results.

5.2 Approach

The basic idea behind the RADON [She+17] approach to topological relation discovery is to provide an indexing method combined with space tiling that allows efficient computation of topological relations between geospatial resources. In particular, RADON presents a novel sparse index for geospatial resources. Then, based on the bounding boxes of the indexed geospatial resources, RADON applies a strategy to eliminate unnecessary computations of DE-9IM relations. In this work, our concern is focused on the computing of IM used in the DE-9IM standard. Initially, in RADON, each topological relation is individually computed by selecting the relation first and then calculating the intersection matrix, while in this extension, we first compute the intersection matrix and, according to the mask of each topological relation, we define the relation. In particular, we buffer the result of IM of each pair of geometries so that all topological relations of the same pair can be retrieved without the need to recompute their respective IM again. For instance, if the $IM = [T * F * *FFF*]$, where T is true, F is false, and $*$ is not care. To this end, the IM is buffered and then compared to the mask of all topological relations, such as `equals`, which has the mask $mask = [T * F * *FFF*]$. By applying this computing strategy, we can gain the time used during the computation of each individual relation, which leads to a significant impact. Especially, when the geospatial resource is represented by thousands of spatial points, as we can see clearly in today's geospatial RDF KGs (e.g., NUTS[1] and CLC[2]). Moreover, calculating IM at once for all topological relations does not affect the accuracy of linking, i.e., the F-measure. Works that are related to this topic are discussed in Chapter 3, specifically in Sections 3.2 and 3.2.3.

5.3 Evaluation

We have now prepared all the ingredients needed for our extension of the RADON algorithm. Since the F-measure does not change, we study only the impact of intersection matrix computing on the second main requirement (i.e., runtime) of LD over RDF KGs containing geospatial entities. We evaluated the impact of intersection matrix computing in the RADON2 versus RADON algorithm and then the effect of processor number on the LD performance when we applied the proposed approach. We aimed to answer the following questions with our experimental evaluation:

Q_1 How well does the geospatial LD approach presented in RADON2 scale (i.e., runtime speedup) in case of parallel computing of IM?

Q_2 Does the processor number have a significant impact on IM computing?

5.3.1 Experimental Setup

Hardware All the experiments were carried out on the *OCuLUS* cluster running *OpenJDK* 64-Bit 1.8.0_161 on *Ubuntu* 16.04.3 *LTS*. *OCuLUS* is a high-performance machine located at the computer science institute on the main campus of *Paderborn University*. It consists of 9,920 processor cores, 2.6 GHz *Intel Xeon* "*Sandy Bridge*", with a main memory of capacity 45 TB. For our created jobs, we assigned 16 and 32 CPUs and 20 GB of RAM for each job with a termination time of 4 hours.

Datasets We evaluated our approach using two real-world datasets. (1) NUTS[3] is manually curated by the *Eurostat group of the European Commission*. NUTS contains a detailed hierarchical description of whole European regions. (2) CLC is an activity of the EEA that collects data on the land cover of European countries. CLC contains 44 subdatasets ranging in size from 240 to 248,242 resources.[4] We merged all CLC

[1] http://nuts.geovocab.org/data/0.91/

[2] https://datahub.io/dataset/corine-land-cover

[3] Version 0.91 (http://nuts.geovocab.org/data/0.91/) is used in this paper

[4] For more details about CLC Cover, see https://datahub.io/dataset/corine-land-cover

subdatasets into one big dataset of $2,209,538$ resources (dubbed CLC). As LIMES can only read geometries in the WKT format, we adopted the same preprocessing technique proposed by [She+17]. In particular, we preprocessed NUTS and CLC by converting `ngeo:posList` serialization into WKT.

5.3.2 Runtime Analysis

To answer Q_1, we conducted an experiment to measure the run time of the RADON2 vs. the RADON algorithm using NUTS as a source dataset and CLC as a target dataset. As shown in Figure 5.1, the overall speedup in runtime is up to 28% for all topological relations `equals`, `within`, `contains`, `disjoint`, `touches`, `meets`, `covers`, `coveredBy`, `intersects`, `crosses` and `overlaps`.

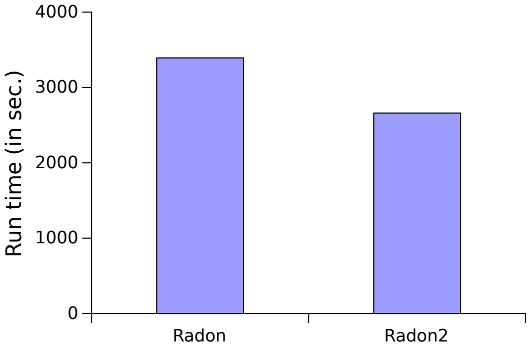

Fig. 5.1.: Runtimes of RADON2 and RADON for topological relation discovery among $NUTS \times CLC$ datasets.

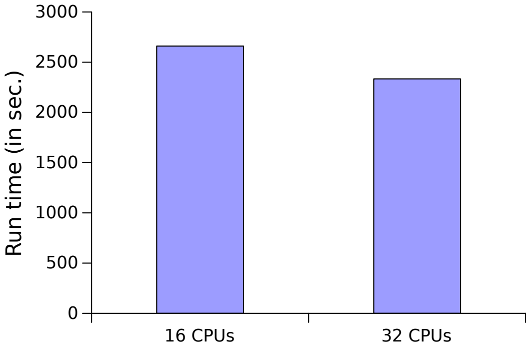

Fig. 5.2.: Runtimes of RADON2's implementation using 16 CPUs of topological relations LD for $NUTS \times CLC$ datasets vs. the runtime of the RADON2's implementation using 32 CPUs.

To answer Q_2, we conducted a second experiment to study the impact of the processor number on IM computing. We kept the same setting in the first experiment, except for the number of processors, which is doubled in this experiment. The result in Figure 5.2 shows that by duplicating the number of processors, we gain a speedup of 35%, which, even if an acceptable speedup, we do not concede it as significant. A more sophisticated parallelization techniques is still a subject for future research.

6 Cobalt: A Content-Based Similarity Approach For Link Discovery over Geospatial Knowledge Graphs

Preamble: This chapter is based on the paper [Bec+23] that proposes COBALT to speed up the discovery of geospatial links.

COBALT combines content measures with R-tree indexing. Content measures are based on the area, diagonal, and distance of the MBB of polygons, which speed up the process but are not perfectly accurate. Thus, we propose two polygon splitting approaches to improve the accuracy of COBALT. Our experiments in real-world datasets show that COBALT is able to accelerate the discovery of topological relationships in geospatial KGs by up to 1.47×10^4 times over state-of-the-art linking algorithms while maintaining an F-measure between 0.7 and 0.9, depending on the topological relation. Furthermore, we achieved an F-measure of up to 0.99 by applying our polygon-splitting approaches before applying the content measures. The process of discovering links between geospatial resources can be significantly faster without sacrificing the optimality of the results. This is especially important for real-time data-driven applications such as emergency response, location-based services, and traffic management. In future work, additional measures, such as the location of polygons or the name of the entity represented by the polygon, could be integrated to further improve the accuracy of the results.

6.1 Motivation

The necessity for highly scalable methods for finding links between geospatial resources has arisen due to a result of the rapid proliferation of Linked Data in geospatial data. Only 7.1% of the relationships between resources relate geographical elements, as was noted in earlier publications [Ngo13]. There are two basic causes for this: 1) The vast quantity of geospatially represented resources on LOD requires scalable techniques for computing linkages between geospatial resources. Examples include LGD [ALH09], which has more than 20 billion triples that describe millions of geographical resources. 2) The computation of certain relations, such as distance and topological links between geospatial resources, is required by the vector representation of geospatial data. For example, identifying the nearby point of interest within a certain radius. The discovery of links among KGs in RDF is crucial for many semantic web applications, according to the Linked Data principles. However, using geospatial resources in time-critical applications [ZL05], such as emergency response, location-based services, and real-time traffic management, needs instant access to geospatial KGs to make quick decisions and take instantaneous actions. Thus, the efficiency of the LD process becomes more challenging. Efficient LD approaches must be developed to achieve scalability and efficiency in such real-time applications. Recently, algorithms such as RADON [She+17], RADON2 [ASN18c] (see Chapter 5), GIA.NT [Pap+21], and DORIC [Jin+21] have been developed. These algorithms compute topological relations between geographical resources quickly and effectively and the DE-9IM [CSE94] is used in all of them. The DE-9IM defines the topological relations between two-dimensional geometries by calculating the dimensions of the intersections between the interior, boundary, and exterior of two geometries. The relations defined by DE-9IM are commonly used in natural language [EMH94]: `Equals`, `Disjoint`, `Intersects`, `Touches`, `Crosses`, `Within`, `Contains`, `Overlaps`, `Covers`, and `Covered By`. Ahmed et al. [ASN18b] (see Chapter 4) have studied the effect of simplifying resource geometries on the runtime and F-measure of LD approaches. However, computing the DE-9IM is very expensive in terms of runtime.

In our efforts to improve the scalability of LD over geospatial KGs, in Chapter 4, we focused on the scalability of LD by employing line simplification algorithms as preprocessing steps. Specifically, these algorithms were applied to IM and point-set distance measures to efficiently detect geospatial relations. This exploration was centered around evaluating their effect on two critical metrics: the F-measure, indicating the efficiency of discovered relations, and runtime, which measures scalability. Then, in Chapter 5, we shifted our focus solely to the role of IM in LD over geospatial KGs. We examine the scalability implications of parallelizing IM computations, particularly in terms of runtime. An extension of the RADON algorithm [She+17] was introduced, which uses a specialized indexing method combined with space tiling to calculate the topological relations between geospatial resources based on the DE-9IM standard.

In this chapter, we further refine our approach to address the remaining challenges associated with the high computational cost involved in both line simplification and IM methods. To achieve this, we introduce COBALT. COBALT is a hybrid methodology that combines content measure similarity and R-tree indexing. This synergistic approach is designed to significantly reduce computational overhead, thus improving the efficiency of LD in geospatial KGs. COBALT uses content measures combined with R-Tree indexing to discover the topological relations defined in [CSE94] and [EMH94]. To our knowledge, this is the first work that uses content measures integrated with R-Tree indexing to discover links among RDF geospatial resources. We summarize our contribution as follows:

1. We present and formalize the problem of topological relation discovery for geospatial resources based on content measures and R-tree indexing.
2. We study the effect of using different R-tree building algorithms, node capacities, and the impact of indexing both KGs.
3. We study the impact of using the content-based measures for topological relations discovery on both runtime and accuracy.
4. To increase the accuracy of our approach, we propose two polygon-splitting strategies and analyze their effect on both runtime and accuracy.

The remainder of this chapter is structured as follows. We begin by introducing our approach in Section 6.2. In Section 6.3, we present our evaluation and results.

6.2 Approach

We start our approach by indexing the source dataset polygons using the R-tree; we then apply content measures on the indexed polygons.

6.2.1 R-tree Indexing

R-trees [BS12] are an improved variant of binary trees, where an R-tree stores the MBBs of the polygons instead of the polygons themselves. In COBALT, we use *Guttman*'s R-tree [Gut84] to index the source dataset in order to filter out as many disconnected polygon pairs as possible to reduce the runtime of the linking process. Every node's MBB contains all of its children's MBBs, so in case an MBB of a parent node does not intersect a query rectangle (a query rectangle is an MBB from target data), none of its descendants can [Gut84]. The bottom layer of an R-tree stores the MBBs of polygons in the source dataset, and all layers above it match the criterion applied to indexing the bottom layer. An example of a handcrafted R-tree is shown in Figure 6.1.

6.2.2 Querying R-trees

R-trees are easy to query recursively. Let q be the target MBB of the polygon for which we want to find the intersected MBBs of the source polygons. Since the nodes of an R-tree are R-trees, we use the same algorithm for each node. If the current node of the R-tree is in the bottom layer (i.e., it contains no other R-trees but polygons), we then verify if q intersects each of the MBBs of the source polygons saved in this node and add such source polygons to the query result. In case the current node of the R-tree is not on the bottom layer, we check if each child node's MBB has at least one common point with q, and if that is the case, we recursively repeat the method for that node. In Figure 6.1, for instance, the MBBs of the two blue nodes on the left overlap. If the query rectangle q lies in the area where two nodes' MBBs overlap, we have to check the children of both nodes. Therefore, we need a fast-building approach that minimizes overlapping parent nodes.

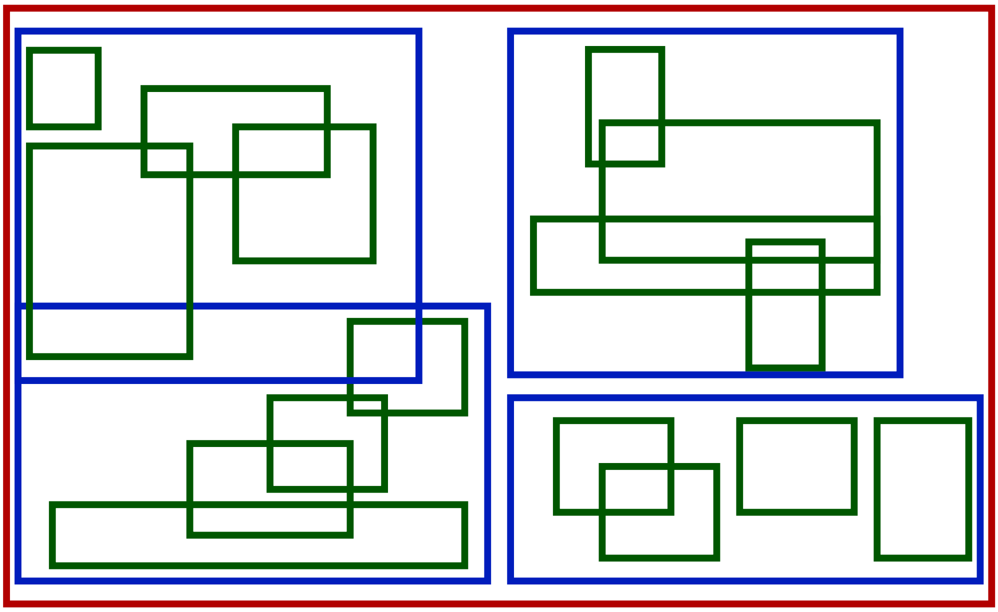

Fig. 6.1.: A handcrafted R-tree, where green is the bottom layer, blue is the middle, and red is the top layer.

6.2.3 Building R-trees

There are two main ways for constructing R-trees: (i) *Static building algorithms* work by getting all data as the input and then constructing the tree with all data at once. (ii) *Dynamic building algorithms* work by inserting data one by one into the tree. As our KGs do not change frequently, we focus on static algorithms, as dynamic algorithms will require more runtime for reinserting data to keep the R-tree balanced. Because we only query the R-tree once for every target geometry, the build quality (i.e., overlap) is less important for an overall fast execution. To test the impact of different building algorithms, we use four static R-tree building algorithms (i.e., *SmallestX*, *STR*, *OTM*, and *PackedHilbertR-tree*) and one dynamic algorithm (i.e., R*-tree). The first static algorithm is *SmallestX* [KF93], which sorts the MBBs by the smallest x coordinate. The *SortTileRecursive* [LLE97] algorithm (STR) builds the R-tree in a bottom-up fashion: it divides the MBBs into slices, initially sorting them by the x-coordinate and subsequently by the y-coordinate. It then recursively combines the parent nodes of the bottom layer. The *OTM* algorithm [LL03] works similarly to STR but recursively sorts MBBs by alternating x- and y-coordinates with a top-down bulk-loading approach. The last static building algorithm we use is the *PackedHilbertR-tree* [KF94] algorithm, which sorts the MBBs by their position on the Hilbert curve. In contrast, we use the dynamic R-tree building algorithm *R*-tree* [Bec+90], which supports the insertion of new elements after creation and tries to minimize the area occupied by nodes. In our experiments, we insert the polygons one by one into the R*-tree and use the values $\{4, 8, 16, 32, 64, 128, 256\}$ for the capacities of each node.

6.2.4 Content Measures for Topological Relations

Given two MBBs, A and B, the *area-based content measure* (F_a) is the first of the three content measures from [GR04]. F_a is the normalization of the area of each MBB of both A and B by the area of the MBB of the union of A and B. Formally,

$$F_a(A, B) = \frac{\text{area(A)}}{\text{area(MBB(A} \cup \text{B))}}, \tag{6.1}$$

where $F_a(B, A)$ is defined analogously. The range of $F_a \in (0, 1]$. In Table 6.1, we present the values of $F_a(A, B)$, $F_a(B, A)$, and $F_a(A, B) + F_a(B, A)$ for the different topological relations. F_a cannot distinguish the following pairs of relations: (`Meet`, `disjoint`), (`covers`, `contains`), and (`covered by`, `inside`). For instance, the union MBB will be the same as the MBB of the MBBs for `contains` and `covers`.

Tab. 6.1.: Area-based content measure relations based on values of F_a [GR04].

Relation	Disjoint	Meets	Overlap	Equals	Covers	CoveredBy	Contains	Inside
$F_a(A, B)$	(0,1)	(0,1)	(0,1)	1	1	(0,1)	1	(0,1)
$F_a(B, A)$	(0,1)	(0,1)	(0,1)	1	(0,1)	(0,1)	(0,1)	1
$F_a(A, B)+F_a(B, A)$	(0,1)	(0,1]	(0,2)	2	(1,2)	(1,2)	(1,2)	(1,2)

However, this measure can accurately detect the `equalrelation` because both input MBBs have the same area as the MBB of their union. The second content measure is the *diagonal-based content measure* (F_d) [GR04], formally defined as:

$$F_d(A, B) = \frac{\text{diagonal(A)}}{\text{diagonal(MBB(A} \cup \text{B))}}. \tag{6.2}$$

The range of $F_d \in (0, 1]$, and it cannot distinguish `(covers, contains)` and `(covered by, inside)` for the same reason as in the case of F_a. The third measure of content is the *mixed content measure* (F_m) [GR04]. F_m utilizes the area, diagonal, and distance of the MBBs to find the topological relations. Unlike the other two content measures, it is capable of distinguishing between `(contains, covers)` and `(inside, covered by)`. Formally,

$$F_m(A, B) = \frac{\text{area(A)} - 2 \cdot \text{area(MBB(A} \cap \text{B))}}{\text{area(A)}} + \frac{\text{distance(A, B)}}{\text{diagonal(A)}}. \tag{6.3}$$

6.2.5 Combining R-tree Indexing and Content Measures

Our R-tree indexing filters out `disjoint` polygon pairs based on their MBB. We only keep the indexed source dataset in memory, which reduces the space complexity as we stream-process the target dataset. In the case of the `disjoint` relation, we first add all pairs of geometries that the indexing would filter out to the result set, and then we check the other relations on the rest of the geometries pairs. In Algorithm 13, we outline the steps for the area-based content measure F_a. For the other measures, we replace F_a with F_d for the diagonal measure and F_m for the mixed measure (Lines 8–9). Additionally, the values of F_a need to be checked against the other measures' values from [GR04] (Line 11).

6.2.6 Indexing Both Datasets

In many cases, swapping the source and target datasets results in different runtimes. To reduce the impact of data set ordering on runtime, we study the possibility of

Algorithm 3: DiscoverLinksAreaBased(G_s, G_t, r)

```
input  : Source KG $G_s$, Target KG $G_t$, Topological relation $r$
output : Mapping: $M = \{(s, r, t) | s \in G_s, t \in G_t\}$
$tree \leftarrow buildRtree(G_s)$;
Initialise $M \leftarrow \{\}$ ;
for MBB(t) $t \in G_t$ do
    $I \leftarrow$ queryRtree(tree, $t$));
    if $r$ is disjoint then
        Add all pairs$(s, r, t) \forall s \in (G_s \setminus I)$ to $M$
    for MBB(s) $s \in I$ do
        $X \leftarrow \mathrm{F}_a(MBB(s), MBB(t))$;
        $Y \leftarrow \mathrm{F}_a(MBB(t), MBB(s))$;
        $Z \leftarrow X + Y$ ;
        if $X, Y, Z$ match the respective values of the relation $r$ in Table 6.1 then
            Add $(s, r, t)$ to $M$;
return $M$
```

Algorithm 4: MatchTrees(sourceTree, targetTree)

```
Result $\leftarrow \{\}$;
if area(sourceTree) < area(targetTree) then
    swap sourceTree and targetTree;
foreach child of sourceTree do
    if child is leaf then
        Result = Result $\cup$ queryRtree(targetTree, MBB(child));
    else
        Result = Result $\cup$ MatchTrees(child,targetTree);
return Result;
```

indexing both data sets instead of one. Instead of querying the R-tree for each target geometry, we use Algorithm 4 to match two R-trees to each other and recursively find all pairs that intersect. This approach removes the need to choose which dataset to index, but it comes with the price of increasing the memory footprint of our approach as we have to keep both datasets in memory.

6.2.7 Splitting Polygons to Gain Accuracy

We can improve the F-measure of COBALT for some relations by splitting the geometries into multiple pieces before using the content measurement functions to determine the relation. In particular, we split polygons recursively t into four pieces using two strategies: **1) Equal split** and **2) Fitting split**. With equal split, we split the original polygon into equal-sized parts. The resulting polygon parts are the intersection of a grid pattern over the polygon and the original polygon itself. In some cases, this leads to some splits not achieving any additional information, as their parts of the grid are empty. For example, in Figure 6.2, any further splits of the top left corner cell (the blue highlighted cell) would not increase the accuracy of COBALT as the original polygon does not have any points within this cell.

With fitting split, we divide the polygon into parts of equal size but use the MBB of the current polygon part for further splitting. Splitting the cell in the upper left corner of the same polygon of the previous example using this strategy will result in the splitting presented in Figure 6.3, where further splitting of the blue-highlighted cell results in more detailed splits that fit better to the shape of the polygon. After splitting the polygons, we compute the MBBs for all parts. Now, since we have multiple polygon parts, we change the way the relation of the original polygon is determined. Let t be the number of divisions into four parts. Let $A_{(i,j)}$ be the split part of geometry A in column i and row j and $B_{(k,l)}$ be the split part of geometry B in column k and row l for $\{(i,j,k,l) \in \mathbb{N}^4 | 0 \leq i,j,k,l < 2^t\}$. The newly defined relations can be found in Table 6.2. In particular, every grid pattern $A_{(i,j)}$ must be equal to $B_{(k,l)}$ for the `equals` relation to hold. For the `intersects` relation, at least one $A_{(i,j)}$ has to intersect with at least one $B_{(k,l)}$. For the `within` relation, all $A_{(i,j)}$ have to be contained in the MBB of the union of all $B_{(k,l)}$ it intersects.

For the `contains` relation, we swap A and B and then compute the `within` relation instead. For the `overlaps` relation, three conditions must hold: 1) at least one $A_{(i,j)}$ is not within B, 2) at least one $B_{(k,l)}$ is not within A, and 3) at least one $A_{(i,j)}$ intersects with at least one $B_{(k,l)}$. For the `touches` relation, at least one $A_{(i,j)}$ must touch any $B_{(k,l)}$ and every $B_{(i,j)}$ is related to every $B_{(k,l)}$ by the `touches` or `disjoint` relation.

6.3 Evaluation and Results

6.3.1 Datasets

We use two real-world KGs for evaluating COBALT: 1) The NUTS[1] dataset from the *Eurostat* group, which describes the territory of countries in the European Union,

[1] `https://ec.europa.eu/eurostat/de/web/gisco/geodata/reference-data/administrative-units-statistical-units/NUTS`, accessed on 01.09.2022

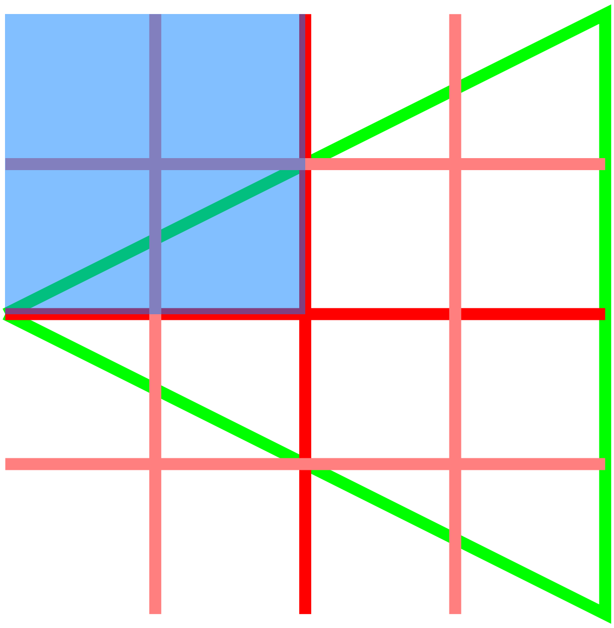

Fig. 6.2.: Splitting the polygon into equal-size parts two times. The green triangle is the original polygon, the dark red lines are the splitting lines for the first split iteration, and the light red lines are the splitting lines for the second split iteration. The blue rectangles indicate the MBB used to determine the second iteration splitting lines of the top left corner.

(potential) candidate countries, and countries belonging to the European Free Trade Association, and 2) the CLC[2] [BK17] created by the EEA. The CLC contains information on land use by the 39 EEA39 countries[3].

6.3.2 Hardware and Software

All experiments were carried out in the NOCTUA1[4] cluster of *Paderborn University*. NOCTUA1 consists of 256 compute nodes, each having two Intel *Xeon Gold* "Skylake" 6148 processors, with a total of 40 cores with 2.4 GHz and 192 GiB main memory. All algorithms were implemented in *Java*, and the compute nodes were run on *OpenJDK* version 11.0.2. For an accurate measurement of runtime, all experiments were started with all KGs already loaded into the main memory. In addition, to link each pair of data sets, we run the algorithms on the same compute node. All experiments were carried out with a memory limit of 30 GB. Unless otherwise stated, we use only one core for all experiments.

[2] `https://land.copernicus.eu/pan-european/corine-land-cover`, accessed on 01.09.2022

[3] `https://land.copernicus.eu/portal_vocabularies/geotags/eea39`

[4] `https://pc2.uni-paderborn.de/hpc-services/available-systems/noctua1`

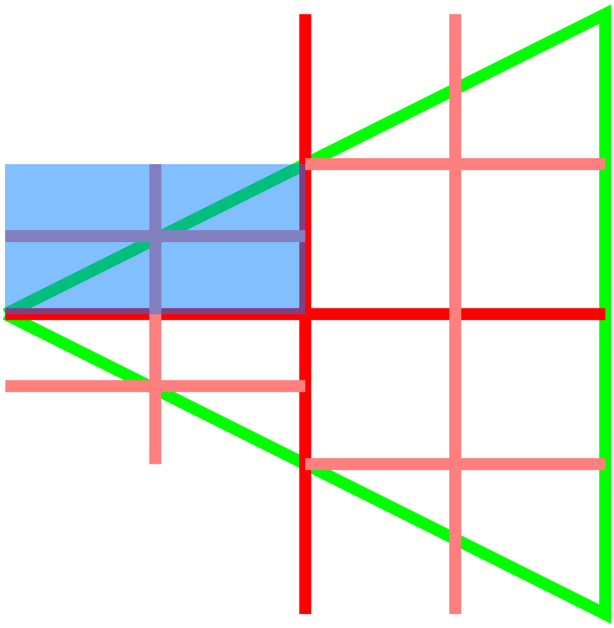

Fig. 6.3.: Splitting the polygon into fitting parts two times. The green triangle is the original polygon, the dark red lines are the splitting lines for the first split iteration, and the light red lines are the splitting lines for the second split iteration. The blue rectangles indicate the MBB used for determining the second iteration splitting lines of the top left corner.

6.3.3 Experiments Settings

We use COBALT$_{area}$, COBALT$_{diagonal}$, and COBALT$_{mixed}$ to dub the *area*, *diagonal*, and the *mixed* measures of COBALT, respectively. We use the following four baselines:

1. RADON [She+17],
2. RADON with only the MBBs of the original polygons (dubbed RADON$_{\text{MBB}}$),
3. GIA.NT [Pap+21], and
4. GIA.NT with only the MBBs of the original polygons (dubbed GIA.NT$_{\text{MBB}}$). For a fair runtime comparison, we use a version of GIA.NT that computes only one relation at a time.

We also implemented a version based on space indexing of COBALT, where we optimized content-based measures based on space indexing of RADON [She+17]. We use COBALT$_{area(R)}$, COBALT$_{diagonal(R)}$, and COBALT$_{mixed(R)}$ to dub the *area*, *diagonal*, and *mixed* measures of the space-tiled indexing measures of COBALT, respectively. We also used the *Douglas-Peucker* polygon simplification algorithm [DP73]. Simplification is applied to the data set using simplification thresholds {0.05, 0.1, 0.2};

Tab. 6.2.: Topological relations based on multiple splits of polygons.

`equals`	$\forall \{(i,j) \in \mathbb{N}^2 \mid 0 \leq i,j < 2^t\}$: $A_{(i,j)}$ `equals` $B_{(i,j)}$
`intersects`	$\exists \{(i,j,k,l) \in \mathbb{N}^4 \mid 0 \leq i,j,k,l < 2^t\}$: $A_{(i,j)}$ `intersects` $B_{(k,l)}$
`disjoint`	$\forall \{(i,j,k,l) \in \mathbb{N}^4 \mid 0 \leq i,j,k,l < 2^t\}$: $A_{(i,j)}$ `disjoint` $B_{(k,l)}$
`within`	$\forall \{(i,j) \in \mathbb{N}^2 \mid 0 \leq i,j < 2^t\}$: $A_{(i,j)}$ `within` MBB$(\{B_{(k,l)} \mid \forall\{(k,l) \in \mathbb{N}^2 \mid 0 \leq k,l < 2^t\} :$ $A_{(i,j)}$ `intersects` $B_{(k,l)}\})$
`contains`	swap A and B then compute `within`
`overlaps`	$(\exists \{(i,j,k,l) \in \mathbb{N}^4 \mid 0 \leq i,j,k,l < 2^t\}$: $A_{(i,j)}$ `equals` $B_{(k,l)}) \vee A_{(i,j)}$ `within` $B_{(k,l)}) \vee$ $A_{(i,j)}$ `contains` $B_{(k,l)}) \vee A_{(i,j)}$ `overlaps` $B_{(k,l)})\wedge$ $(\exists \{(i,j) \in \mathbb{N}^2 \mid 0 \leq i,j < 2^t\}$: $A_{(i,j)} \neg$ `within` MBB $(\{B_{(k,l)} \mid \forall\{(k,l) \in \mathbb{N}^2 \mid 0 \leq k,l < 2^t\} :$ $A_{(i,j)}$ `intersects` $B_{(k,l)}\}))\wedge$ $(\exists \{(i,j) \in \mathbb{N}^2 \mid 0 \leq i,j < 2^t\}$: $B_{(i,j)}$ $\neg$`within` MBB$(\{A_{(k,l)} \mid \forall\{(k,l) \in \mathbb{N}^2 \mid 0 \leq k,l < 2^t\} :$ $B_{(i,j)}$ `intersects` $A_{(k,l)}\}))$
`touches`	$(\exists \{(i,j,k,l) \in \mathbb{N}^4 \mid 0 \leq i,j,k,l < 2^t\}$: $A_{(i,j)}$ `touches` $B_{(k,l)})\wedge$ $\neg(\exists \{(i,j,k,l) \in \mathbb{N}^4 \mid 0 \leq i,j,k,l < 2^t\}$: $A_{(i,j)}$ `equals` $B_{(k,l)}) \vee A_{(i,j)}$ `within` $B_{(k,l)}) \vee$ $A_{(i,j)}$ `contains` $B_{(k,l)}) \vee A_{(i,j)}$ `overlaps` $B_{(k,l)})$

subsequently, the relations are calculated using RADON. They were labeled as RADON$_{\text{simp}(0.05)}$, RADON$_{\text{simp}(0.1)}$, and RADON$_{\text{simp}(0.2)}$. Within all experiments, we computed the topological relations {`equals`, `intersects`, `contains`, `within`, `touches`, and `overlaps`}.

6.3.4 Research Questions

We aim to answer the following research questions:

Q_1. What is the effect of indexing the input datasets on the runtime of COBALT?

Q_2. How much efficiency (i.e., less runtime) can we gain by using content measures for topological relation discovery?

Q_3. How much accuracy do we lose (i.e., less F-measure) by using content measures for topological relation discovery?

Q_4. In case we use a simplified version of the original polygons, will we have a better trade-off between accuracy and efficiency than using COBALT?

Q_5. Will COBALT benefit from parallelization for big KGs such as CLC?

Q_6. What is the trade-off between accuracy and efficiency when we integrate our polygon splitting strategies into COBALT?

Research question Q_1. The aim of our *first set of experiments* was to evaluate different R-tree indexing options for COBALT. To measure the difference in runtime between indexing only one dataset vs. indexing both the source and target datasets, we linked NUTS to CLC (see Table 6.3) and CLC to NUTS (see Table 6.4). First, we compared the algorithms that index both datasets to the algorithms that only index one dataset. Our results showed that when linking NUTS to CLC, most of the algorithms that only index one dataset are faster than the matching algorithms that index both of them.

However, in the CLC to NUTS experiment (CLC×NUTS), the matching algorithms that index both datasets are faster than the algorithms that only index one dataset. This shows that the choice of the source dataset makes a difference regarding the runtime, and the smaller dataset should be indexed instead of the bigger one. Because of the greater memory needed for indexing both datasets, we decided to index only one dataset in our further experiments. In addition, our results showed that computing the *Hilbert curve*, or inserting entries one by one with the *R*-tree*, takes much more time than the other algorithms. OMT and STR have the best runtime of the algorithms as their computations are not expensive and produce high-quality R-trees. We conclude that the choice of the R-tree building algorithm and the capacity of the R-tree are highly dependent on the datasets used for benchmarking. It is important to find a balance between a fast-building algorithm and an algorithm that allows efficient queries. Sorting entries by both x- and y-coordinates, such as OMT and STR, is a good way to achieve this. This answers our first research question, Q_1.

For the following experiments, we use the STR building algorithm with a capacity of 4, but to respect the downside of only indexing one dataset, we also use the longer-taking dataset combination for runtime values.

To answer Q_2, Q_3, and Q_4, we conducted our *second set of experiments* where we evaluated the performance of COBALT vs. all baselines in terms of runtime and F-measure. In particular, we aimed to find the topological relations within the NUTS dataset against itself (i.e., NUTS×NUTS) and CLC×NUTS using each of the

Tab. 6.3.: Runtime in milliseconds for linking NUTS to CLC using different R-tree building algorithms and capacities combined with the mixed content measure.

Algorithm	4	8	16	32	64	128	256
R*-tree	55745	49757	61810	77957	137733	202192	208227
Hilbert	258193	245062	233901	236357	363893	270579	273802
SmallestX	204226	95004	58875	64362	65360	62508	65948
OMT	35055	35778	36415	44063	71217	82937	103397
STR	36986	37088	38366	44042	48191	61899	77262
MatchHilbert	398502	345562	328975	320710	327772	357677	410224
MatchSmallestX	85300	71626	62810	59780	46495	53344	66957
MatchOMT	72359	56792	55213	50404	44134	48084	54525
MatchSTR	45716	42130	41216	41769	41859	42278	43248

Tab. 6.4.: Runtime in milliseconds for linking CLC to NUTS using different R-tree building algorithms and capacities combined with the mixed content measure.

Algorithm	4	8	16	32	64	128	256
R*-tree	135037	139044	189439	372823	1067386	3527572	1067386
Hilbert	801618	866145	621211	793548	556130	540250	556130
SmallestX	263744	245692	422600	195218	93560	134091	324985
OMT	77928	62193	60354	55639	48598	54950	66138
STR	51448	47783	46708	47289	47359	48344	47359
MatchHilbert	401815	357914	363288	358353	344742	370740	417821
MatchSmallestX	138362	106530	73954	70796	57164	68377	107256
MatchOMT	72590	57643	56258	51175	45958	50478	56514
MatchSTR	45245	42892	42077	42613	43233	43218	43849

algorithms mentioned above. For linking NUTS× NUTS, the total runtimes required to compute the topological relations are shown in Figure 6.4. The F-measure for each relation can be seen in Table 6.5. For linking CLC×NUTS, the total required runtimes to compute the same six relations are shown in Figure 6.5. The F-measure of each relation can be seen in Table 6.6.

Research question Q_2. From Figure 6.4, we can see that all the content-based measures implemented in Cobalt (i.e., Cobalt$_{area}$, Cobalt$_{diagonal}$, and Cobalt$_{mixed}$) with R-tree indexing are 4 to 8 times faster than their counterparts (i.e., Cobalt$_{area(R)}$, Cobalt$_{diagonal(R)}$, and Cobalt$_{mixed(R)}$) deployed based on the Radon's space tiling indexing. For example, the total runtime of Cobalt$_{mixed}$ is 195 milliseconds while the total runtime of Cobalt$_{mixed(R)}$ is 849 milliseconds, which means that Cobalt$_{mixed}$ is 4.3 times faster than Cobalt$_{mixed(R)}$. The slowest content-based measure of Cobalt (that is, Cobalt$_{area}$) is, *on average*, 4840 times faster than Radon.

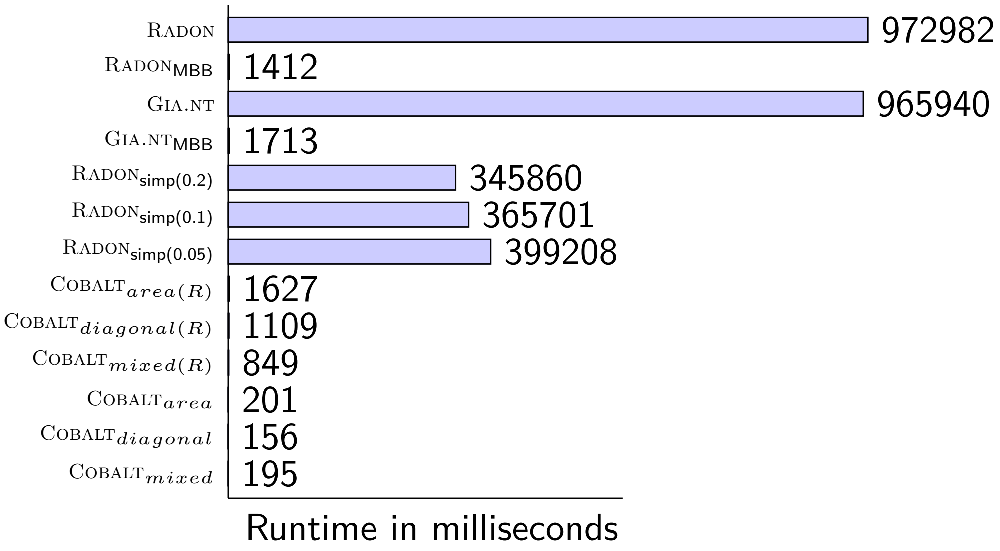

Fig. 6.4.: Total runtime of linking NUTS×NUTS.

COBALT$_{mixed}$ is up to 1.47×10^4 times faster than RADON, which is the best speedup COBALT has in comparison to all other algorithms. This clearly shows how efficient the content-based measures are when it comes to run time, which answers our second research question, Q_2.

Research question Q_3. Based on the results of Table 6.5, we analyzed the impact of using the content-based measures on the F-measure. For discovering the `equals` relation based on the MBBs of the original polygons, all algorithms achieved an F-measure of 0.996. For the `intersects`, `contains`, and `within` relations, the F-measures were 0.852, 0.853, and 0.853, respectively. The `overlaps` relation was the most impacted relation using the MBBs. For example, in the NUTS× NUTS experiment, using MBB instead of the original polygons as input to discover the `overlaps` relation resulted in 586 true positives (out of 790 or 74.47%), 26884 false positives, 4012426 true negatives (out of 4039310 or 99.33%) and 204 false negatives. In total, using MBBs falsely classified 45 more polygon pairs as overlapped than the true number of overlapping pairs. The high number of pairs that was correctly identified as not overlapping is caused by the indexing algorithm, which filters out a high percentage of nonintersecting pairs.

The only relation in which content-based measures produce better F-measures than both RADON$_{\text{MBB}}$ and GIA.NT$_{\text{MBB}}$ was the `touches` relation. Both RADON$_{\text{MBB}}$ and GIA.NT$_{\text{MBB}}$ were unable to detect the `touches` relation correctly in most cases since the intersection matrix of the MBBs is highly dependent on the shape of the polygon.

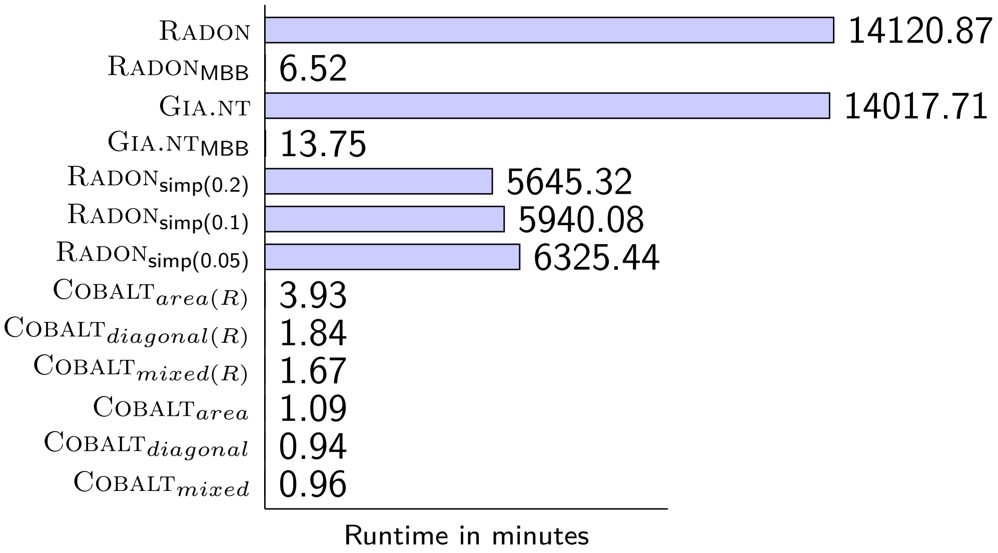

Fig. 6.5.: Total runtime of linking CLC×NUTS.

For example, using both RADON$_{\mathrm{MBB}}$ and GIA.NT$_{\mathrm{MBB}}$ to discover the `touches` relation for the NUTS× NUTS experiment (again, see Table 6.5) resulted in an F-measure of 0.001, while COBALT$_{area}$ and COBALT$_{diagonal}$ achieved F-measures of 0.678 and 0.779, respectively.

Both the area and diagonal measures benefitted from the fact that there are 20150 pairs that `touch` each other but only 790 pairs that `overlap`. To summarize, using the content-based measures, we lose, *on average*, 32% of the F-measure compared to the F-measure of 1.0 produced by RADON or GIA.NT. This answers our third research question, Q_3.

Research question Q_4. State-of-the-art approaches tend to use polygon simplification to speed up the LD of topological relations [ASN18b]. As part of our second set of experiments, we studied the trade-off between accuracy and efficiency by using content-based measures on the MBBs of the polygons versus using a simplified version of the original polygons. Based on the results in Table 6.5, the F-measures of RADON$_{\mathrm{simp(0.05)}}$, RADON$_{\mathrm{simp(0.1)}}$ and RADON$_{\mathrm{simp(0.2)}}$ for the relations `contains` and `within` were worse than all results produced using polygon MBBs.

For example, RADON$_{\mathrm{simp(0.05)}}$, RADON$_{\mathrm{simp(0.1)}}$, and RADON$_{\mathrm{simp(0.2)}}$ achieved the F-measures 0.7, 0.72, and 0.733, respectively. While using COBALT on the MBBs of the original polygons achieved an F-measure of 0.853 for the `contains` and `within` relations. When using polygon simplification algorithms, the F-measure for the `equals` relation is 1.0 for RADON with simplified polygons when linking

Tab. 6.5.: F-measure for linking NUTS×NUTS (all values rounded to three decimal places). The results of COBALT combined with RADON indexing are omitted because the indexing does not change the accuracy.

Algorithm	equals	intersects	contains	within	touches	overlaps
RADON	1.000	1.000	1.000	1.000	1.000	1.000
$\text{RADON}_{\text{MBB}}$	0.996	0.852	0.853	0.853	0.001	0.041
GIA.NT	1.000	1.000	1.000	1.000	1.000	1.000
$\text{GIA.NT}_{\text{MBB}}$	0.996	0.852	0.853	0.853	0.001	0.041
$\text{RADON}_{\text{simp(0.2)}}$	1.000	0.916	0.733	0.733	0.177	0.068
$\text{RADON}_{\text{simp(0.1)}}$	1.000	0.953	0.721	0.721	0.199	0.064
$\text{RADON}_{\text{simp(0.05)}}$	1.000	0.980	0.700	0.700	0.209	0.061
COBALT_{area}	0.996	0.852	0.853	0.853	0.678	0.041
$\text{COBALT}_{diagonal}$	0.996	0.852	0.853	0.853	0.779	0.041
COBALT_{mixed}	0.996	0.852	0.853	0.853	0.001	0.041

Tab. 6.6.: F-measure for linking CLC×NUTS. All values are rounded to three decimal places and - indicates the total absence of the relation in the result set. The results of COBALT combined with RADON indexing are omitted because the indexing does not change the accuracy.

Algorithm	equals	intersects	contains	within	touches	overlaps
RADON	-	1.000	1.000	-	-	1.000
$\text{RADON}_{\text{MBB}}$	-	0.709	0.689	-	-	0.066
GIA.NT	-	1.000	1.000	-	-	1.000
$\text{GIA.NT}_{\text{MBB}}$	-	0.709	0.689	-	-	0.066
$\text{RADON}_{\text{simp(0.2)}}$	-	0.938	0.931	-	-	0.332
$\text{RADON}_{\text{simp(0.1)}}$	-	0.963	0.958	-	-	0.419
$\text{RADON}_{\text{simp(0.05)}}$	-	0.980	0.975	-	-	0.540
COBALT_{area}	-	0.709	0.689	-	-	0.066
$\text{COBALT}_{diagonal}$	-	0.709	0.689	-	-	0.066
COBALT_{mixed}	-	0.709	0.689	-	-	0.066

NUTS×NUTS. The content measures are able to achieve an F-measure of 0.996 for this relation.

From the aforementioned results, we can conclude that using content-based measures on the polygons' MBBs results in a better trade-off between efficiency and accuracy than using a simplified version of polygons. We can see the same behavior for our second linking task, i.e., CLC×NUTS (see results in Table 6.6). This clearly answers our fourth research question, Q_4.

Research question Q_5 To answer Q_5, we conducted our *third set of experiments* by linking CLC against itself, i.e., CLC×CLC. For this experiment, we implemented a

Tab. 6.7.: Runtime for linking CLC×CLC using different numbers of threads. All Runtimes are recorded in hours, with all values are rounded to three decimal places.

Algorithm	1 Thread	2 Threads	4 Threads	8 Threads
RADON	1179.489	686.007	340.827	186.913
RADON$_{\text{MBB}}$	0.703	0.729	0.597	0.586
GIA.NT	1179.463	675.589	334.928	179.415
GIA.NT$_{\text{MBB}}$	0.635	0.458	0.346	0.329
COBALT$_{area(R)}$	0.710	0.670	0.583	0.573
COBALT$_{diagonal(R)}$	0.539	0.564	0.513	0.505
COBALT$_{mixed(R)}$	0.494	0.550	0.509	0.503
COBALT$_{area}$	0.209	0.215	0.186	0.192
COBALT$_{diagonal}$	0.190	0.195	0.175	0.183
COBALT$_{mixed}$	**0.179**	**0.190**	**0.171**	**0.180**

parallelized version of COBALT, where we used $\{1, 3, 4, 8\}$ thread(s). As shown in Table 6.7, all MBB-based algorithms did not benefit from using multiple threads because the MBB-based algorithms are so fast, to the extent that the time needed for thread coordination is the same as the time saved by allocating the work to other threads. All MBB-based algorithms were able to finish linking CLC to itself in less than one hour. This answers our research question, Q_5.

On the other hand, multiple threads decreased the runtime of RADON and GIA.NT because they use the intersection matrix, which requires expensive computing and can take advantage of employing more threads. In particular, RADON is 6.31 times faster with eight threads than with only one thread. All content measures with R-tree indexing are at least three times faster than RADON with MBBs.

Research question Q_6. To study the trade-off between accuracy and efficiency when we apply our polygon splitting strategies (i.e., the equal split and the fitting split strategies) before applying the content measures, we conducted our *last set of experiments*. In particular, we are interested in comparing COBALT with the two splitting strategies with other approximation algorithms (i.e., polygon simplification). For this experiment, we computed the topological relations for NUTS×NUTS. We benchmarked both split strategies defined in Section 6.2.7 against RADON and the combination of RADON and polygon simplification as we did in the previous experiments. The splitting algorithms are combined with the diagonal-based content measure. We dubbed our first splitting strategy (depicted in Figure 6.2) as EQUAL-t-FD and the second splitting strategy as FITTING-t-FD (depicted in Figure 6.3), with t being the number of recursive splits (we used 0 to 4 recursive splits) and FD

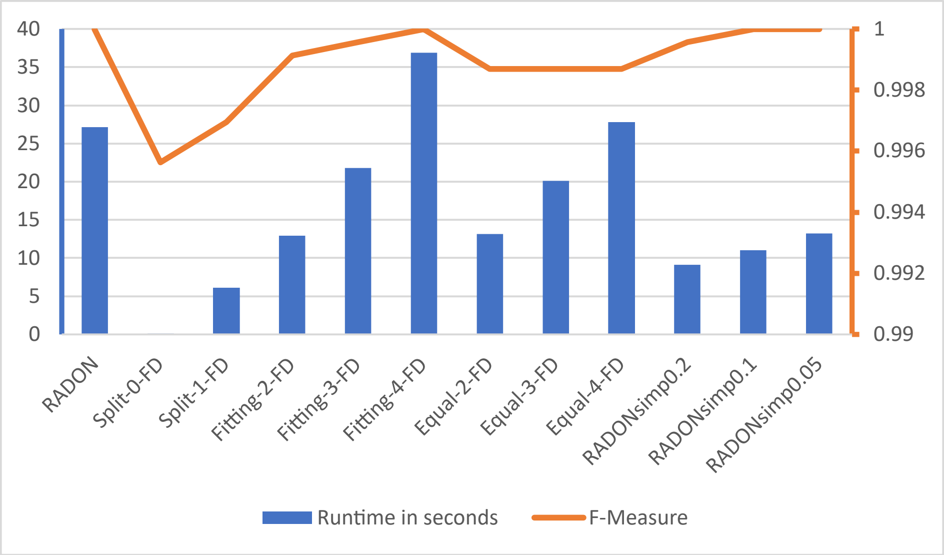

Fig. 6.6.: The `equals` relation. Runtime in seconds (blue) and F-measure (orange) results for linking NUTS×NUTS.

being the diagonal-based content measure. As both split strategies produce the same result for 0 and 1 recursive splits, we only use each of them once as SPLIT-0-FD and SPLIT-1-FD. For the `equals` relation (see Figure 6.6), the diagonal content measure function achieved an F-measure of 0.996 before applying the splitting algorithm on the polygons. The fitting split strategy with 3 and 4 recursive splits (i.e., FITTING-3-FD and FITTING-4-FD) achieved the most accurate results. On the other hand, all the polygon simplification algorithms were able to achieve perfect results in less time than both FITTING-3-FD and FITTING-4-FD. However, the diagonal content measure achieved a high F-measure of 0.996 while being more than 100 times faster than the simplification algorithms. For the `intersects` relation (see Figure 6.7), SPLIT-0-FD achieved an F-measure of 0.852 without splitting polygons. The *fitting-split strategy* has better accuracy for the `intersects` relation than the *equal-split strategy* for each t but with increased runtime. When we compared FITTING-3-FD to EQUAL-4-FD, we also noticed that FITTING-3-FD is faster and more accurate than EQUAL-4-FD. FITTING-3-FD is three times faster than the simplification algorithms and is only slightly worse in accuracy than the RADON$_{\text{simp}(0.05)}$ algorithm but better than RADON$_{\text{simp}(0.1)}$ and RADON$_{\text{simp}(0.2)}$. For the `contains` and `within` relations (see Figures 6.8 and 6.9), the non-split content measures were able to achieve an F-measure of 0.853, which was already better than the simplification algorithms. By splitting the polygons, we were able to achieve a higher F-measure. In particular, FITTING-3-FD was 99% accurate and used only 26.4% of the runtime RADON to compute the relation `that contains the`. This indicated that FITTING-3-FD was the strategy with the best runtime–accuracy trade-off. The `overlaps` relation (see

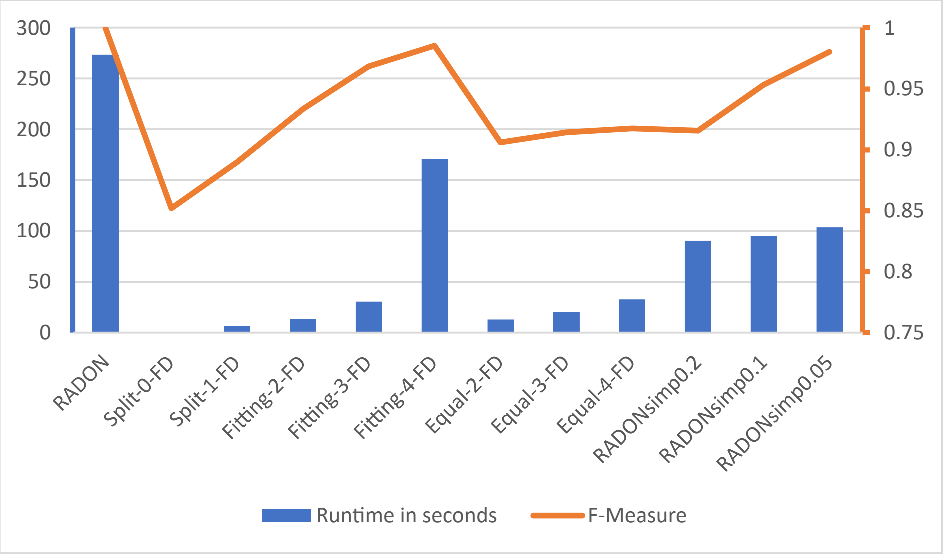

Fig. 6.7.: The `intersects` relation. Runtime in seconds (blue) and F-measure (orange) results for linking NUTS×NUTS.

Figure 6.11) is a relation that cannot be accurately detected by the content measures or the polygon simplification algorithms. In this case, splitting the polygons has a positive effect on the accuracy, but it is still too low to be usable, with all F-measures being smaller than 0.07. For the `touches` relation (see Figure 6.10), the diagonal content measure was able to achieve a higher F-measure (0.779) without splitting. This happens because the diagonal content measure focuses on recall rather than precision, and by splitting the polygons, there are more cases where parts from the two polygons overlap. Therefore, splitting reduces accuracy, and the normal diagonal content measure function should be used for the `touch` relation. In general, splitting polygons is a good way to improve the precision of COBALT for the spatial relations `intersects`, `contains`, and `within`. Our experiments show that FITTING-T-FD achieves a higher F-measure than EQUAL-T-FD algorithms for each respective T but also have a longer runtime. By using our splitting technique, we could guarantee finishing a linking task in a predetermined amount of time while also fully using the time to maximize the accuracy of the result. This answers our research question, Q_6.

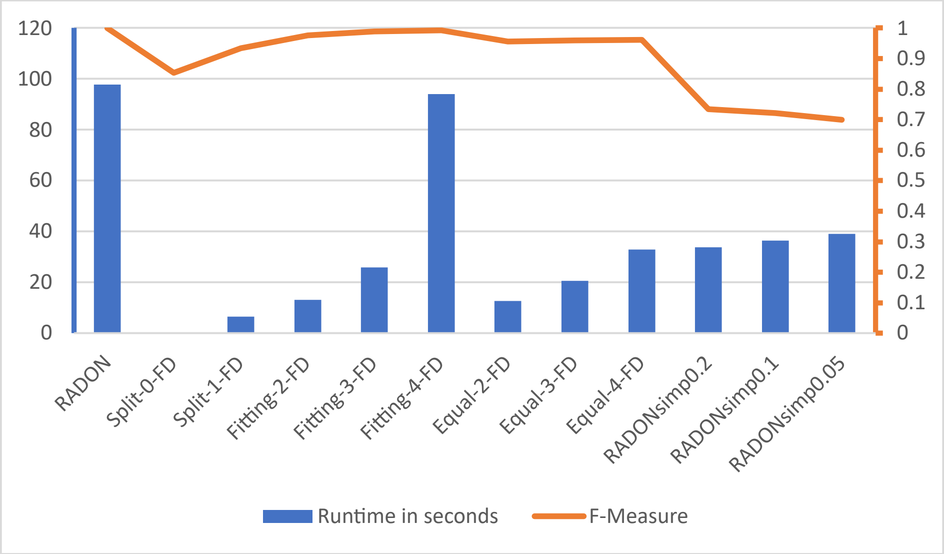

Fig. 6.8.: The `contains` relation. Runtime in seconds (blue) and F-measure (orange) results for linking NUTS×NUTS.

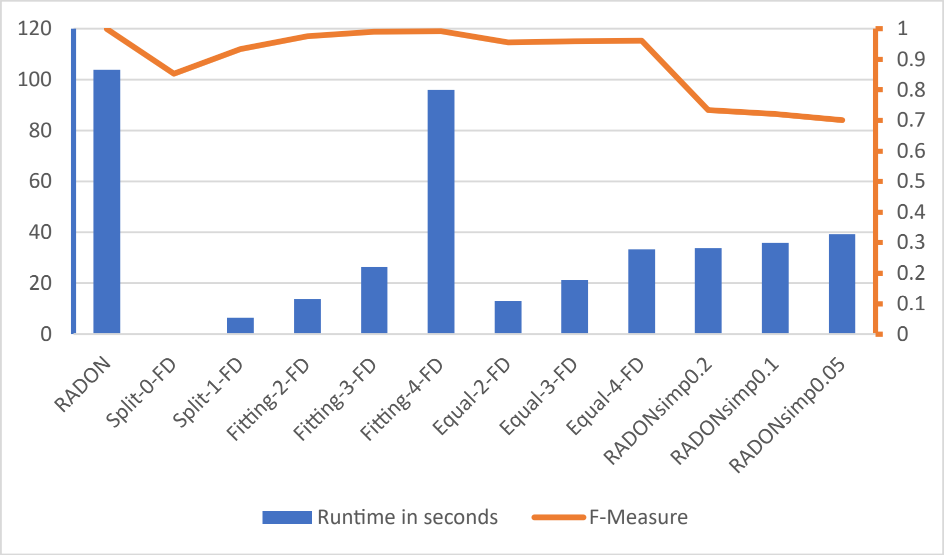

Fig. 6.9.: The `within` relation. Runtime in seconds (blue) and F-measure (orange) results for linking NUTS×NUTS.

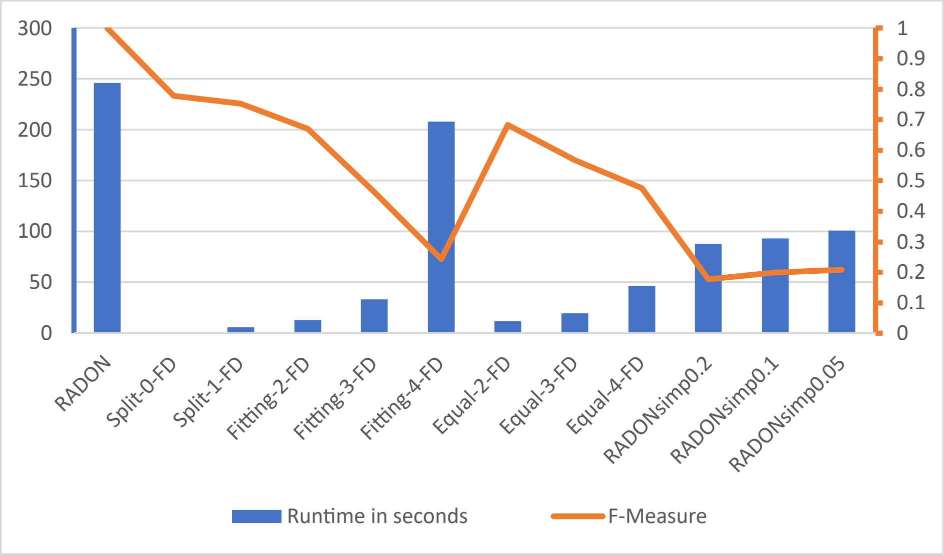

Fig. 6.10.: The `touches` relation. Runtime in seconds (blue) and F-measure (orange) results for linking NUTS×NUTS.

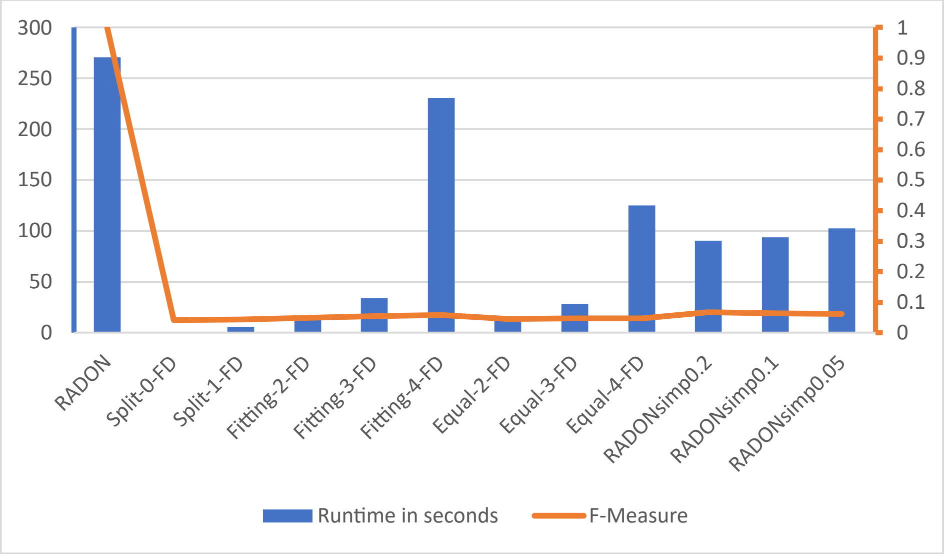

Fig. 6.11.: The `overlaps` relation. Runtime in seconds (blue) and F-measure (orange) results for linking NUTS×NUTS.

NELLIE: Never-Ending Linking for Linked Open Data

7

Preamble: This chapter is based on Ahmed et al. [ASN23] that introduces NELLIE. NELLIE is a pipeline architecture to build a chain of modules in which each module addresses a challenge of data augmentation. The ultimate goal of the proposed architecture's goal is to build a single fused KG from the LOD. NELLIE starts by crawling the available KGs in the LOD cloud. Then, it finds a set of matching KG pairs. NELLIE uses a two-phase linking approach for each pair: first, an ontology matching phase, then an instance matching phase. Based on ontology and instance matching, NELLIE fuses each pair of KGs into a single KG. The resulting fused KG is an ideal data source for knowledge-driven applications such as search engines, question answering, digital assistants, and drug discovery. Our evaluation shows an improved score of $Hit@1$ of the link prediction task on the resulting fused KG by NELLIE in up to 94.44% of the cases. Our evaluation also shows a runtime improvement by several orders of magnitude when comparing our two-phase link approach with the estimated runtime of linking using a naïve approach.

7.1 Motivation

The number of heterogeneous KGs that obey the principles of linked data increases steadily. These KGs are broadly used in data-driven applications, including information retrieval, NLP, recommendation systems, search engines, conversational agents, e-commerce solutions, and drug discovery. Currently, there are no holistic models for the LOD to build a single fused KG out of the LOD, i.e., the development of a 24/7 solution (similar to Never Ending Language Learner) to fuse KGs on the LOD.

For LOD to have such a complete model, the instances and ontologies in each KG must be linked. Currently, only a small number of such KGs are linked. In particular, the current statistic[1] of LOD[2] shows that there are 1564 KGs with 395.121

[1] Accessed 10.03.2022 https://lod-cloud.net/#about, retrieved using https://github.com/lod-cloud/lod-cloud-draw/blob/master/scripts/count-data.py

[2] https://lod-cloud.net/

billion triples and only 2.72 billion links (0.07%) between them. Therefore, finding links among these KGs is a major challenge in achieving the vision behind LOD[3]. Establishing links is a tedious process when performed manually, especially in giant KGs such as *DBpedia*[4], LGD [5] [ALH09], Bio2RDF[6], KEGG [Kan+16], and *Wikidata*[7]. In addition to the ever-increasing number of published KGs, the size of individual KGs increases with each new edition. For example, *DBpedia* has grown from 103 million triples (`DBpedia` 2.0), representing 1.95 million entities, to 10.094 billion triples, representing more than 8.85 million entities in 2022[8].

Moreover, as the number of independent data providers increases, it is more likely that the simultaneous publication of KGs with the same information will take place. For instance, `DBLP` has been published by several bodies[9], leading to duplicate content in the Data Web. Furthermore, different KGs contain different facets related to the same data. For example, drug data within the `DrugBank`[10] KG mainly describes the drugs' interactions, pharmacology, chemical structures, targets, and metabolism, while the `Sider`[11] KG contains data concerning drug side effects. As a result of such huge data expansion and multifaceted data publishing, there is a growing demand for data augmentation tasks such as ontology and instance linking, as well as data fusion. Many frameworks have been developed to address different data augmentation challenges. Prior to fusing KGs, such systems mainly identify semantically equivalent entities in different KGs where they try to achieve both high effectiveness and efficiency in the linking process. For example, *LogMap* [JC11] and CODI [NMS10] use structural matching based on the ontology structure to discover links between ontologies. *Nentwig et al.* [Nen+17] list many data augmentation systems that have been developed in the last two decades. For example, LIMES [Ngo11; Ngo+21] and SILK [IJB11] apply matching strategies on the instance level for computing the property values. *Nentwig et al.* [Nen+17] address many challenges and aspects of the current LD frameworks. In a more recent survey [MT19], Mountantonakis and Tzitzikas presented some linked data integration approaches, including LD and KG fusion. For fusing data, the linked data quality assessment and fusion SIEVE [MMB12] is proposed, which is integrated into the linked data integration framework (LDIF) [Sch+11]. DEER [SNL15] is another data

[3] `https://www.w3.org/DesignIssues/LinkedData.html`

[4] `https://wiki.DBpedia.org/`

[5] `http://linkedgeodata.org/About`

[6] `https://download.bio2rdf.org/release/4`

[7] `https://www.wikidata.org/`

[8] Accessed in 10.03.2022 from `https://DBpedia.org/sparql`

[9] `http://dblp.l3s.de/`, `http://datahub.io/dataset/fu-berlin-dblp` and `http://dblp.rkbexplorer.com/`

[10] `https://go.drugbank.com/`

[11] `http://sideeffects.embl.de/`

augmentation framework that is able to perform LD and fusion to produce enriched data.

In this work, we propose NELLIE, a pipeline architecture to build a chain of modules in which each of our modules addresses a data augmentation challenge. Figure 7.1 shows the architecture of NELLIE. NELLIE first addresses the problem of finding relevant KGs to be integrated. Subsequently, NELLIE tackles the KG data integration task at both the ontology and instance levels. NELLIE then fuses the matched classes and instances to generate a fused KG. Finally, NELLIE carries out KG embedding of the resulting fused KG. The ultimate goal of the proposed architecture is to build a single fused KG out of the LOD, i.e., the development of a 24/7 solution (similar to the Never Ending Language Learner) to fuse KGs in the LOD, especially since such a graph does not yet exist.

Our proposed architecture consists of three layers: the core layer, the application layer, and the publication layer. In this chapter, we pay more attention to the core layer, as it contains the main components and modules of our architecture. In particular, we address the following challenges: 1) KG matching (i.e., matching KGs based on their content); 2) KG linking, including ontology and instance matching; 3) KG fusion; and 4) KG embedding. Note that all these challenges are implemented in our core layer. In the application layer, we address the link prediction challenge to evaluate the impact of our KG fusion on the link prediction task. We summarize our contributions as follows:

- We develop a modular pipeline architecture as a milestone toward the 24/7 linking and fusing of the LOD.
- We propose the two-phase linking strategy, starting with ontology matching and then instance matching.
- In the KG matching stage, we implemented the three methods presented by ourselves.
- In the ontology matching stage, we implemented content-based class matching ourselves and integrated two state-of-the-art systems.
- For the instance matching stage, we base our implementation on the state-of-the-art LIMES LD framework, LIMES [Ngo+21], where we modified how to train the WOMBAT [SNL17] to generate link specifications. We then integrated LIMES into NELLIE, as listed in Algorithm 5.

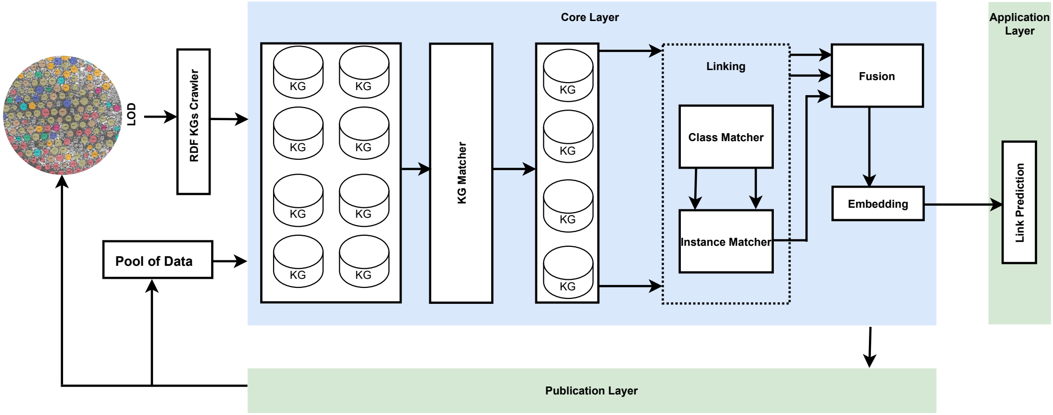

Fig. 7.1.: The modular pipeline architecture of NELLIE.

- In the KG fusion stage, we implemented the additive fusion operator with many different fusion strategies. Finally, we study the impact of KG fusion on the link prediction task.

We evaluated our two-phase linking by computing a pseudo-F-measure. We also evaluated our approach to the link prediction task and studied the impact of KG fusion on this task. We used different KGs and different link prediction models. Evaluating the efficiency and dependability of NELLIE as a whole is worth considering but is currently too resource-intensive to implement. We used existing benchmarks for the sake of comparability. However, we agree that the benchmarks we have now are made for specific subtasks, such as link prediction, ontology matching, and instance matching.

7.2 Knowledge Graph Matching

7.2.1 Approach

For the current version of NELLIE, we implemented three methods for KG matching: metadata-based KG matching, content-based KG matching, and manual KG matching.

Metadata-Based KG Matching

In this method, we first collect the KG metadata from the LOD Cloud[12]. The KG metadata includes various features of each KG, such as links to other KG websites, SPARQL endpoints, keywords, and domain. We then configure the LD framework LIMES [Ngo+21] to match KGs using string similarities such as `Jaccard` and `Cosine` to compute the similarity among both `keywords` and `domain` features. We provide the full LIMES configuration file in Listing 2.

Content-Based KG Matching

For each KG, we retrieve all the text within the literal objects using the SPARQL query in Listing 1. We then concatenate all the literals contained in each KG to generate a content document for each input KG. Afterward, we preprocess each KG content document by applying the following:

1. *Tokenization.* We perform word tokenization by breaking a raw text into words (tokens) using *White Space Tokenization*[13].

2. *Stop words removal.* We remove all stop words, such as {`a, an, the, in,` $\cdots$}, to increase performance during the string similarity measure.

3. *Text Normalization.* We use the `Normalization Form KC (NFKC)`[14]. Next, we clean the text from numbers and special symbols using regular expressions.

To this end, a set of tokens as a first document $A = \{a_1, a_2, ..., a_n\}$ and a set of tokens as a second document $B = \{b_1, b_2, ..., b_n\}$ are produced. We used the similarities of `Jaccard`, `Cosine` with TF-IDF document vectors, `weighted Jaccard`, `Dice`, and `BERT` [Dev+18] to calculate the similarities between A and B.

Given $A = \{a_1, a_2, \ldots, a_n\}$, a set of tokens representing the first document, and $B = \{b_1, b_2, \ldots, b_n\}$, a set of tokens representing the second document, we compute the `weighted Jaccard` similarity between the two documents as defined in [F C21].

[12] `https://lod-cloud.net/lod-data.json`

[13] `https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/process/WhitespaceTokenizer.html`

[14] `https://docs.oracle.com/javase/tutorial/i18n/text/normalizerapi.html`

```
SELECT DISTINCT  ?literal
WHERE {  ?s ?p ?literal
  FILTER isLiteral(?literal) }
```

Listing 1: SPARQL query to retrieve literal objects.

For Cosine with TF-IDF document vectors, we used the framework[15] developed by DKPro. We first started by calculating the vectors of the TF-IDF document for each document and second by applying cosine similarity.

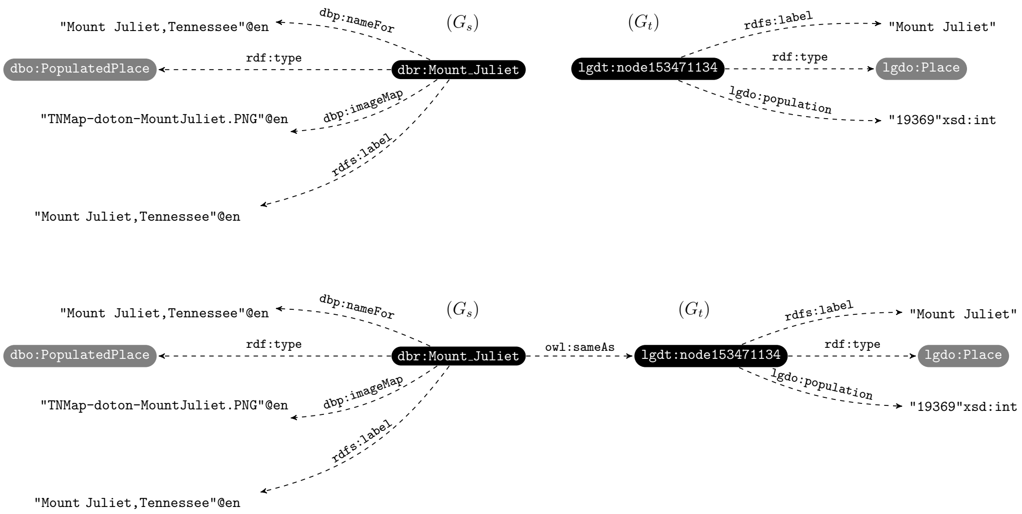

Fig. 7.2.: Example of the DBpedia resource dbr:Mount_Juliet with the LinkedGeoData resource lgdt:node153471134. We used owl:sameAs to link the DBpedia resource dbr:Mount_Juliet with the LinkedGeoData resource lgdt:node153471134.

Manual KG Matching

In order to evaluate the performance of NELLIE within a small set of KGs, we manually select some KGs that belong to the *biology* domain: Kegg, Drugbank, Sider, Omim, and Sgd. We also select to match the two KGs LGD and DBpedia. Although LGD belongs to the geographic domain and DBpedia belongs to general domain, they still have the potential to be linked since they have many classes in common (e.g., organization, place, location, and city) with many instances that refer to the same physical facts. For example, :Mount_Juliet,_Tennessee is a city located in western Wilson County, Tennessee, as described in DBpedia, and :node153471134 refers to the same city in LGD, which we used as our running example (see Figure 7.2).

[15] https://github.com/dkpro/dkpro-similarity

Tab. 7.1.: Manual annotation results of our KG matching. A_1 to A_3 are our three annotators; MA is the mutual agreement.

KG_s	KG_t	A_1	A_2	A_3	MA
harvard_eagle-i_net_sparqler.nt	onto_fel_cvut_cz_rdf4j-server_repositories.nt	✗	✗	✗	✗
harvard_eagle-i_net_sparqler.nt	ldf_fi_ww1lod.nt	✓	✗	✓	✓
harvard_eagle-i_net_sparqler.nt	lov_linkeddata_es_dataset_lov.nt	✓	✓	✓	✓
harvard_eagle-i_net_sparqler.nt	dbtune_org_bbc_peel_sparql.nt	✓	✓	✓	✓
harvard_eagle-i_net_sparqler.nt	www_imagesnippets_com_sparql.nt	✓	✓	✓	✓
harvard_eagle-i_net_sparqler.nt	dbtune_org_magnatune_sparql.nt	✓	✗	✓	✓
harvard_eagle-i_net_sparqler.nt	data_nobelprize_org.nt	✓	✓	✓	✓
onto_fel_cvut_cz_rdf4j-server_repositories.nt	ldf_fi_ww1lod.nt	✗	✗	✗	✗
onto_fel_cvut_cz_rdf4j-server_repositories.nt	lov_linkeddata_es_dataset_lov.nt	✓	✓	✓	✓
onto_fel_cvut_cz_rdf4j-server_repositories.nt	dbtune_org_bbc_peel_sparql.nt	✗	✓	✗	✗
onto_fel_cvut_cz_rdf4j-server_repositories.nt	www_imagesnippets_com_sparql.nt	✗	✓	✗	✗
onto_fel_cvut_cz_rdf4j-server_repositories.nt	dbtune_org_magnatune_sparql.nt	✗	✓	✗	✗
onto_fel_cvut_cz_rdf4j-server_repositories.nt	data_nobelprize_org.nt	✗	✗	✗	✗
ldf_fi_ww1lod.nt	lov_linkeddata_es_dataset_lov.nt	✓	✗	✓	✓
ldf_fi_ww1lod.nt	dbtune_org_bbc_peel_sparql.nt	✓	✓	✓	✓
ldf_fi_ww1lod.nt	www_imagesnippets_com_sparql.nt	✓	✓	✗	✓
ldf_fi_ww1lod.nt	dbtune_org_magnatune_sparql.nt	✓	✓	✓	✓
ldf_fi_ww1lod.nt	data_nobelprize_org.nt	✓	✓	✓	✓
lov_linkeddata_es_dataset_lov.nt	dbtune_org_bbc_peel_sparql.nt	✓	✓	✓	✓
lov_linkeddata_es_dataset_lov.nt	www_imagesnippets_com_sparql.nt	✓	✓	✓	✓
lov_linkeddata_es_dataset_lov.nt	dbtune_org_magnatune_sparql.nt	✓	✓	✓	✓
lov_linkeddata_es_dataset_lov.nt	data_nobelprize_org.nt	✓	✓	✓	✓
dbtune_org_bbc_peel_sparql.nt	www_imagesnippets_com_sparql.nt	✓	✓	✗	✓
dbtune_org_bbc_peel_sparql.nt	dbtune_org_magnatune_sparql.nt	✓	✓	✓	✓
dbtune_org_bbc_peel_sparql.nt	data_nobelprize_org.nt	✓	✓	✓	✓
www_imagesnippets_com_sparql.nt	dbtune_org_magnatune_sparql.nt	✓	✓	✓	✓
www_imagesnippets_com_sparql.nt	data_nobelprize_org.nt	✓	✓	✓	✓
dbtune_org_magnatune_sparql.nt	data_nobelprize_org.nt	✓	✓	✓	✓

7.2.2 Evaluation

Metadata-Based KG Matching

We started by retrieving the global metadata of all KGs available on the LOD.[16] In total, we retrieved the metadata of 1118 KGs. We converted the metadata into RDF format.[17] We then used the LD framework LIMES for matching KGs based on their metadata. The LIMES configuration file we used to match the KGs' metadata

[16] https://lod-cloud.net/lod-data.json, accessed in October 2022

[17] https://git.cs.uni-paderborn.de/kgfusionpg/kgfusion/-/blob/main/lod_metadata_11_2022.ttl

Tab. 7.2.: Content-based KG matching.

Similarity Method	Cosine TF-IDF	Jaccard	Weighted Jaccard	DICE	BERT
Precision	0.95	1.0	1.0	1.0	0.79
Recall	0.95	1.0	0.68	0.86	1.0
F-Measure	0.95	1.0	0.81	0.93	0.88

is presented in Listing 2 and is also publicly available on the project web site[18]. In particular, we configure LIMES to compute the *exact match* string similarity between the `keywords` and `domain` properties. With this configuration, LIMES generated 186452 links. We believe that metadata-based KG matching using LIMES is an efficient approach in the case of datasets with rich metadata.

Content-Based KG Matching

To the best of our knowledge, there is no benchmark for the KG matching task; therefore, we had to create our own benchmark. In particular, we chose a list of 8 KGs and manually annotated them by three annotators into either *matched* (✓) or *not matched* (✗). We then computed the mutual agreement (*MA*) of our three annotators as listed in Table 7.1. To this end, we applied our content-based KG matching, as described in Section 7.2.1. We set the threshold of similarity between the generated documents of KGs to be ≥ 0.1. Using the mutual agreement among our annotators (*MA*) as the ground truth, we computed the precision, recall, and band F-measure among the KG content documents using the `Cosine`, `Jaccard`, `Weighted Jaccard`, `Dice`, and `BERT` similarity measures. The results are listed in Table 7.2. In particular, we achieve an F-measure of 1.0 using `Jaccard` similarity and 0.95 using `Cosine-TF-IDF` similarity. On the other hand, using the `BERT` similarity resulted in an F-measure of only 0.88. However, the *document similarity scores* resulting from using the `BERT` similarity are, in general, higher than the other similarity measures such as `Jaccard`. The reason is that `BERT` is an advanced language model that takes into account semantic, contextual and relation between words in its word representation vectors. For computing `Dice` similarity, we used the open-source Java library `SimMetrics`[19]. We use the pre-trained BERT from `HuggingFace`[20] for the embedding of the preprocessed documents, where we calculate the similarity of vectors using the cosine similarity.

[18] https://git.cs.uni-paderborn.de/kgfusionpg/kgfusion/-/blob/main/LIMESConfig.xml

[19] https://github.com/Simmetrics/simmetrics

[20] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE LIMES SYSTEM "limes.dtd">
<LIMES>
  <PREFIX>
    <NAMESPACE>http://www.w3.org/2000/01/rdf-schema#</NAMESPACE>
    <LABEL>rdfs</LABEL>
  </PREFIX>
  <PREFIX>
    <NAMESPACE>http://www.w3.org/2002/07/owl#</NAMESPACE>
    <LABEL>owl</LABEL>
  </PREFIX>

  <SOURCE>
    <ID>S</ID
    <ENDPOINT>lod\_metadata\_11\_2022.ttl</ENDPOINT>
    <VAR>?x</VAR>
    <PAGESIZE>-1</PAGESIZE>
    <RESTRICTION> </RESTRICTION>
    <PROPERTY>rdfs:keywords</PROPERTY>
    <PROPERTY>rdfs:domain</PROPERTY>
    <TYPE>NT</TYPE>
  </SOURCE>

  <TARGET>
    <ID>T</ID>
    <ENDPOINT>lod\_metadata\_11\_2022.ttl</ENDPOINT>
    <VAR>?y</VAR>
    <PAGESIZE>-1</PAGESIZE>
    <RESTRICTION> </RESTRICTION>
    <PROPERTY>rdfs:keywords</PROPERTY>
    <PROPERTY>rdfs:domain</PROPERTY>
    <TYPE>NT</TYPE>
  </TARGET>
  <METRIC>AND(exactmatch(x.rdfs:keywords,y.rdfs:keywords)|0.9, exactmatch(x
    .rdfs:domain,y.rdfs:domain)|0.9)
  </METRIC>

  <ACCEPTANCE>
    <THRESHOLD>0.98</THRESHOLD>
    <FILE>accepted.ttl</FILE>
    <RELATION>owl:sameAs</RELATION>
  </ACCEPTANCE>

  <REVIEW>
    <THRESHOLD>0.80</THRESHOLD>
    <FILE>review.ttl</FILE>
    <RELATION>owl:sameAs</RELATION>
  </REVIEW>

  <EXECUTION>
    <REWRITER>default</REWRITER>
    <PLANNER>default</PLANNER>
    <ENGINE>default</ENGINE>
  </EXECUTION>

  <OUTPUT>TTL</OUTPUT>
</LIMES>
```

Listing 2: LIMES configuration for metadata-based KG matching.

Tab. 7.3.: Evaluation of KG characteristics.

#	Kegg	Omim	Sider	Drugbank	Sgd
Classes	71	34	12	98	85
Entities	8.627 M	1.127 M	0.479 M	0.421 M	0.991 M
Triples	67.89 M	9.68 M	5.57 M	5.5 M	12.95 M

Manual KG Matching

For *Manual KG matching*, we selected the following KGs from the biological domain: Kegg, Drugbank, Sider, Omim, and Sgd. We selected to manually match these KGs to ensure the best matching of them, as we used them to evaluate all the next components of NELLIE (i.e., ontology matching, instance matching, fusion, and link prediction). Table 7.3 provides the characteristics of these KGs.

7.3 Two-Phase Linking Strategy: Ontology Matching and Instance Matching

7.3.1 Approach

For each pair of matched KGs, we carry out our two-phase linking process. In particular, we first perform ontology matching followed by instance matching. In the following, we explain these two linking phases in detail.

Class Matching

Based on our formal definition of ontology matching in Section 2, we implemented three methods for class matching:

- **Content-based class matching.** We match classes based on the assumption that similar classes describe similar things. Therefore, we measure class similarity based on the overall similarity of the literal objects within those classes.

  We start our class matching process by extracting all classes $\mathcal{C}_s$ and $\mathcal{C}_t$ from source KG G_s and target KG G_t, respectively. As shown in Listing 3, we query

```
PREFIX owl:<http://www.w3.org/2002/07/owl#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?c where { ?c a owl:Class.
    FILTER NOT exists { [] rdfs:subClassOf ?c } }
```

Listing 3: SPARQL query for finding leaf classes.

only for the most specific classes, i.e., the leaf classes, of each KG. However, a more specific list of classes can be provided by the user if necessary.

We then rank the properties for each class by calculating their coverage and pick up only the properties with coverage that exceed a certain propriety-coverage threshold $\beta \in [0,1]$ defined by the user. The goal of the ranking of properties is to make sure that only the most important properties have been retrieved. For example, properties such as (`label`, `name`, `title`) have a high coverage, which leads to the retrieval of more information. Formally, we query for proprieties with $coverage(p) \geq \beta)$, where the coverage is defined as

$$coverage(p) = \frac{|\{s : (s,p,o) \in c_i\}|}{|\{s : \exists q \ (s,q,o) \in c_i\}|}, \tag{7.1}$$

where $c_i \in \mathcal{C}_s$. β is a user-defined value between $[0,1]$. For the target KG, we replace c_i with c_j in Equation 7.1. After extracting all classes and ranking properties, we retrieve only the objects with literal values using SPARQL queries. The retrieved data (i.e., the literals) must be preprocessed before it can be used for the class matching task. We used the preprocessing steps as follows.

1. *Tokenization.* We perform word tokenization by breaking a raw text into words (tokens) using *White Space Tokenization*[21].

2. *Stop words removal.* We remove all stop words such as {`a`, `an`, `the`, `in`, $\cdots$} to increase the performance during string similarity measure.

3. *Text normalization.* We use `Normalization Form KC (NFKC)`[22]. Next, we clean the text from numbers and special symbols using regular expressions.

[21] https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/process/WhitespaceTokenizer.html

[22] https://docs.oracle.com/javase/tutorial/i18n/text/normalizerapi.html

Subsequently, we store the set of distinct tokens (that is, words) that belong to each class $c_i \in G_s$ as $\mathcal{L}_i = \{l_1 \cdots l_n\}$. Formally, $(key = c_i, value = \mathcal{L}_i)$. We repeat the same procedures for each target class $c_j \in G_t$.

Now, the classes, together with their cleaned literal object values, are ready for matching. Our content-based class matching is formally defined as $ClassMatching(c_i, \mathcal{L}_i, c_j, \mathcal{L}_j)$, where $c_i \in G_s$, $c_j \in G_t$, G_s is the source KG and G_t is the target KG. We define the class similarity threshold $\tau \in [0, 1]$. If $StringSimilarity(\mathcal{L}_i, \mathcal{L}_i) \geq \tau$, then c_i is equivalent to c_j. These equivalent pairs of classes are stored in a list of $equivalentClasses(c_i, c_j)$. By default, we use the similarity of the `Jaccard` to measure the similarity between $\mathcal{L}_i$ and $\mathcal{L}_j$. Still, the user can configure NELLIE to use other string similarities.

- **Class matching using LogMap.** LogMap [JC11] can match a pair of ontologies by producing mapped pairs utilizing lexical indexation, structural indexation, and generating anchor mappings. It can even do mapping fixes. It employs various "confidence" levels to obtain an overestimation of the mappings. LogMap is scalable and can successfully match semantically rich ontologies of classes. It also includes techniques for detecting and repairing unsatisfiability on the fly. Furthermore, LogMap generates a ‘clean’ set of output mappings in many circumstances, implying that the ontology acquired by integrating LogMap’s output mappings with the input ontologies is consistent and does not include unsatisfactory classes.

- **Class matching using FCA-Map.** FCA-Map [Zha+18b] is based on formal concept analysis to find and evaluate mappings across ontologies, including one-to-one mappings, complicated mappings, and correspondences between object characteristics. It generates lexical mappings from class names and labels, as well as mappings based on ontology structures. FCA-Map generates three types of formal contexts before extracting mappings from the resultant lattices. To begin, the token-based formal context illustrates how class names, labels, and synonyms all share lexical tokens, leading to lexical mappings (anchors) between ontologies. Second, the relation-based formal context specifies how classes are connected to anchors in taxonomic, partonomic, and disjoint ways, yielding positive and negative structural evidence for lexical matching validation. Third, the positive relation-based context may be leveraged to find more structural mappings once the incoherence has been rectified [ZZ16]. Thus, we can use FCA-Map to extract lexical and structural mappings of matched classes, objects, and data attributes.

For LogMap and FCA-Map, first, we can get the output of the preceding step, that is, the KG matching, in two separate forms. The first form is a model with pairs of SPARQL endpoints that are the same to a certain degree. The second form of the output is the pairs of KGs that are matched. The outputs with the matched SPARQL endpoints are queried to retrieve RDF datasets, which are then saved locally. Second, we also can feed LogMap and FCA-Map with two sets of ontologies directly, as we did in our experiments. We ran LogMap and FCA-Map as standalone. Note that this class matching phase reduces the runtime needed for the instance matching phase (our second linking phase), as we only perform instance matching among instances of the matched pairs of classes.

Instance Matching

Based on our formal definition of the matching of instances in Section 2.4, we focus on `owl:sameAs` as a relation r between s and t. We rely on LIMES [Ngo+21] as it is a state-of-the-art declarative LD framework with open source implementation that can be easily adopted and extended in NELLIE. Algorithm 5 shows the procedures we follow to establish the link between the instances of the source KG G_s and the target KG G_t. Computing the mapping *M* among all instances of the source and target KGs in a trivial way would result in quadratic complexity, i.e., $O(|G_s| \times |G_t|)$. Therefore, we calculate an approximate mapping $M' = \{(s,t) \in G_s \times G_t \ \texttt{owl:sameAs(s,t)} \geq \theta\}$, where θ is a threshold between $[0,1]$, to filter out all pairs with similarity measures less than θ.

LSs need to be generated, either manually or automatically, to express the conditions necessary to link resources within these KGs. An LS consists of two types of atomic components: *similarity measures* m and *operators* op. *Similarity measures* m are used to compare the property values of input instances and *operators* op that link these similarities with more complex specifications. We define a similarity measure m as a function $m : G_s \times G_t \to [0,1]$. An LS is called *atomic* when it only contains one similarity measure, while a complex specification (*complex LS*) can be obtained by gluing two specifications, L_1 and L_2, through an *operator* op that combines the results of two LSs, L_1 and L_2. Here, we use the operators $\sqcap$, $\sqcup$, and $\setminus$ as they are complete and frequently used to define LS [SNL17] (see also Section 2.4.1). An LS is also called a linkage rule in the literature [IJB11]).

In NELLIE, we use the state-of-the-art algorithm WOMBAT [SNL17] to automatically generate LSs. WOMBAT learns LSs based on the concept of generalization in quasi-ordered spaces. We use an unsupervised version of WOMBAT. While lines 1–11 in Algorithm 5 describe the configuration and preparation for instance matching using LIMES, lines 12–18 describe the preparation of caches to train WOMBAT. The goal of the procedures stated in lines 12–18 is to reduce training time in case there is a large KG. For instance, the idea in line 12 is to filter out any cache that has a small number of instances (that is, less than 100). Consequently, we define an integer parameter mcs, i.e., the minimum cache size, for both the source and target caches. For example, setting mcs to 100 means that the caches must contain at least 100 instances. In addition to the parameter mcs, we define a second integer parameter, mss, that refers to the *minimum sample size*. As the large size of the cache increases the time to train WOMBAT, the parameter mss plays an important role by training only a sample of the data if the cache size is greater than the mss. For example, if we have a source cache size of 10000 instances, a target cache size of 5000 instances, and a mss with the value of 4000, we then select a sample size of 4000 instances from the smaller cache, which is the target cache in our example here. By taking the sample from the smaller cache, we have a better chance of finding matches, if such matches exist. This is shown in lines 13–16 in Algorithm 5. We then train WOMBAT using the source and target training caches from the previous step in line 19 to generate the best link specification. Using LIMES, we generate the mapping among the instances of the pair of input classes in line 20 by applying the best link specification to the original KG.

7.3.2 Evaluation

For the *Linking Task*, we set the configuration of NELLIE as follows: *propriety-coverage* threshold $\beta = 0.5$ and $\tau = 0.2$ as a minimum threshold for string similarity when computing equivalent classes. In other words, we select all properties that have $\beta \geq 0.5$ and classes that have similarity $\geq \tau = 0.2$. We also configure the parameters of WOMBAT as follows: We set the string similarity measures to {`jaccard, cosine, qgrams, levenshtein`}; the maximum iteration number to 10; the maximum execution time to 200 minutes; and the minimum coverage of properties to 0.9.

Content-Based Class Matching

In Table 7.3, we present the results of applying the content-based class matching to the classes within the manually matched KGs from our previous step. In particular,

Algorithm 5: Linking of Knowledge Graphs

```
input  : $EQ = \{(c_{s_1}, c_{t_1}) \ldots (c_{s_n}, c_{t_n})\}$ is the list of equivalent classes
input  : $mcs$ is the minimum cache size
input  : $mss$ is the minimum saple size
output : approximate mapping: $M' = \{(s,t) \in G_s \times G_t : sameAs(s,t) \geq \theta\}$
foreach $(c_{s_i}, c_{t_i}) \in EQ$ do
    $p_{s_i}$=GetPropertiesWitAtLeast Covering($c_{s_i}, \beta$);
    $p_{t_i}$=GetPropertiesWitAtLeast Covering($c_{t_i}, \beta$);
    // Configure LIMES
    LIMES.sourceKG($G_s$);
    LIMES.sourceKGRestriction($c_{s_i}$);
    LIMES.sourceKGProperties($p_{s_i}$);
    $souceCache$ = LIMES.fillSourceCache();
    LIMES.targetKG($G_t$);
    LIMES.targetKGRestriction($c_{t_i}$);
    LIMES.targetKGProperties($p_{t_i}$);
    $targetCache$ = LIMES.fillTargetCache();
    if $souceCache.size() > mcs$ & $targetCache.size() > mcs$ then
        if $souceCache.size() > mss$ & $targetCache.size() > mss$ then
            $sourceTrainingCache = Max(souceCache, targetCache)$;
            $targetTrainingCache = Sample(Min(souceCache, targetCache))$;
        else
            $sourceTrainingCache = souceCache$;
            $targetTrainingCache = targetCache$;
    $BestLS$ = LIMES.runUnsupervisedWombat(
     $sourceTrainingCache, targetTrainingCache$);
    $M' = $ LIMES$(sourceCache, targetCache, BestLS)$;
return $M'$
```

we computed the Jaccard string similarity between the cleaned literals of the pairs of classes within each pair of KGs, as described in Section 7.3.1. Consequently, a pair of classes is matched if there is similarity between their respective cleaned literals ≥ 0.2. In the next step, we run LIMES on each pair of matched classes to link the instances with these classes, following the procedures defined in Algorithm 5. In Table 7.4, we list the matched classes within the manually matched KGs (from Section 7.3). For example, in the case of *DBpedia* and LGD, we have the matched classes: {*(Settlement, Place)*, *(Settlement, Village)*, ..., *(Place, Village)*}. Algorithm 5 then runs LIMES on each pair of the matched classes. We also calculate the pseudo-F-measure F [NL13] for the instances with each pair of classes. The basic assumption behind the pseudo-F-measure is that symmetrical one-to-one links exist between the resources in source and target datasets. For example, pseudo-F-measure F= 0.89 for the matched classes *(Settlement, Place)* of *DBpedia* and LGD. We used

Tab. 7.4.: Class matching results. $|c|$ is the number of instances of a class c. Time is in milliseconds. F is pseudo-F-measure.

G_s	G_t	Source class (c_s)	$\|c_s\|$	Target class (c_t)	$\|c_t\|$	Time	$\mathcal{F}$
Kegg	**Omim**	uniprot_vocabulary:Resource	25009	uniprot_vocabulary:Resource	14259	180686	0.82
		ncbigene_vocabulary:Resource	39726	ncbigene_vocabulary:Resource	16080	227034	0.68
		ensembl_vocabulary:Resource	22171	ensembl_vocabulary:Resource	29006	586008	0.67
						Average macro Pseudo-F-Measure	**0.72**
Kegg	**Drugbank**	cas_vocabulary:Resource	21271	cas_vocabulary:Resource	3167	17261	0.35
		atc_vocabulary:Resource	3775	atc_vocabulary:Resource	1739	5	0.77
						Average macro Pseudo-F-Measure	**0.56**
Kegg	**Sgd**	go_vocabulary:Resource	3665	go_vocabulary:Resource	5555	6094	0.38
		ec_vocabulary:Resource	6172	ec_vocabulary:Resource	444	570	0.27
						Average macro Pseudo-F-Measure	**0.32**
Drugbank	**Sider**	pubchem.compound_vocabulary:Resource	6111	pubchem.compound_vocabulary:Resource	2097	8376	0.31
Drugbank	**Omim**	uniprot_vocabulary:Resource	8392	uniprot_vocabulary:Resource	14259	62370	0.34
Drugbank	**Sgd**	pfam_vocabulary:Resource	1819	pfam_vocabulary:Resource	3488	2291	0.46
DBpedia	**LGD**	Settlement	11705	Place	11485	22516	0.89
		Settlement	11705	Village	8443	15756	0.83
		PopulatedPlace	11318	Place	11485	16709	0.89
		PopulatedPlace	11318	Village	8443	11966	0.83
		Place	11367	Place	11485	17717	0.89
		Place	11367	Village	8443	15174	0.83
						Average macro Pseudo-F-Measure	**0.86**

an unsupervised version of the Wombat algorithm [SNL17] to calculate pseudo-F-measure F. We also calculated the average macro pseudo-F-measure in case there is more than one pair of matched classes, such as *(DBpedia, LGD)*, *(Kegg, Omim)*, *(Kegg, Drugbank)*, and *(Kegg, Sgd)*. The average macro pseudo-F-measure is listed in Table 7.4 in boldface font. We also recorded the time required to link the instances with each pair of matched classes. On the contrary, it does not take into account the co-occurrence of the words and the context and the relation between words.

LogMap and FCA-Based Class Matching

We integrate the ontology matching components of *LogMap* and *FCA* into the Nellie ontology matching phase. For the evaluation of each of the two systems, please refer to the original papers of these systems [Che+21b; Zha+18b].

By applying the ontology matching phase prior to the instance matching phase, we aim to reduce the overall runtime of the linking procedure. In particular, when applying a single-phase linking of all-against-all instances directly, we would need $\left(\sum\limits_{i=1}^{n}|C_i|\right)\left(\sum\limits_{j=1}^{n}|D_j|\right)$ comparisons between the instances of the leaf classes C_i of the source KG G_s and the leaf classes D_j of the target KG G_t. Note that we assume that both G_s and G_t have the same number of classes n without loss of generality. W.l.o.g, we will also assume that the classes are ordered in such a manner that the first $k \leq n$ classes match. When using our two-phase linking, we need n^2

comparisons for the first linking phase, i.e., class matching. Note that, in general, the average class in a KG has a magnitude greater than the total number of classes. Hence, $n << \sum_{i=1}^{n} \left(\frac{|C_i|}{n}\right)$. The analogy holds for G_t. For the second linking phase, i.e., instance matching, we need $\sum_{i=1}^{k} |C_i||D_i|$ for the k pairs of matched leaf classes from G_s and G_t. This gives us a total cost of $\left(\sum_{i=1}^{k} |C_i||D_i| + n^2\right)$ comparisons of our two-phase linking. Our gain is then the difference between the number of comparisons of all-against-all instance linking and our two-phase linking. Given that $n << \sum_{i=1}^{n} \left(\frac{|C_i|}{n}\right)$, the expected value of this difference is positive for $k < n$. Empirically, we can compute the speedup achieved by our two-phase linking as the number of comparisons using all-against-all instance linking divided by the number of comparisons of our two-phase linking. For example, if we carry out the all-against-all for the KGs `Drugbank` and `Sgd`, we get $0.421 \times 10^6 \times 0.991 \times 10^6 = 41.211 \times 10^{10}$ comparisons. On the other hand, the number of comparisons using our two-phase linker only needs $1819 \times 3488 + n^2 = 6.3446 \times 10^6 + 8330$, given that $k = 1$, where $|C_1| = 1819$, and $|D_1| = 3488$, and n^2 is 8330. Consequently, our speedup here is

$$\frac{41.211 \times 10^{10}}{6.3446 \times 10^6 + 8330} = 6.48695 \times 10^4$$

which is 4 orders of magnitude. From Table 7.4, we see that our two-phase linking strategy maintains and achieves 0.86 average macro pseudo-F-measure for DBpedia and LGD.

7.4 Knowledge Graphs Fusion And Link Prediction

7.4.1 Approach

Merging and Fusion of Knowledge Graphs

Merging and fusion are implemented in a straightforward manner. *Merging* is a process of combining all produced *mappings (i.e., links of instances)* from the process of interlinking a pair of KGs. Formally, $M_{merge} = M_{c1} \cup M_{c2} \cup M_{c3} \cup \cdots \cup M_{cn}$. In order to perform the KG fusion, we merge all mappings $M_{set} = \{M_1, \cdots, M_n\}$ of the matched instances (from the previous linking step) into one universal mapping $M_{merge} = M_1 \cdots \cup M_n$. We dub this task as the merging task. Accordingly, the KG fusion task uses M_{merge} to fuse G_s and G_t. In the following, we present our fusion operator and the strategies implemented so far in NELLIE.

- **Additive Fusion Operator.** Based on the mapping M_{merge} among the resources from G_s and G_t, we implement our additive fusion operator to combine all resources from G_s and G_t. In particular, our additive fusion operator starts by adding all the resources of G_s to the fused KG $G_{s \oplus t}$. Then, it combines all the resources of G_t fused with all similar resources from G_s, where all the subjects of the fused resources are from G_s. We present our additive fusion operator formally in Algorithm 6. Figure 7.3 shows an example of fusing one DBpedia resource with one LGD resource using our additive fusion operator. Note that our operator is additive in the sense that we keep all triples of source KG G_s, even the ones without similar triples in G_t (see Figure 7.4 as an example).

Algorithm 6: KG additive fusion algorithm.

```
input  : Source KG $G_s$,
         Target KG $G_t$,
         Mapping $M_{merg} = \{(x, y) | x \in G_s, y \in G_t\}$
output : Fused KG $G_{s \oplus t}$
$G_{s \oplus t} = G_s$
foreach Mapping pair $(x, y) \in M_{merg}$ do
    foreach $triple(< y, p, o >) \in G_t$ do
        $G_{s \oplus t} = G_{s \oplus t}.addTriple(< x, p, o >)$
return $G_{s \oplus t}$
```

- **Fusion Strategies** After applying our additive fusion operator, we define a number of type-based strategies for fusing the literal objects of the same subject and predicate. For example, in our example in Figure 7.3, we have the triples `<dbr:Mount_Juliet, rdfs:label, "Mount Juliet,Tennessee"@en>` and `<dbr:Mount_Juliet, rdfs:label, "Mount Juliet"@en>`. For the two literal objects `"Mount Juliet,Tennessee"@en` and `"Mount Juliet"@en`, we need to decide to keep one of them, both of them, or to combine them somehow. Formally, for two or more triples $< s, p, \lambda_1 >$ and $< s, p, \lambda_2 >$, we implement the following type-based fusion strategies:
    - `KeepBoth` STRATEGY. We add the two triples $< s, p, \lambda_1 >$ and $< s, p, \lambda_1 >$ to the fused KG.
    - `PreferSource` STRATEGY. We add only the triples $< s, p, \lambda_1 >$ from the source KG to the fused KG.

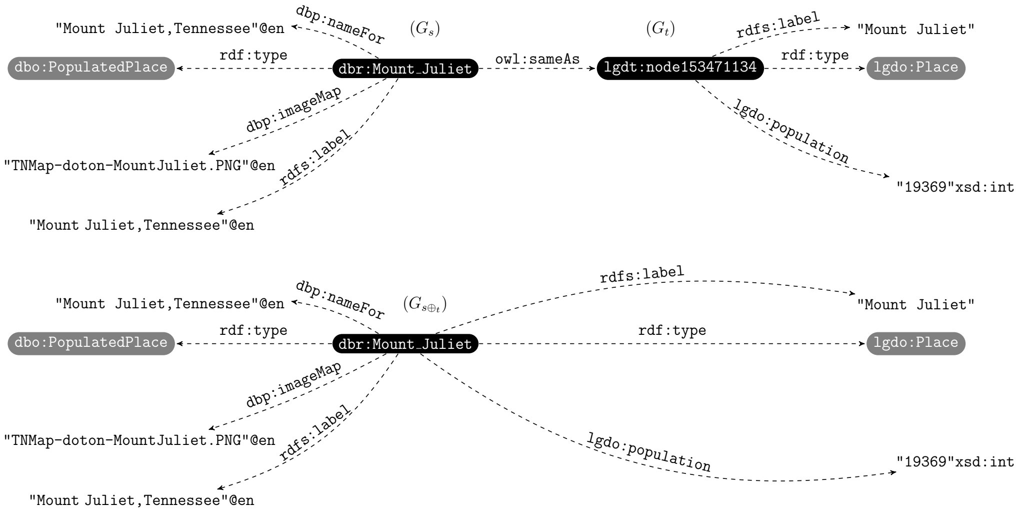

Fig. 7.3.: Example of fusing the DBpedia resource `dbr:Mount_Juliet` with the LGD resource `lgdt:node153471134` using our additive fusion operator from Algorithm 6.

- **`PreferTarget` STRATEGY.** We add only the triples $< s, p, \lambda_2 >$ from the target KG to the fused KG.

- **`Maximum` STRATEGY.** We define the `Maximum` strategy for all numeric literals such as `xsd:integer`[23] and `xsd:decimal`, as well as dates (e.g., `xsd:date`), where we add the triple $< s, p, \max(\lambda_1, \lambda_2) >$ to the resultant fused KG. For string literals, the `Maximum` strategy selects the longest string. Formally, add the triple $< s, p, \arg\max(|\lambda_1|, |\lambda_2|) >$ to the resultant fused KG, where $|\lambda_1|$ is the string length of the string λ_1. For `xsd:boolean`, the triple $< s, p, \lambda_1 || \lambda_2 >$ is added to the fused KG, where $||$ is the logical OR operator.

- **`Minimum` STRATEGY.** Following the same manner of the `Maximum` strategy, we define the `Minimum` strategy for numeric and date literals, where we add the triple $< s, p, \min(\lambda_1, \lambda_2) >$ to the result fused KG. For string literals, the `Minimum` strategy selects the shortest string. i.e., add the triple $< s, p, \arg\max(|\lambda_1|, |\lambda_2|) >$ to the resultant fused KG. For `xsd:boolean`, the triple $< s, p, \lambda_1 \&\& \lambda_2 >$ is added to the fused KG, where $\&\&$ is the logical AND operator.

- **`Average` STRATEGY.** We define the `Average` strategy for numeric and date literals, where we add the triple $< s, p, \frac{1}{2}(\lambda_1 + \lambda_2) >$ to the result

[23] `xsd=<http://www.w3.org/2001/XMLSchema#>`

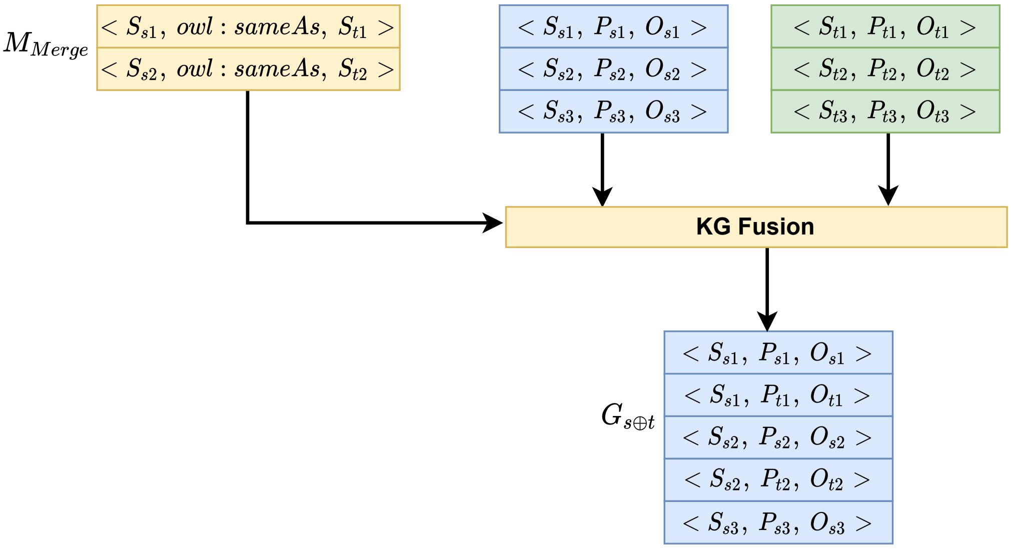
M_{Merge}
$< S_{s1},\ owl : sameAs,\ S_{t1} >$
$< S_{s2},\ owl : sameAs,\ S_{t2} >$
$< S_{s1},\ P_{s1},\ O_{s1} >$
$< S_{s2},\ P_{s2},\ O_{s2} >$
$< S_{s3},\ P_{s3},\ O_{s3} >$
$< S_{t1},\ P_{t1},\ O_{t1} >$
$< S_{t2},\ P_{t2},\ O_{t2} >$
$< S_{t3},\ P_{t3},\ O_{t3} >$
KG Fusion
$G_{s\oplus t}$
$< S_{s1},\ P_{s1},\ O_{s1} >$
$< S_{s1},\ P_{t1},\ O_{t1} >$
$< S_{s2},\ P_{s2},\ O_{s2} >$
$< S_{s2},\ P_{t2},\ O_{t2} >$
$< S_{s3},\ P_{s3},\ O_{s3} >$

Fig. 7.4.: An example of fusing KGs using our additive fusion operator $\oplus$.

Tab. 7.5.: Fusion strategies for fusing the two triples $< s, p, \lambda_1 >$ and $< s, p, \lambda_2 >$.

Method	`xsd:string`	`xsd:integer`	`xsd:date`	`xsd:boolean`
KEEPBOTH	$< s, p, \lambda_1 >$, $< s, p, \lambda_2 >$	$< s, p, \lambda_1 >$, $< s, p, \lambda_2 >$	$< s, p, \lambda_1 >$, $< s, p, \lambda_2 >$	$< s, p, \lambda_1 >$, $< s, p, \lambda_2 >$
PREFERSOURCE	$< s, p, \lambda_1 >$	$< s, p, \lambda_1 >$	$< s, p, \lambda_1 >$	$< s, p, \lambda_1 >$
PREFERTARGET	$< s, p, \lambda_2 >$	$< s, p, \lambda_2 >$	$< s, p, \lambda_2 >$	$< s, p, \lambda_2 >$
MAXIMUM	$< s, p, \arg\max(\vert\lambda_1\vert, \vert\lambda_2\vert) >$	$< s, p, \max(\lambda_1, \lambda_2) >$	$< s, p, \max(\lambda_1, \lambda_2) >$	$< s, p, \lambda_1 \vert\vert \lambda_2 >$
MINIMUM	$< s, p, \arg\max(\vert\lambda_1\vert, \vert\lambda_2\vert) >$	$< s, p, \max(\lambda_1, \lambda_2) >$	$< s, p, \max(\lambda_1, \lambda_2) >$	$< s, p, \lambda_1 \&\& \lambda_2 >$
AVERAGE	$< s, p, \frac{1}{2}(\lambda_1 + \lambda_2) >$	$< s, p, \frac{1}{2}(\lambda_1 + \lambda_2) >$	$< s, p, \frac{1}{2}(\lambda_1 + \lambda_2) >$	$< s, p, \lambda_1 \&\& \lambda_2 >$
UNION	$< s, p, \lambda_1 + \lambda_2 >$	-	-	$< s, p, \lambda_1 \&\& \lambda_2 >$

fused KG. For string literals, the `Average` strategy is not defined. For `xsd:boolean`, the triple $< s, p, \lambda_1 \&\& \lambda_2 >$ is added to the fused KG.

- **`Union` STRATEGY.** We define the `Union` strategy for the literals of type `xsd:boolean`, where we add the triple $< s, p, \lambda_1 \&\& \lambda_2 >$ to the resultant fused KG. For string literals, the `Union` strategy is the string concatenation operator, i.e., the triple $< s, p, \lambda_1 + \lambda_2 >$ is added to the resultant fused KG, where $+$ is the string concatenation operator. The `Union` strategy is not defined for data types of numerical and date.

Table 7.5 lists all the type-based fusion strategies that we have implemented so far. For our experiments, we apply the **KEEPBOTH** strategy.

KG Embedding and Link Prediction

Although there are dozens of embedding models that can be used to perform our link prediction task, we deploy the three embedding models *TuckER* [BAH19], *ComplEx* [Tro+16], and *DistMult* [Yan+15] for embedding in our link prediction task. We use these models as they are state-of-the-art linear models for link prediction on KGs. On the basis of the NELLIE architecture, any other embedding model could be easily added to it.

- **TuckER.** TuckER is based on Tucker decomposition [Tuc64] that factorizes a tensor into a set of matrices and a smaller core tensor. In a three-mode case, given the original tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, Tucker decomposition produces a tensor $\mathcal{Z} \in \mathbb{R}^{P \times Q \times R}$ and three matrices $\mathbf{A} \in \mathbb{R}^{I \times P}$, $\mathbf{B} \in \mathbb{R}^{J \times Q}$, $\mathbf{C} \in \mathbb{R}^{K \times R}$:

  $$\mathcal{X} \approx \mathcal{Z} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}. \tag{7.2}$$

  And, the score function of the TuckER model:

  $$\phi(e_s, r, e_o) = \mathcal{W} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_r \times_3 \mathbf{e}_o, \tag{7.3}$$

  For link prediction in a KG's binary tensor representation, the TuckER model uses Tucker decomposition by constructing an entity embedding matrix $\mathbf{E}$ that is equal for subject and object entities, i.e., $\mathbf{E} = \mathbf{A} = \mathbf{C} \in \mathbb{R}^{n_e \times d_e}$ and the relation embedding matrix $\mathbf{R} = \mathbf{B} \in \mathbb{R}^{n_r \times d_r}$, where n_e and n_r denote the number of entities and relations and d_e and d_r the dimensionality of entity and relation embedding vectors. The TuckER architecture can be seen in Figure 7.5, where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{d_e}$ are the rows of $\mathbf{E}$ representing the embedding vectors of the subject and object entity, $\mathbf{w}_r \in \mathbb{R}^{d_r}$ the rows of $\mathbf{R}$ representing the embedding vector of the relation, and $\mathcal{W} \in \mathbb{R}^{d_e \times d_r \times d_e}$ is the core tensor.

- **DistMult.** The scoring function of DistMult in Table 2.3 can be regarded as equivalent to the scoring function of TuckER in Equation 7.2. The scoring function consists of a core tensor $\mathcal{Z} \in \mathbb{R}^{P \times Q \times R}$, $P = Q = R = d_e$. The *superdiagonal* of $\mathcal{Z}$ is with 1s, i.e., all elements z_{pqr} with $p = q = r$ are 1 and all the other elements are 0. In *DistMult*, the subject and object entities $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{d_e}$ are represented by rows of $\mathbf{E} = \mathbf{A} = \mathbf{C} \in \mathbb{R}^{n_e \times d_e}$, and rows of $\mathbf{R} = \mathbf{B} \in \mathbb{R}^{n_r \times d_e}$ represents the embedding vectors of the relation $\mathbf{w}_r \in \mathbb{R}^{d_e}$. Given that matrices $\mathbf{A}$ and $\mathbf{C}$ are identical, the TuckER interpretation of the DistMult scoring function can alternatively be interpreted as a special case of CP decomposition [Hit27]. DistMult belongs to the family of bilinear models.

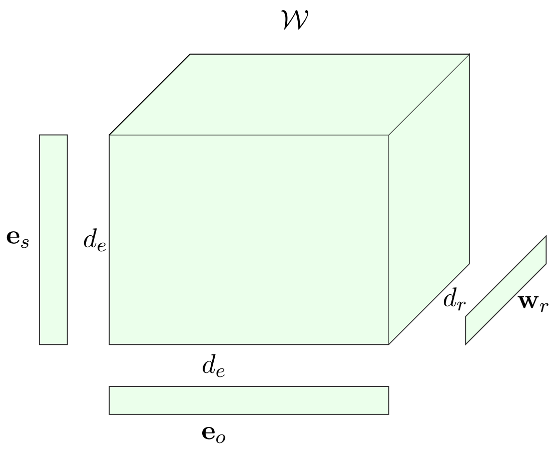

Fig. 7.5.: The TuckER architecture [BAH19].

- **ComplEx.** The scoring function of ComplEx in 2.3 can also be viewed as equivalent to the scoring function of TuckER in Equation 7.2. The core tensor $\mathcal{Z} \in \mathbb{R}^{P \times Q \times R}$, $P = Q = R = 2d_e$, in which $3d_e$ elements on different tensor diagonals are set to 1 and d_e elements on one tensor diagonal are set to -1 while all other elements are set to 0. [KP18] explained that ComplEx can be considered a bilinear model with the real and imaginary parts of an embedding for each entity concatenated in a single vector.

7.4.2 Evaluation

In this task, we study the impact of fusion on the link prediction task. Since the cost of computing and the allocation of resources for the link prediction task in KGs that contain millions of triples is very high, we made two data augmentation scenarios to conduct experiments on data fusion and link prediction tasks:

- *Scenario A.* The idea of this scenario is to study the impact of fused KGs on the quality of the link prediction task. Thus, we used augmented versions of the source KG G_s and target KG G_t by only filtering them to the entities within the *mapping*: $M = \{(s,t)|s \in G_s, t \in G_t\}$. Formally, we augmented our data as follows: Given the *mapping* M, we retrieve the sub-KG $G'_s = \{(s,r,o) \mid s \in G_s$ and $\forall (s,t) \in M\}$ and the sub-KG $G'_t = \{(t,r,o) \mid t \in G_t$ and $\forall (s,t) \in M\}$. To this end, we use Algorithm 6 to fuse the triples of G'_s and G'_t into the fused KG $G_{s \oplus t}$. For evaluating the link prediction task, we compute *Hit@1, Hit@3, Hit@10, and MRR* for the source KGs G'_s and fused KGs $G_{s \oplus t}$. The results are in Tables 7.7, 7.8, 7.9, 7.10, 7.11, and 7.12.

Tab. 7.6.: Hyperparameter values for TuckER, DistMult, and ComplEx across all KGs, where lr denotes learning rate, dr decay rate, and ls label smoothing.

Model	lr	dr	d_e	d_r	ls
TuckER	0.005	1.0	200	200	0.1
DistMult	0.001	0.99	200	200	0.1
ComplEx	0.001	0.99	200	200	0.1

Tab. 7.7.: Link prediction for Drugbank, Omim, and the fused data in *Scenario A*.

Model	**Data**	**Hit@1**	**Hit@3**	**Hit@10**	**MRR**	**|Data|**
TuckER	Source	0.6580	0.6581	0.7223	0.6741	18.545
	Fused	0.5735	0.6209	0.6506	0.6034	20.891
	Improve	-8.45	-3.73	-7.17	-7.07	
DistMulti	Source	0.5458	0.6378	0.6858	0.5997	18.545
	Fused	0.5919	0.6279	0.6750	0.6197	20.891
	Improve	4.61	-0.98	-1.08	2.00	
ComplEx	Source	0.6887	0.7486	0.7704	0.7236	18.545
	Fused	0.5768	0.6468	0.6800	0.6179	20.891
	Improve	-11.19	-10.17	-9.04	-10.58	

- *Scenario B.* The idea of this scenario is similar to that of *Scenario A*. However, we modify *Scenario A* by adding a random subset X of resources from G_s, which are not included within the mapping M, to G'_s. The goal here is to perform the link prediction task on KGs that contain some enriched entities and some non-enriched ones. Formally, $X = \{(s, r, o) \subseteq G_s \mid \forall s \in G_s \wedge s \notin M\}$ and $|X| = |M|$. Note that we limit the size of X to the size of M to keep the balance between the enriched and non-enriched resources. This results in $G'_s = \{(s, r, o) \subseteq G_s \mid \forall s \in M \cup X\}$. To this end, we repeat the procedures performed in *Scenario A*. See the results in Tables 7.13, 7.14, 7.15, 7.16, 7.17, and 7.18.

Setup. We set the hyperparameters as listed in Table 7.6. We followed the same setting, training, and evaluation procedures introduced in the TuckER model [BAH19] and the codebase[24] for TuckER, ComplEx, and DistMult.

Results. We calculate the improvement percentage of each source KG and its fused KG using the formula $(Hit@k_{fusedKG} - Hit@k_{sourceKG}) * 100$. For MRR, we apply a similar formula. For example, the improvement of fused KG in Table 7.7 for $Hit@1$ is

[24] `https://github.com/ibalazevic/TuckER`

Tab. 7.8.: Link prediction for Kegg, Drugbank, and the fused data in *Scenario A*.

Model	Data	Hit@1	Hit@3	Hit@10	MRR	|Data|
TuckER	Source	0.5326	0.5540	0.5999	0.5523	30.351
	Fused	0.4351	0.5207	0.5625	0.4830	34.731
	Improve	-9.75	-3.34	-3.74	-6.92	
DistMulti	Source	0.4172	0.4596	0.4936	0.4459	30.351
	Fused	0.4160	0.4727	0.5043	0.4517	34.731
	Improve	-0.12	1.31	1.07	0.58	
ComplEx	Source	0.4713	0.6075	0.6336	0.5445	30.351
	Fused	0.4057	0.4479	0.4646	0.4311	34.731
	Improve	-6.56	-15.96	-16.90	-11.33	

Tab. 7.9.: Link prediction for Drugbank, Sider, and the fused data in *Scenario A*.

Model	Data	Hit@1	Hit@3	Hit@10	MRR	|Data|
TuckER	Source	0.6460	0.6735	0.723	0.6668	11.511
	Fused	0.6426	0.7669	0.8132	0.7076	25.491
	Improve	-0.34	9.33	8.99	4.08	
DistMulti	Source	0.6086	0.668766	0.7040	0.6446	11.511
	Fused	0.6215	0.7297	0.7964	0.6839	25.491
	Improve	1.29	6.09	9.24	3.93	
ComplEx	Source	0.7040	0.7995	0.8249	0.7569	11.511
	Fused	0.6003	0.7066	0.7727	0.6613	25.491
	Improve	-10.37	-9.29	-5.23	-9.56	

$(0.5008 - 0.3808) * 100 = 12.0\%$ using the DistMulti model. Repeating this procedure, we calculate the values for all KGs. In particular, we found that fused KGs show an improvement of $Hit@1$ compared to $Hit@1$ from source KG. To our knowledge, there is no scientific evidence of the impact of KG alignment or KG fusion on the link prediction task. However, the results show an improvement in all metrics $Hit@1$, $Hit@3$, $Hit@10$, and (MRR). For example, in *Scenario A*, KG fusion improved to 10 $Hit@1$ out of 18 $Hit@1$, which is 55.55% of the cases. While in *Scenario B*, KG fusion improved to 17 $Hit@1$ out of 18 $Hit@1$, which is 94.44% of cases. The results of *Scenario A* for the KG Drugbank and KG Omim (Table 7.7) show that KG fusion improves $Hit@1$, $Hit@3$, $Hit@10$, and (MRR) by up to 12%, 8.31%, 7.66%, and 10.10%, respectively. Therefore, we could see that the KG fusion plays an important role in improving the link prediction task. In *Scenario B*, using TucKER embedding for the KG Drugbank and Omim (Table 7.15), the results show an improvement in

Tab. 7.10.: Link prediction for Kegg, Sgd, and the fused data in *Scenario A*.

Model	Data	Hit@1	Hit@3	Hit@10	MRR	|Data|
TuckER	Source	0.551	0.5982	0.632	0.5809	12.641
	Fused	0.5628	0.6357	0.6761	0.604	21.215
	Improve	1.18	3.75	4.41	2.31	
DistMulti	Source	0.4527	0.5249	0.5649	0.4962	12.641
	Fused	0.5132	0.5861	0.6264	0.5568	21.215
	Improve	6.05	6.12	6.16	6.07	
ComplEx	Source	0.5279	0.6665	0.6928	0.6026	12.641
	Fused	0.5244	0.6	0.6409	0.5679	21.215
	Improve	-0.35	-6.65	-5.19	-3.47	

Tab. 7.11.: Link prediction for Kegg, Omim, and the fused data in *Scenario A*.

Model	Data	Hit@1	Hit@3	Hit@10	MRR	|Data|
TuckER	Source	0.5020	0.5350	0.5541	0.5212	307.375
	Fused	0.5023	0.5605	0.5856	0.5350	496.462
	Improve	0.03	2.55	3.14	1.38	
DistMulti	Source	0.3808	0.4773	0.5090	0.4322	307.375
	Fused	0.5008	0.5604	0.5856	0.5332	496.462
	Improve	12.00	8.31	7.66	10.10	
ComplEx	Source	0.4421	0.5481	0.5703	0.4976	307.375
	Fused	0.4959	0.5668	0.5855	0.5323	496.462
	Improve	5.38	1.87	1.52	3.48	

Hit1, Hit3, Hit10, and (MRR) by up to 1.7%, 1.49%, 2.22% and 1.17%, respectively. Another example is the KGs Kegg and Sgd (Scenario B): the improvements are up to 9.34%, 8.94%, 7.05%, and 8.77% for *Hit1, Hit3, Hit10, and (MRR)*, respectively (see Table 7.14). From the results, we can also see that, in some cases, KG fusion lowers the performance of the link prediction task. For example, KG Kegg and KG Drugbank in *Scenario A* using TuckER (see Table 7.8).

However, because of the absence of benchmark KGs for such a task (i.e., the impact of KG fusion on the quality of the link prediction task), it is difficult to say that this improvement is caused by the models used or the KGs. Fundamentally, the performance of TuckER, DistMulti, and ComplEx in the literature is evaluated on benchmark KGs (see [BAH19]). These benchmark KGs are tailored to evaluate KGE. DistMulti is limited to symmetric relations, while ComplEx is able to capture

Tab. 7.12.: Link prediction for DBpedia, LGD, and the fused data in *Scenario A*.

Model	**Data**	**Hit@1**	**Hit@3**	**Hit@10**	**MRR**	**|Data|**
TuckER	Source Fused	0.4504 0.4802	0.5326 0.5598	0.5997 0.6088	0.5033 0.5278	271.706 326.792
	Improve	2.98	2.72	0.91	2.45	
DistMulti	Source Fused	0.3601 0.3772	0.4548 0.4515	0.5249 0.5224	0.4187 0.4277	271.706 326.792
	Improve	1.71	-0.33	-0.25	0.89	
ComplEx	Source Fused	0.3350 0.3725	0.4501 0.4507	0.5127 0.5361	0.4022 0.4275	271.706 326.792
	Improve	3.76	0.06	2.34	2.52	

Tab. 7.13.: Link prediction for Drugbank, Sider, and the fused data in *Scenario B*.

Model	**Data**	**Hit@1**	**Hit@3**	**Hit@10**	**MRR**	**|Data|**
TuckER	Source Fused	0.4210 0.4571	0.4567 0.5248	0.5047 0.5940	0.4485 0.5037	29.02 36.222
	Improve	3.60	6.81	8.94	5.52	
DistMulti	Source Fused	0.3332 0.6943	0.4233 0.8016	0.4854 0.8303	0.3885 0.7485	29.02 36.222
	Improve	36.11	37.82	34.49	36.01	
ComplEx	Source Fused	0.4061 0.4189	0.4842 0.53	0.5380 0.5941	0.4551 0.4846	29.02 36.222
	Improve	1.28	4.58	5.61	2.95	

antisymmetric relations. Thus, if a KG contains many antisymmetric relations, DistMulti may perform poorly. ComplEx can handle antisymmetric relations, but its parameter number grows quadratically with the number of relations, which frequently leads to overfitting, especially for connections with a limited number of training triples. TuckER manages this problem by modeling relations as vectors w_r, so the number of parameters scales linearly with the number of relations. Another reason that can affect the results and the performance is the selection of hyperparameters. In our current experiments, we did not perform any sort of hyperparameter optimizations. We use hyperparameters from [BAH19]. Based on these observations, the impact of KG fusion on the quality of the link prediction task is still an open question and needs a thorough investigation, from benchmarking KGs to hyperparameter optimization.

Tab. 7.14.: Link prediction for Kegg, Sgd, and the fused data in *Scenario B*.

Model	Data	Hit@1	Hit@3	Hit@10	MRR	|Data|
TuckER	Source	0.4445	0.4840	0.5366	0.4747	23.712
	Fused	0.5379	0.5733	0.6071	0.5624	31.337
	Improve	9.34	8.94	7.05	8.77	
DistMulti	Source	0.3123	0.3792	0.4283	0.3549	23.712
	Fused	0.3998	0.4736	0.5373	0.4482	31.337
	Improve	8.75	9.44	10.90	9.33	
ComplEx	Source	0.3687	0.5128	0.5614	0.4489	23.712
	Fused	0.4580	0.5598	0.5874	0.5150	31.337
	Improve	8.93	4.70	2.60	6.61	

Tab. 7.15.: Link prediction for Drugbank, Omim, and the fused data in *Scenario B*.

Model	Data	Hit@1	Hit@3	Hit@10	MRR	|Data|
TuckER	Source	0.4158	0.4537	0.4936	0.4428	46.981
	Fused	0.4328	0.4686	0.5157	0.4607	49.308
	Improve	1.70	1.49	2.22	1.79	
DistMulti	Source	0.3169	0.3906	0.4627	0.3671	46.981
	Fused	0.3297	0.3930	0.4585	0.3737	49.308
	Improve	1.28	0.24	-0.42	0.66	
ComplEx	Source	0.3921	0.4644	0.5054	0.4363	46.981
	Fused	0.4038	0.4765	0.5188	0.4476	49.308
	Improve	1.17	1.21	1.34	1.14	

Tab. 7.16.: Link prediction for Kegg, Omim, and the fused data in *Scenario B*.

Model	Data	Hit@1	Hit@3	Hit@10	MRR	|Data|
TuckER	Source	0.4085	0.4397	0.4642	0.4285	591.5
	Fused	0.4410	0.4856	0.5133	0.4674	690.224
	Improve	3.26	4.59	4.92	3.89	
DistMulti	Source	0.2883	0.3284	0.3483	0.3116	591.5
	Fused	0.3480	0.4091	0.4452	0.3839	690.224
	Improve	5.97	8.07	9.68	7.23	
ComplEx	Source	0.3760	0.4555	0.4766	0.4192	591.5
	Fused	0.4243	0.4950	0.5261	0.4651	690.224
	Improve	4.83	3.95	4.95	4.59	

Tab. 7.17.: Link prediction for Kegg, Drugbank, and the fused data in *Scenario B*.

Model	Data	Hit@1	Hit@3	Hit@10	MRR	|Data|
TuckER	Source	0.4451	0.4718	0.4921	0.4636	56.970
	Fused	0.4530	0.4795	0.5046	0.4736	61.332
	Improve	0.79	0.76	1.26	1.00	
DistMulti	Source	0.3146	0.3749	0.4150	0.3526	56.970
	Fused	0.5325	0.5540	0.6122	0.5548	61.332
	Improve	21.79	17.90	19.72	20.22	
ComplEx	Source	0.3591	0.4869	0.5105	0.4245	56.970
	Fused	0.3743	0.4878	0.5115	0.4344	61.332
	Improve	1.52	0.09	0.10	1.00	

Tab. 7.18.: Link prediction for DBpedia, LGD, and the fused data in *Scenario B*.

Model	Data	Hit@1	Hit@3	Hit@10	MRR	|Data|
TuckER	Source	0.3662	0.4313	0.4906	0.4107	511.294
	Fused	0.3728	0.4348	0.4969	0.4161	566.380
	Improve	0.66	0.35	0.63	0.54	
DistMulti	Source	0.2684	0.3645	0.4275	0.3273	511.294
	Fused	0.2637	0.3406	0.4113	0.3152	566.380
	Improve	-0.48	-2.39	-1.62	-1.22	
ComplEx	Source	0.2598	0.3149	0.3924	0.3034	511.294
	Fused	0.2813	0.3349	0.3966	0.3217	566.380
	Improve	2.16	2.00	0.42	1.83	

Part III

Link Discovery Explainability And Human In The Loop

8 Multilingual Verbalization and Summarization for Explainable Link Discovery

Preamble: This chapter is based on the paper by Ahmed et al. [Ahm+21] addressing the explainability of data integration in declarative LD. Declarative LD frameworks rely on complex LS to express the conditions under which two resources should be linked. Understanding such an LS is not a trivial task for non-expert users, particularly when such users are interested in generating LS to match their needs. Even if the user applies a machine learning algorithm for the automatic generation of the required LS, the challenge of explaining the resultant LS persists. Hence, providing explainable LS is the key challenge in enabling users who are unfamiliar with the underlying LS technologies to use them effectively and efficiently. In this paper, we extend our previous work [ASN19a] by proposing a generic multilingual approach that allows the verbalization of LS in many languages, i.e., converts LS into understandable natural language text. In this work, we ported our LS verbalization framework to German and Spanish, in addition to the English language. Our adequacy and fluency evaluations show that our approach can generate complete and easily understandable natural language descriptions even by lay users. Moreover, we devised an experimental neural approach to improve the quality of our generated texts. Our neural approach achieves promising results in terms of BLEU, METEOR and chrF++.

8.1 Motivation

With the rapid increase in the number and size of RDF datasets comes the need to link such datasets. Declarative LD frameworks rely on complex LS to express the conditions necessary for linking resources within these datasets. For example, state-of-the-art LD frameworks such as Limes [Ngo11] and Silk [IJB11] adopt a property-based computation of links between entities. To configure LD frameworks, the user can either (1) manually enter an LS or (2) use machine learning for automatic LS generation.

There are a number of machine learning algorithms that can find LS automatically, using either supervised, unsupervised or active learning. For example, the EAGLE algorithm [NL12] is a supervised machine-learning algorithm able to learn LS using genetic programming. In newer work, the WOMBAT algorithm [SNL17] implements a positive-only learning algorithm for automatic LS finding based on generalization via an upward refinement operator. While LD experts can easily understand the generated LS from such algorithms, and even modify if necessary, most lay users lack the expertise to proficiently interpret those LSs. In addition, so far, these algorithms have been unable to explain the LS they generate to lay users. Consequently, these users will face difficulty when they i) assess the correctness of the generated LS, ii) adapt their LS, or iii) choose in an informed manner between possible interpretations of their input. In this chapter, we address the readability of LS in terms of natural language. To the best of our knowledge, this is the first work that shows how to verbalize LS and targets many languages, such as English, German, and Spanish. As a result, our framework will help people who are unfamiliar with the underlying technology of LS to understand and update it efficiently. The contributions of this paper are as follows:

1. We propose the first (to the best of our knowledge) multilingual *template-based* approach to produce natural text from LS. Our approach is motivated by the pipeline architecture for NLG systems performed by systems such as those introduced by Reiter & Dale [RD00]. Our evaluations show that our LS verbalization template-based approach can generate complete and easily understandable natural language descriptions even by lay users.

2. We propose a first version of our *neural-based* LS verbalization approach trained by automatically generated verbalization from our template-based LS approach. We also identify the challenges that the research community needs to address to carry out this task.

3. Finally, we propose a multilingual selectivity-based approach to generate a summarized verbalization of LS.

The rest of this chapter is structured as follows: First, we introduce an overview of our template-based approach underlying LS verbalization in Section 8.2. In Sections 8.2.1 and 8.2.2, we explain *document-planner* and the tasks carried out in the *micro-planner*, respectively. In Sections 8.2.3 and 8.2.4, we explain the German and Spanish verbalization and the modification carried out. In Section 8.3, we

```
OR(jaccard(x.name,y.name)|0.42,trigrams(x.name,y.description)|0.61)
```

Listing 1: Running example.

introduce our neural-based LS verbalization approach. Subsequently, we introduce our summarization approach in Section 8.4. We then evaluate our approach with respect to the *adequacy* and *fluency* [Dod02] of the natural language representations it generates in Section 8.5.

Throughout the rest of the chapter, we use the LS shown in Listing 1 as our running example. It is generated by the WOMBAT [SNL17] algorithm to link the ABT-BUY benchmark dataset from [KTR09], where the source resource x will be linked to the target resource y if our running example's LS holds.

8.2 Template-Based LS Verbalization Approach

Our goal here is to generate a complete and correct natural language representation of an arbitrary LS. Our template-based LS verbalization approach is motivated by the pipeline architecture for NLG systems, as introduced by Reiter & Dale [RD00]. The NLG architecture consists of three main stages: *document-planner*, *micro-planner* and *surface realizer*. Since this work is the first step towards the verbalization of LS, our efforts will be focused on *document-planner* (as explained in Section 8.2.1) with an overview of the tasks carried out in the *micro-planner* (Section 8.2.2). The *surface realizer* is used to create the output text.

8.2.1 Documentplanner

The *document-planner* consists of the content determination process to create messages and the document structuring process that combines those messages. We focus on document structuring to create independently verbalizable messages from the input LS and to decide on their order and structure. These messages are used for representing information. This part is carried out in the preprocessing and processing steps.

Preprocessing

The goal of the preprocessing step is to extract the central information of LS. This step mainly relies on the *atomic LS*, where the necessary information can be extracted. The input for this step is the atomic LS, while the output is the realization of each individual part of it. To this end, we break down the atomic LS into its individual parts:

- p_s property of the source resources $s \in S$,
- p_t property of the target resources $t \in T$,
- similarity threshold θ and
- similarity measure m.

The first property p_s comes from dataset S (source), and the second property p_t comes from dataset T (target). For example, $p_s = label$ while $p_t = name$ that means we have a pair of properties $(p_s, p_t) = (label, name)$. After that, on each part of the *atomic LS*, we apply the dependency rule introduced in Table 1. We start with the realization of similarity measure m (e.g. $jaccard$ as stated in our running example in Listing 1) as follows.

1. ζ(m) $\Rightarrow$ nn(m,similarity)

Now we can combine ζ(m) and $\zeta(\theta)$.

2. $\zeta(m,\theta)$ = ζ(m) $\wedge$ $\zeta(\theta)$ $\Rightarrow$ prep_of($\zeta(\theta)$,$\zeta(m)$)

Furthermore, if θ equals 1, we replace its value with *"exact match"*, and in cases where θ is equal to 0, we replace it with *"complete mismatch"*. Otherwise, we keep the θ value (e.g., in the case of our running example). Regarding the properties p_s and p_t, we move the explanation into the processing step since they play an important role in the construction of a *subject* to be used later in sentence building.

Processing

In this step, we aim to map all *atoms* z into their realization function ζ(z) and to define how these atomic realizations are to be combined. The input for this step is the LS and the output is the verbalization of the LS at hand. Given our formalization of LS in Section 2.4.1, any LS is a binary tree, where the root of the tree is an operator op and each of its two branches are LSs. Therefore, we recursively and in order apply our processing step to the LS tree at hand. As the complete verbalization of an atomic LS mainly depends on the properties p_s and p_t, we distinguish two cases here: the *first case* where p_s and p_t are equal, so we only need to verbalize p_s. In this case, the realization function of an atomic LS $a \in \mathcal{A}$ is constructed as follows.

3. ζ`(a)`$\Rightarrow$`subj(have,nn(prep_of(`ζ`(`p_s`), `ζ`(source and target)),` ζ`(resources)))` $\wedge$ `dobj(have,`ζ`(`m`,`θ`))`

The *second case* is where p_s and p_t are not equal. Here, both properties need to be verbalized as follows.

4. ζ`(`p_s`,`p_t`)`$\Rightarrow$ ζ`(`p_s`)` $\wedge$ ζ`(`p_t`)`

8.2.2 Microplaner

The micro-planner is divided into three processes: *lexicalization, referring expression generation*, and *aggregation*. We explain each process in the following.

Lexicalization

Within the lexicalization process, we decide what specific words should be used to express the content. In particular, we choose the actual nouns, verbs, adjectives, and adverbs to appear in the text from a lexicon. Also, we decide which particular syntactic structures to use, for example, whether to use the phrase `the name of the resource` or **resource's name**.

5. ζ`(`p_s`)`$\Rightarrow$ `prep_of( poss(`ζ`(resource), `p_s`),`ζ`(source))`

6. ζ`(`p_t`)`$\Rightarrow$ `prep_of( poss(`ζ`(resource), `p_t`),`ζ`(target))`

7. ζ`(a)`$\Rightarrow$ `subj(have,`ζ`(`p_s`,`p_t`))` $\wedge$ `dobj(have,`ζ`(`m`,`θ`))`

Applying *preprocessing* and *processing* steps followed by the *Lexicalization* step on our running example from Listing 1 generates the following verbalization: `The name of source and target resources has a 42% of Jaccard similarity or the resource's name of the source and the resource's description of the target have a 61% of Trigrams similarity.` Note that our running example contains both cases.

```
OR(jaccard(x.name,y.name)|0.42,qgrams(x.name,y.name)|0.61)
```

Listing 2: Grouping example.

Referring Expression Generation

Here, we carry out the task of deciding which expressions should be used to refer to entities. Considering the example, the source and the target have a resource's name and they have a 45% of Jaccard similarity, they is referring to the expression the source and the target. However, we avoid such a construction in our verbalization because we aim to generate a simple yet readable text that contains the central information of the LS at hand.

Aggregation

The goal of aggregation in NLG is to avoid duplicating information that has already been presented. In our LS verbalization, we mainly focus on the *subject collapsing*, defined in [DH96] as the process of *"collecting clauses with common elements and then collapsing the common elements"*. Formally, we define *subject* subj(v_i, s_i) as s_i, *object* dobj(v_i, o_i) as o_i

8. ζ(s_1)= ζ(s_2)=...=ζ(s_n)$\Rightarrow$ subj(v_1,s_1) $\wedge$ dobj(v_1, coord(o_1,o_2,...,o_n))

In Listing 2, we present a second example LS, where grouping is applicable.

The original verbalization of LS from Listing 2 is: The name of source and target resources has a 42% of Jaccard similarity or the name of source and target resources has a 61% of Qgrams similarity. And after applying grouping, our verbalization will become more compact as follows: The name of source and target resources has a 42% of Jaccard similarity or a 61% of Qgrams similarity.

8.2.3 German Verbalization

Regarding the German language, we make use of the genitive case instead of using the possessive case since it is widely used in the German language. For instance, the words "Name" and "Datenquelle" will be verbalized into "Namens der Datenquelle". We generated German text using the implementation of [Bra+19].

Considering the complexity of the German language, our running example from Listing 1 will be converted into "Name von der Datenquelle und dem Datenziel Ressourcen hat 42% einer Jaccards Ähnlichkeit oder die Ressource Namens der Datenquelle und die Ressource Descriptions des Datenziels haben 61% einer Trigramss Ähnlichkeit". The same LS can be verblized in German as follows: "der Link passiert, wenn Name von der Datenquelle und dem Datenziel Ressourcen 42% einer Jaccards Ähnlichkeit hat oder der Link passiert, wenn die Ressource Namens der Datenquelle und die Ressource Descriptions des Datenziels 61% einer Trigramss Ähnlichkeit haben". This is due to the fact that there are many possible word orders for the same sentence in German. However, the position of the verb should be changed according to the way of ordering the words. For instance, in the first verbalization, the verb "hat" is in the middle of the sentence, while in the second verbalization, after changing the order of words, the verb "hat" moved to the end of the sentence. From a surface realization perspective, many inflection rules make the German language more complex than English. While table look-ups for inflected forms can be performed reasonably in the English language, it is not feasible for German. The German language has "ein" and "eine" as indefinite articles and "das", "der", and "die" as definite articles, whereas in the English language, the as definite article and a/an as indefinite articles satisfy. Accordingly, all articles and pronouns must be inflected according to gender, number, person and grammatical case (nominative, genitive, dative, accusative), resulting in more article forms, for instance for indefinite articles in "einen", "einem", "einer", and "eines". For example, "die Resource Descriptions des Datenziels 61% einer Trigramss Ähnlichkeit", where the indefinite article "eine" changed to "einer" obeying the grammar of genitive case. This can be seen for definite articles as well. For example, "die Resource Descriptions des Datenziels" is an inflection for the noun "das Datenziel" in the genitive case (i.e. "... des Datenziels")

8.2.4 Spanish Verbalization

For the verbalization in the Spanish language, we rely on the implementation proposed in [RJB17], in which the authors extended the bilingual English-French SimpleNLG-EnFr realizer. The Spanish realizer is named SimpleNLG-ES. In SimpleNLG-EnFr, English and French share most of the basic framework, such as document elements and some grammar rules which are common for both English and Spanish. The Spanish language is not rich in terms of morphology rules compared to other close languages. There are two main contractions between the prepositions "a" and `"de"`, and the masculine singular determinant `"el"` that are always used as follows: `"a el"` → `"al"` meaning `to the`, and the second case is `"de el"` → `"del"` meaning `of the`. As a result, SimpleNLG-ES provides only support for `"al"` and `"del"` .

The verbalization in Spanish was adapted in a straightforward manner. In our running example in Listing 1, the Spanish verbalization was generated as follows: `"El enlace será generado se La propiedad "name" de la fuente y los recursos de destino tiene un 42% de Jaccard similitud o el recurso name de la fuente y el recurso description de destino tiene un 61% de Trigrams similitud"`.

8.3 Neural-Based LS Verbalization Approach

To improve our verbalization module quality, we devised a neural approach that relies on standard sequence-to-sequence [BCB15] models for generating text. It is well known that neural models require a considerable amount of training data to achieve good performance [ASN19a]. However, to the best of our knowledge, our work is the pioneer in generating verbalization, i.e., explanations in natural language from link specifications data. Therefore, there is a lack of training data for this task. Thus, our neural approach's intuition was to train the neural network on the texts generated by our template-based approach (Section 8.2). Using our template-based approach, we create parallel training data, the source is composed of the link specifications, and the target refers to the natural language sentences. Every link specification is aligned to one natural language sentence. We generated $35,000$ parallel data. Our neural architecture is a bidirectional RNN-LSTM 2-layer encoder-decoder model with an attention mechanism [Mou+19]. The training uses a batch size of 32 and the stochastic gradient descent with an initial learning rate of 0.0002. We set a source and target word embeddings size of 500 and hidden layers to size 500, dropout $= 0.3$ (naïve). We used a maximum sentence length of 80, a vocabulary of $50,000$ words, and a beam size of 5.

8.4 Selectivity-Based LS Summarization Approach

We define the *selectivity score* of a sub-LS $L_s \in L$ as a function $\sigma(L)$ that returns the F-measure achieved by the mapping $[[L_s]]$ of L_s by considering the mapping $[[L]]$ generated by the original LS L as its reference mapping.

We propose a sentence-scoring-based LS summarization approach. The basic idea behind our summarization approach is to simplify the original LS tree by pruning LS sub-trees that achieve the minimum *selectivity score*, i.e., keep the information loss minimum. Given an input LS L_i, our summarization approach first generates an ordered list $\mathbf{L}$ of simplified LSsof L_i, where $\mathbf{L}$ is ordered by the selective score of each of its elements in descending order. This step is carried out by iteratively pruning the sub-tree of L_i with the minimum selectivity score.

In cases where a summarization threshold $\tau \in [0, 1]$ is given, the output of our summarization algorithm will be generated by applying our LS verbalization approach to the LS $L \in \mathbf{L}$ with the highest selectivity score $\sigma(L) \leq \tau$. Otherwise, the output of our summarization approach will be a list of the verbalization of the whole list $\mathbf{L}$. For instance, assume we have the LS `OR(jaccard(x.name,y.name)|0.45, qgrams(x.name,y.name)|0.67)`, from which our summarization approach generates the ordered list $\mathbf{L} = \{L_1, L_2\}$, where $L_1 =$ `(jaccard(x.name,y.name)|0.45)` and $L_2 =$ `(qgrams(x.name, y.name)|0.67)`. Accordingly, we compute the scores of $\sigma(L_1) = 0.8$ and $\sigma(L_2) = 0.6$. Assume we have the summarization threshold $\tau = 0.9$. Our approach will thus verbalize the link specification L_1 with the highest selectivity score $\sigma(L) \leq \tau$.

8.5 Evaluation

We evaluated our approaches for LS verbalization and summarization in order to elucidate the following questions:

Q_1: Does the LS verbalization help the user to better understand the conditions sufficient to link the resources in comparison to the original LS?

Q_2: How fluent is the generated LS verbalization, i.e., how good is the natural language description of the LS verbalization in terms of comprehensibility and readability?

Q_3: How adequate is the generated LS verbalization? That is, how well does the verbalization capture the meaning of the underlying LS?

Q_4: How much information do we lose by applying our summarization approach?

8.5.1 Experimental Setup

To answer the first three questions, we conducted a *user study* to evaluate our LS verbalization. We used our approach to verbalize a set of five LSsautomatically generated by the EAGLE algorithm [NL12] for the benchmark datasets of `Amazon-GP`, `ABT-BUY`, `DBLP-ACM`, and `DBLP-Scholar` from [KTR09]. Our user study consists of four tasks, and each task consists of five multiple choice questions[1]. Altogether, we have a group of 18 participants in our user study from the DICE[2] and AKSW[3] research groups. In the following, we explain each task:

Task 1: This task consists of five identical sub-tasks. For each, we present to the survey participant a LS and three pairs of source and target resources represented by their respective concise bounded descriptions (CBD)[4] graph. These pairs are matched based on the provided LS with different degrees of confidence. To this end, the participant is asked to find the best-matched pair, and we measure the response time for each participant.

Task 2: This task also consists of five identical sub-tasks. We again follow the same process in *Task 1* of presenting the participant with the CBDs of matched resources, but this time, we give the survey participant the verbalization of the LSs. Again, we record the response time of each participant.

Task 3: Within this task, a survey participant is asked to judge the fluency of the provided verbalization. Here, we follow the machine translation standard introduced in [Dod02]. Fluency captures how good the natural language description is, in terms of comprehensibility and readability, according to the following six ratings: (6) Perfectly clear and natural; (5) Sounds a bit artificial, but is clearly comprehensible (may contain minor grammatical flaws); (4) Sounds very artificial, but is understandable (although may contain significant grammatical flaws); (3) Barely comprehensible, but can be understood with some effort; (2) Only a loose and incomplete understanding of the meaning can be obtained, and (1) Completely incomprehensible.

LS They are then asked to judge the adequacy of the verbalization. Here, we follow the machine translation standard from [Dod02]. Adequacy addresses how well the verbalization captures the meaning of the LS, according to the following six ratings: (6) Perfect; (5) Mostly correct, although maybe

[1] The survey interface for English can be accessed at `https://umfragen.uni-paderborn.de/index.php/186916?lang=en`

[2] `https://dice-research.org/`

[3] `http://aksw.org/About.html`

[4] `https://www.w3.org/Submission/CBD/`

some expressions don't match the concepts very well; (4) Close, but some information is missing or incorrect; (3) There is significant information missing or incorrect; (2) Natural Language (NL) description and LS are only loosely connected; and (1) NL description and LS are in no conceivable way related.

To answer the last question, we conducted an experiment on the benchmark datasets from [KTR09]. We ran the supervised version of the WOMBAT algorithm to generate an automatic LS for each dataset. We again used [SNL17] to configure WOMBAT. Afterwards, we applied our summarization algorithm to each of the generated LSs. Because of the space limitation, we present only the verbalization of the original LS (the ones generated by WOMBAT) as well as the first summarization of it for the `Amazon-GP` and `DBLP-Scholar` datasets in Table 8.1. The complete results are available on the project website[5]. To evaluate the verbalization of German, we reformulated the same 4 *Tasks* but the participants were only 8 participants. The survey can be accessed via [6]. To evaluate our verbalization approach for Spanish, we conducted a survey similar to the ones created for English and German; however, we have only two experts in LD who can speak Spanish.

8.5.2 Results and Discussion of English Language Verbalization

After collecting all the responses from our user study, we filtered out those survey participants who were unlikely to have thoroughly executed the survey (i.e., the ones who took notably less time than the average response time of all other participants) or who were likely distracted while executing it (i.e., the ones who took notably more time than the average time of all other participants). This process reduced the number of valid participants to 16. Our final accepted time window was 3.5–38 minutes for *Tasks 1* & *2*. Accordingly, we start our evaluation by comparing the user time required to find the best-matched source-target pair using LS (*Task 1*) against using the verbalization of the provided LS (*Task 2*).

As shown in Figure 8.1, the average user response time with LS verbalization is less than the response times for LS in the 5 LSs in our user study. On average, using verbalization is 36% faster than using LS. Additionally, we also compared the error rates of participants in *Tasks 1* & *2*, i.e. the number of incorrect answers per question. As shown in Figure 8.2, we have a higher error rate using verbalization (5% mean squared error) than when using LS. These results show that using LS verbalization

[5] `https://bit.ly/2XKDpKZ`

[6] The survey for German language verbalization`https://umfragen.uni-paderborn.de/index.php/288119?lang=en`

decreases the average response time, which is an indicator that our participants were able to better understand underlying LS using verbalization. Still, using the LS verbalization does not always lead our participants to select the correct answer. This is due to the complexity involved in the underlying LSs, which leads to verbalization that is too long. This answers Q_1. Using our simplification approach on the same LS verbalization leads our participants to achieve better results.

The results of *Task 3* (see Figure 8.3) show that the majority of the generated verbalizations (i.e., the natural language descriptions) were fluent. In particular, 87% of the cases achieved a rating of 3 or higher. On average, the fluency of the natural language descriptions is 4.7 ± 0.4. This answers Q_2.

For *Task 4*, the average adequacy rating of our verbalization was 4.75 ± 0.6 (see Figure 8.4), which we consider to be a positive result. In particular, 40% of all verbalizations were judged to be perfectly adequate, and 83% of the cases achieved a rating of 3 or higher. This answers Q_3.

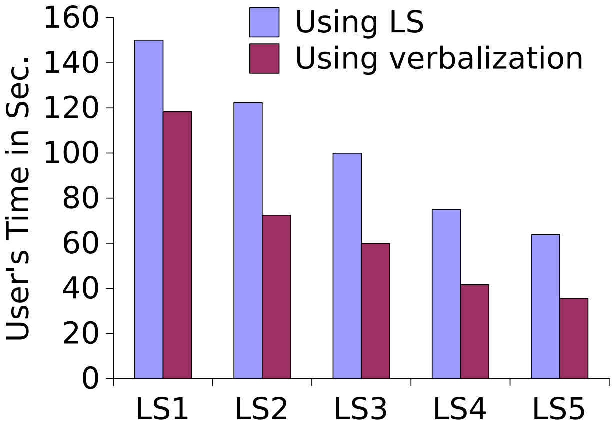

Fig. 8.1.: Average response time of our user study for the English language.

As we can see in Table 8.1, applying our summarization approach reduces the verbalization of the original LS to more than half of its original size. At most, our summarization approach loses an F-Measure of 12% of the original description, which we consider a fair price given the high summarization rate. This clearly answers our last question.

8.5.3 Results And Discussion of German Language Verbalization

As shown in Figure 8.5, the average user response time with LS verbalization is less than the ones for LS in 80% of the LSsused in our users' study. On average, using

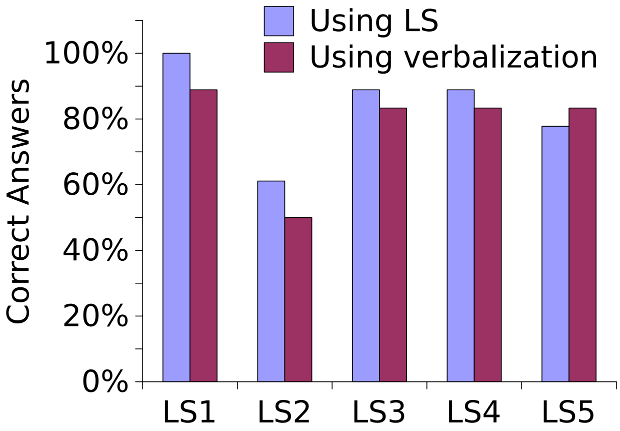

Fig. 8.2.: Correct answers of our user study for the English language.

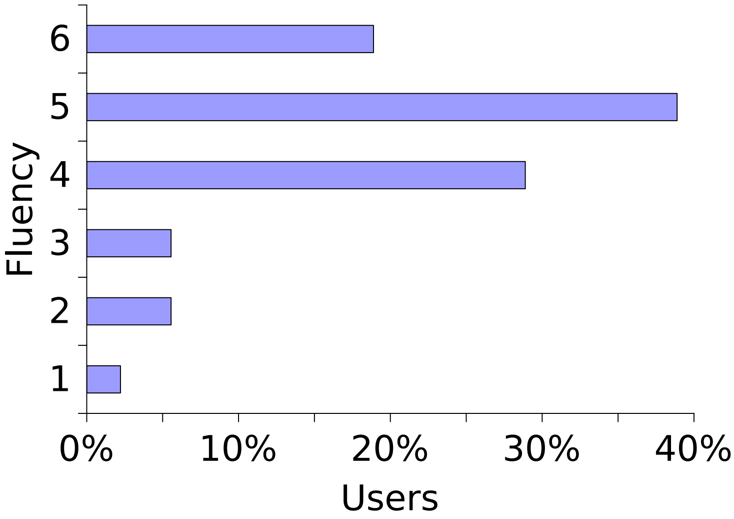

Fig. 8.3.: Fluency results for the English language.

verbalization is 16% faster than using LS. In addition, we measured the error rates of participants in *Tasks 1* & *2*, i.e. the number of incorrect answers per question. Since there are fewer participants, we did not filter out any survey participants, for instance, those who are unlikely to have executed the survey in a thorough manner (i.e., the ones who might take clearly less time than the average response time of all other participants) or who are likely distracted while running it (i.e., the ones who might spend notably more time than the average time of all other participants). As shown in Figure 8.6, using verbalization, we have a higher error rate (1.6% mean squared error) than when using LS. These results indicate that our participants were able to better understand underlying LS using verbalization. However, using the LS verbalization does not always lead our participants to select the correct answer. This is due to the fact that the underlying LSsare very complex, which makes the

Tab. 8.1.: Verbalization of different summarization of an LS for the DBLP-SCHOLAR and Amazon-GP dataset with respective F-measures in the English language.

Dataset	F-measure	Verbalization
DBLP-SCHOLAR	1	The link will be generated if the title of the source and the target resources have a 66% of Cosine similarity
Amazon-GP	1	The link will be generated if the resource's title of the source and the resource's name of the target have a 48% of Cosine similarity or the description of the source and the target resources have a 43% of Cosine similarity or the resource's title of the source and the resource's description of the target have a 43% of Jaccard similarity
Amazon-GP	0.97	The link will be generated if the resource's title of the source and the resource's name of the target have a 48% of Cosine similarity

verbalization too long. This answers Q_1. For *Task 3* (see Figure 8.7), the results show that the bulk of the generated verbalizations (i.e., the natural language text) were fluent. In particular, 90% of the cases achieved a rating of 3 or higher. The average fluency of the natural language descriptions is 4.3 ± 0.06. This answers Q_2. The results of *Task 4* show that the average adequacy rating of our verbalization is 4.8 ± 0.52 (see Figure 8.8), which we recognize as a positive result. Precisely, 42.5% of all verbalizations were judged to be perfectly adequate and 92.5% of the cases achieved a rating of 3 or higher. This answers Q_3. In general, the verbalization for the German language is more difficult compared to English or Spanish because of the natural difficulty of the German language itself; however, our approach is able to generate text with an average adequacy rating of 4.8 ± 0.52 and the average of the fluency is 4.3 ± 0.06.

To answer the last question, Q_4, in Table 8.2, we can see that applying our summarization approach to the German language can reduce the verbalization of the original LS to more than half of its original size. An F-measure of 12% of the original description is the highest loss of our summarization approach. We consider a fair cost given the high summarization rate. Table 8.3 shows the results of summarization applied to the Spanish language. From our summarization results, we can conclude that our summarization approach works independently from the language.

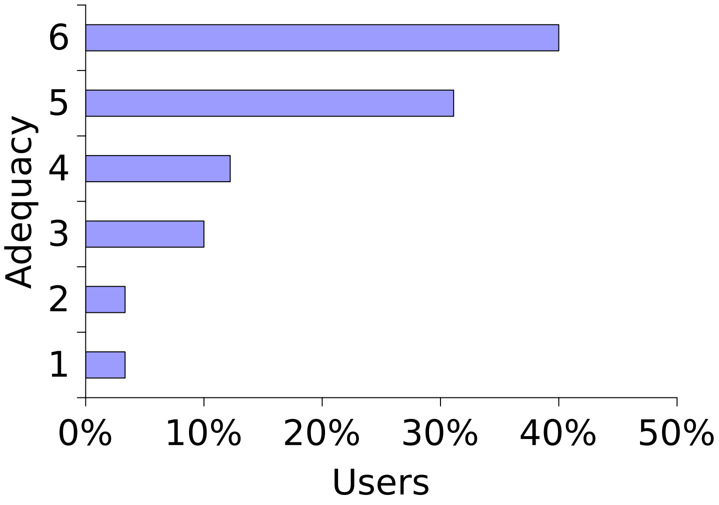

Fig. 8.4.: Adequacy results for the English language.

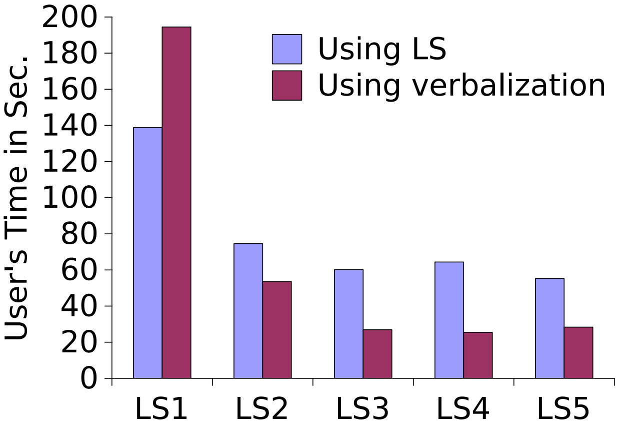

Fig. 8.5.: Average response time of our user study for the German language.

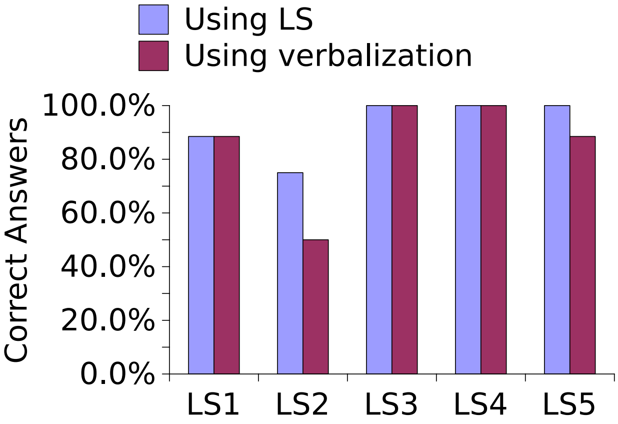

Fig. 8.6.: Correct answers of our user study for the German language.

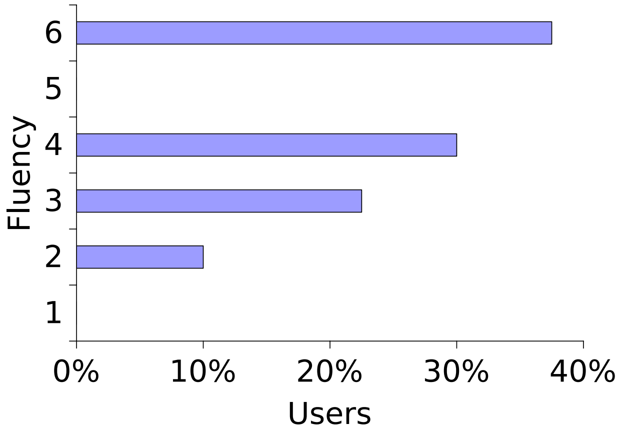

Fig. 8.7.: Fluency results for the German language.

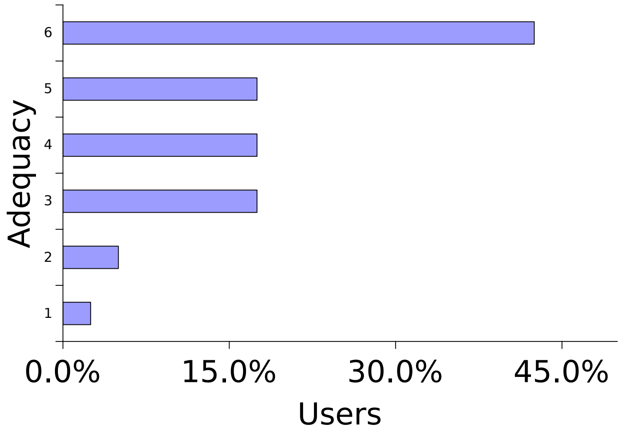

Fig. 8.8.: Adequacy results for the German language.

8.5.4 Results and Discussion of Neural-based Verbalization

As no previous work, to the best of our knowledge, investigated the generation of natural language from link specifications, our goal was to identify the challenges for this new line of research. We reckon that the training data is a *silver standard* because it was not peer-reviewed manually before creating the test data. However, it provided many insights and challenges that the research community needs to address to address this task. Table 8.4 displays the results of our model in the automatic evaluation standard metrics, BLEU, METEOR, and chrf++.

The results of the automatic metrics show that our model was capable of generating fluent texts. However, we looked more in-depth at the generated texts and investigated the sentences manually. We perceived that some metrics, such as `JaroWinkler` and `Ngram` were rarely verbalized, while `Jaccard` and `Levenstein` were frequently

Tab. 8.2.: Summarization of an LS for the DBLP-SCHOLAR and Amazon-GP datasets together with respective F-measures in the German language.

Dataset	F-measure	Verbalization
DBLP-SCHOLAR	1	"Der Link passiert, wenn Title von der Datenquelle und dem Datenziel Ressourcen 66% einer Cosines Ähnlichkeit hat oder der Link passiert, wenn die Ressource Titles der Datenquelle und die Ressource Authorss des Datenziels 43% einer Jaccards Ähnlichkeit haben oder der Link passiert, wenn die Ressource Authorss der Datenquelle und die Ressource Titles des Datenziels 43% einer Trigrams Ähnlichkeit hat"
DBLP-SCHOLAR	0.88	"Der Link passiert, wenn Title von der Datenquelle und dem Datenziel Ressourcen 66% einer Cosines Ähnlichkeit hat"
Amazon-GP	1	Der Link passiert, wenn die Ressource Titles der Datenquelle und die Ressource Namens des Datenziels 48% einer Cosines Ähnlichkeit haben oder der Link passiert, wenn Description von der Datenquelle und dem Datenziel Ressourcen 43% einer Cosines Ähnlichkeit hat oder der Link passiert, wenn die Ressource Titles der Datenquelle und die Ressource Descriptions des Datenziels 43% einer Jaccards Ähnlichkeit hat
Amazon-GP	0.97	Der Link passiert, wenn die Ressource Titles der Datenquelle und die Ressource Namens des Datenziels 48% einer Cosines Ähnlichkeit haben

generated as they appear several times in the training data. This shows the need to generate diverse training data with a balanced distribution of tokens for training the neural models. Additionally, the target side's vocabulary was small, with only 48 different tokens, while the source side, which is regarded as the link specification, contains $10,000$ different tokens. Therefore, we envisage that the text generation for link specification is regarded as a *low-resource translation problem*. Thus, we can apply well-known Machine Translation (MT) strategies for dealing with the data sparsity problem in future work, such as *Byte-per-encoding* (BPE) [SHB16] and *copy-mechanism* [LPM15]. Copy-mechanism tries to substitute the Out-of-vocabulary (OOV) words with target words that have the highest attention weight according to their source words. When the words are not found, it copies the source words to the position of the not-found target word. BPE is a form of data compression

Tab. 8.3.: Summarization of an LS for the DBLP-SCHOLAR and Amazon-GP datasets together with respective F-measures in the Spanish language.

Dataset	F-measure	Verbalization
DBLP-SCHOLAR	1	"El enlace será generado se La propiedad title de la fuente y los recursos de destino tiene un 66% de Cosine similitud o el recurso title de la fuente y el recurso authors de destino tiene un 43% de Jaccard similitud o el recurso authors de la fuente y el recurso title de destino tiene un 43% de Trigram similitud"
DBLP-SCHOLAR	0.88	"El enlace será generado se La propiedad title de la fuente y los recursos de destino tiene un 66% de Cosine similitud"
Amazon-GP	1	El enlace será generado se el recurso title de la fuente y el recurso name de destino tiene un 48% de Cosine similitud o La propiedad description de la fuente y los recursos de destino tiene un 43% de Cosine similitud o el recurso title de la fuente y el recurso description de destino tiene un 43% de Jaccard similitud
Amazon-GP	0.97	El enlace será generado se el recurso title de la fuente y el recurso name de destino tiene un 48% de Cosine similitud

Tab. 8.4.: Results of BLEU, METEOR, and chrf++

Model	BLEU	METEOR	chrf++
LSNeural	32.05	23.41	43.38

that iteratively replaces the most frequent pair of bytes in a sequence with a single, unused byte.

9 Explainable Integration of Knowledge Graphs Using Large Language Models

Preamble: This chapter is based on Ahmed et al. [Ahm+23] that introduced NMV-LS Linked KGs to build the backbone of many data-driven applications, such as search engines, conversational agents, and e-commerce solutions. Declarative LD frameworks use complex LSsto express the conditions under which a link between two resources can be deemed to exist. However, understanding such complex LSsis a challenging task for non-expert users of LD frameworks. In [Ahm+23], we address this drawback by devising NMV-LS, a language model-based verbalization approach for translating complex LSsinto natural language. NMV-LS relies on the results of rule-based link specification verbalization to apply continuous training on T5, an LLM based on the Transformer architecture. We evaluated NMV-LS on English and German datasets using well-known machine translation metrics such as BLUE, METEOR, ChrF++, and TER. Our results suggest that our approach achieves a verbalization performance close to that of humans and outperforms state-of-the-art approaches.

9.1 Motivation

Heterogeneous KGs that obey the principles of linked data are increasing in number. However, relatively few heterogeneous KGs are actually linked. The current LOD statistic[1] shows that there are 1301 KGs having 395.12 billion triples and only 2.72 billion links. Therefore, discovering links among these KGs is a major challenge to achieving the LOD vision[2]. Moreover, the linked KGs build the backbone of various data-driven applications, including information retrieval, recommender systems, search engines, question answering systems, and digital assistants.

[1] Release: 05.05.2021, accessed 24.11.2021 `https://lod-cloud.net/#about`, retrieved using `https://github.com/lod-cloud/lod-cloud-draw/blob/master/scripts/count-data.py`

[2] `https://www.w3.org/DesignIssues/LinkedData.html`

Declarative LD frameworks are used to link entities among KGs. These frameworks use complex LSsto express the conditions required to declare a link between two resources. For example, state-of-the-art LD frameworks such as LIMES [Ngo+21] and SILK [IJB11] adopt a property-based computation of links between entities. For configuring LD frameworks, the user can either (1) manually enter an LS or (2) use machine learning for the automatic generation of LSs. In both cases, a domain expert must manually write LS or set the configuration of machine learning algorithms that are used to generate LS. Furthermore, LD experts can easily understand the LS produced by such algorithms and modify it if needed. However, most lay users lack the experience to proficiently interpret those LSs. Due to this lack of experience, these users have difficulty when they i) check the correctness of the generated LS, ii) customize their LS, or iii) decide between possible interpretations of their input in an informed manner.

The aforementioned challenges can be seen as a bottleneck problem that degrades the effort and potential of ML algorithms to create such LSs automatically. Therefore, the explanation of link discovery-based artificial intelligence has become increasingly popular. For example, we begin our efforts in explaining the LSsin Chapter 8, where we have introduced [ASN19a], a bilingual rule-based approach to verbalize LS, thus addressing the explainability of LD. For example, we begin our efforts to explain the LSsin Chapter 8, where we have introduced a bilingual rule-based approach to verbalize LS, thus addressing the explainability of LD [ASN19a]. Furthermore, in Chapter 8, we extend the previous approach in [ASN19a] and devised a multilingual rule-based approach that includes English, German, and Spanish. We also presented a first attempt to create neural architecture, which is a bidirectional RNN -LSTM two-layer encoder-decoder model with an attention mechanism [MWN18]. However, our neural model did not generalize because the vocabulary was very small and not diverse, Ahmed et al. [Ahm+21].

In this Chapter (Chapter 9), we alleviate the vocabulary problem found in Ahmed et al. [Ahm+21] (see Chapter 8) by proposing a language-based LS approach, named NMV-LS. To this end, we propose a pipeline architecture consisting of two stages. The first stage is a rule-based verbalizer that generates the necessary data to feed the second stage. The second stage relies on a few-shot learning approach by fine-tuning an LLM, in our case T5. The underlying idea of using a language model is to verbalize LS from different types of systems only using a few examples. For example, LSs from LIMES [Ngo+21] differ from the ones used in SILK [IJB11]. Furthermore, the second stage contains a standard two-layer encoder-decoder architecture using different RNN cells such as GRU, LSTM, BiLSTM, and transformer trained with more diverse data. Figure 9.1 shows the proposed architecture.

To evaluate NMV-LS , we designed two settings. In the first setting, we used two datasets to assess the first part of our approach (i.e., standard encoder-decoder architectures). The first dataset contains 107 thousand English pairs, and the second dataset contains 73 thousand German pairs. It should be noted that each pair is nothing but an LS and its verbalization. In the second setting, we used human-annotated data for evaluating the second part of our approach (i.e., few-shot learning using the T5 model). We created human-annotated data from LIMES with only 100 pairs, human-annotated manipulated data from LIMES with only 8 pairs, and human-annotated data from SILK with only 8 pairs. It is important to note that we evaluated our second part only in English.

Our main contributions are as follows.

- We present NMV-LS, a language model-based LS verbalization approach which relies on a few-shot learning strategy.
- We propose an approach that is capable of verbalizing different types of LS, thus mitigating the high efforts to create linguistic rules for each system.
- We propose an approach that is easily extensible to other languages.

The rest of this chapter is structured as follows. First, we give an overview of our approach underlying neural machine verbalization LS in Section 9.2, followed by the evaluation of our approach with respect to the automatic evaluation standard metrics BLEU, METEOR, ChrF++, and TER. We used BENG [Mou+20] to automatically measure the performance of our approach in Section 9.3.

9.2 Approach

NMV-LS consists of two stages. The first stage is the rule-based verbalizer introduced in [Ahm+21] to generate silver data for the second stage, with blue background in Figure 9.1. The second stage is shown with a green background in Figure 9.1. The second stage contains two independent parts. The first part of Stage 2 is based on standard encoder-decoder architectures, such as two layers seq2seq with GRU, LSTM and BiLSTM, and transformer. The second part of stage 2 applies the concept of few-shot learning and is based on the T5 model. In Figure 9.1, ① means that the data is from LIMES silver data, ② means that the training data are a combination of LIMES silver data and human-annotated LIMES silver data, ③ is a combination of ② and human-annotated manipulated LIMES LS, and ④ is a combination of ③ and human-annotated SILK LS. In ②, the human annotation is applied only to the verbalization of LS without changing LS. Manipulated LIMES LS means that we altered the structure of LIMES LS. Listing 1 shows an example of LIMES silver data, Listing 2 is an example of LIMES human-annotated data, Listing 3 is an example of LIMES human-annotated manipulated data, and Listing 4 is an example of SILK human-annotated data.

```
Source =
OR(mongeElkan(x.title,y.title)|0.45,
   cosine(x.title,y.streetName)|0.37)

Target =
A link will be generated if
- the titles of the source and the target have a Mongeelkan similarity of
    45% or
- the title of the source and the streetName of the target have a Cosine
    similarity of 37%
```

Listing 1: LIMES silver data: A pair that contains an LS and its verbalization in English.

```
Source=
AND(AND(ratcliff(x.givenName,y.givenName)|0.0,AND(OR(jaroWinkler(x.
    givenName,y.authors)|0.37,cosine(x.givenName,y.givenName)|0.0)|0.0,
    ratcliff(x.givenName,y.givenName)|0.37)|0.37)|0.0,jaroWinkler(x.
    givenName,y.givenName)|0.37)

Target= a link will be produced supposing that the givenNames of the source
     and the target have a Ratcliff similarity of 0% or the givenName of
    the source and the author of the target have a Jarowinkler similarity
    of 37% or the givenNames of the source and the target have a Cosine
    similarity of 0% and a Ratcliff similarity and a Jarowinkler similarity
     of a 37%
```

Listing 2: LIMES human-annotated data: A pair that contains an LS and its verbalization in English.

9.2.1 Rule-Based Verbalizer

The rule-based verbalizer in [Ahm+21] is based on Reiter & Dale NLG architecture [RD00]. In [Ahm+21], real datasets (knowledge graphs) are used to generate LSs using WOMBAT [SNL17]. Since the number of properties used in [Ahm+21] is limited, it results in less diverse LSs. Our goal is to add more proprieties into each generated LSs. Therefore, in this work, we create ten templates to generate LSsrelying on the rules defined in WOMBAT. The complexity of a LS is formally defined as the number of combined atomic LS, so that an LS is more complex when it contains a higher number of combined atomic LS. For example, the template $(A_1 \sqcup A_2) \sqcap (A_3 \sqcup A_4)$ is less complex than $(A_1 \sqcup A_2) \sqcap (A_3 \sqcup A_4) \sqcap (A_5 \sqcup A_6)$, where A_i is atomic LS.

```
Source=
trigrams(x.givenName, y.name)|0.8 AND cosine(x.title, y.label)|0.7 Or
    levenshtein(x.streetAdress, y.locationAdress)|0.9

Target=
The link will be generated when the givenname of the source and the name of
     the target have a trigrams similarity of 80\% and the title of the
    source and the label of the target have a cosine similarity of 70\% or
    the streetAdressenname of the source and the locationAdress of the
    target have a levenshtein similarity of 90%
```

Listing 3: LIMES human-annotated manipulated data: A pair that contains an LS and its verbalization in English.

```
Source=
 min( mongeelkanSimilarity(?x/p:producer, ?y/p:producer), ratclifDisitance(
    x/p:city,y/p:city))

 Target=
 The link will be generated if the labels of the source and the target have
     minimum mongeelkan similarity or the cities of the source and the
    target have minimum ratclif distance
```

Listing 4: SILK human-annotated data: A pair that contains an LS and its verbalization in English.

9.2.2 Standard Encoder-Decoder Architectures

As we can see in Figure 9.1, the first part of the second stage in our approach deploys a set of standard encoder-decoder architectures. Our first part of the second stage is motivated by the advance in sequence-to-sequence neural translation, which belongs to a family of encoder-decoder architectures [SVL14a]. The encoder neural network reads and encodes a source sentence into a vector. The decoder translates the encoded vectors into a sequence of symbols, i.e., words. The goal here is to maximize the probability of a correct translation by jointly training the entire encoder-decoder system using source sentences. We rely on an RNN for both encoding and decoding [Cho+14b], with the attention mechanism introduced in [BCB15]. We deploy RNN-GRU -2 layer and RNN -Bi/LSTM-2 layer architectures to perform the verbalization. The first architecture is based on LSTM [HS97], while the second architecture is based on GRU [Cho+14b]. Given a sequence of tokens (i.e., words) $\mathbf{x} = (x_1, \cdots, x_T)$ as input in the time step t and a sequence of tokens $\mathbf{y} = (y_1, \ldots, y_T)$ as output, our encoder-decoder is jointly trained to maximize the probability of a correct verbalization. Where $\mathbf{x}$ is the representation of LS and $\mathbf{y}$ is the representation of natural text verbalized by a trained decoder. The length of $\mathbf{x}$ may differ from the length of $\mathbf{y}$. For our proposed NMV-LS (i.e., part one of stage two), we use additive attention (as in [BCB15]) with the conditional probability

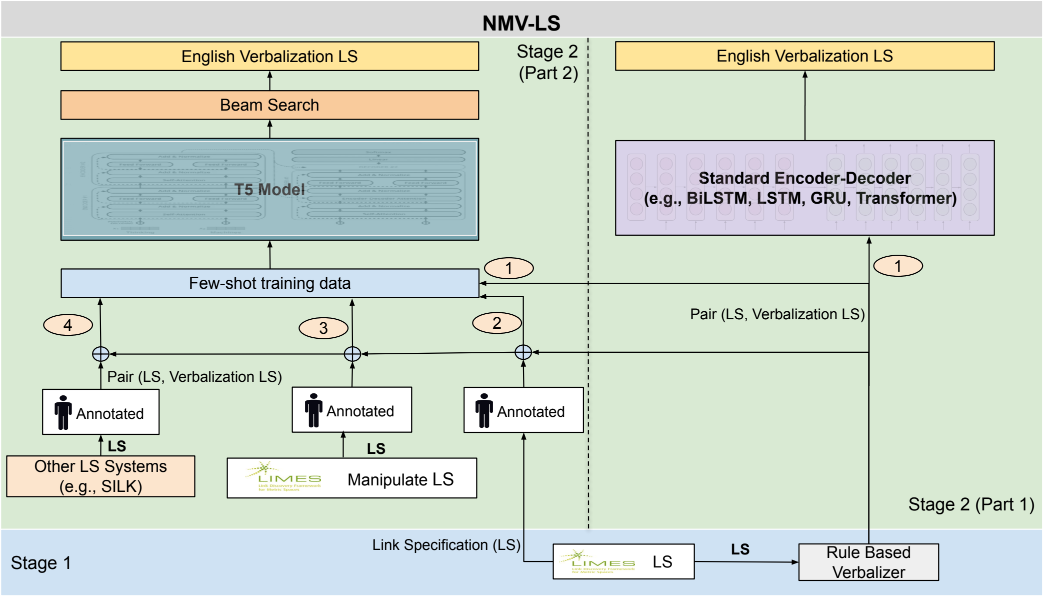

Fig. 9.1.: LS Neural Machine Verbalization System.

$p(y_i|y_1,\ldots,y_{i-1},\mathbf{x}) = g(y_{i-1}, s_i, \mathbf{c}_i)$, where s_i is an RNN decoder's hidden state at time i. Formally, $s_i = f(s_{i-1}, y_{i-1}, \mathbf{c}_i)$ (see [Cho+14b] for more details). In this part of our approach, we also deploy a transformer. Transformer is a sequence-to-sequence architecture that relies entirely on the attention mechanism to transform one sequence into another with the help of a two-part encoder and decoder without implying any recurrent networks (GRU, LSTM, etc). The architecture consists of multiple identical encoders and decoders stacked on top of each other (more details in [Vas+17]). We use our rule-based verbalizer to generate silver data to train our models. However, before feeding these data, we need to apply some preprocessing techniques.

9.2.3 Few-shot Learning Using the T5 model

As depicted in Figure 9.1, the second part of the second stage in our approach is based on a few-shot learning strategy that involves fine-tuning an LLM.

To address the vocabulary issue in Ahmed et al. [Ahm+21], we base our approach on a few-shot learning approach by fine-tuning an LLM such as the T5 model [Raf+19]. T5 is a pre-trained model for text-to-text generative multitasking based on transformer encoder-decoder and is pre-trained on a large pre-training dataset (C4) [Raf+19]. Using the T5 pre-trained model allows the model to learn the structure and pattern of natural language from a vast quantity of diverse, real-world text data. This can assist the model in learning to comprehend and generate high-quality human-like text, which is useful for a variety of natural language processing tasks. In addition, the T5 pre-trained model can frequently be fine-tuned for specific tasks, particularly in our case, to learn the complexity of LS and generate verbalizations of LS with additional data and training time.

To use the T5 pre-trained model for few-shot learning in our model, as shown in Figure 9.1, we fine-tune it on four different small training datasets, as detailed in Section 9.3.3, where those datasets were designed based on the LSsof LIMES. The goal of the model is to generalize the verbalization of a wide range of LSs. Given a sequence of LS tokens as input represented by LS $= \{w_1, w_2, ..., w_N\}$ and mapped into sequence embeddings before being fed into the encoder of T5, which outputs a sequence of embedding vectors. Furthermore, the T5 decoder accepts as input both encoder outputs and previously generated tokens from the decoder during the auto-regressive decoding. Moreover, linear transformation and softmax functions are applied to the decoder outputs. In addition, beam search decoding [SVL14b] is utilized to generate the verbalization LS from the model outputs.

9.3 Evaluation

9.3.1 Data

Since there are no gold standard datasets for an NM verbalization task to translate link specification into natural languages, we generated silver standard datasets using the rule-based approach introduced in [ASN19a] and [Ahm+21]. To evaluate the standard encoder-decoder architecture, we generated three datasets with the following sizes: 107k pairs (English) and 73K pairs (German). Table 9.1 shows the statistical information on the data. To evaluate the fine-tuning of T5, we combined 10k pairs (English) from 107k pairs (English) with 100 pairs of human-annotated data from LIMES, 8 pairs of human-annotated manipulated data from LIMES, and only 8 pairs of human-annotated data from SILK.

9.3.2 Evaluation Metrics

To ensure consistent and clear evaluation, we evaluate our approach with respect to the automatic evaluation standard metrics BLEU [Pap+02], METEOR [BL05], Tab. 9.1.: Statistics about our datasets used in the experiments, where V_{max} is the maximum verbalization length (in words) and V is the verbalization length.

Data	V_{max}	# Records	$V < 50$	$51 < V < 100$	$V > 100$
EN	107	107424	3744 (3.49%)	88320 (82.22%)	15360 (14.30%)
DE	125	73008	3888 (5.33%)	48384 (66.27%)	20736 (28.40%)

ChrF++ [Pop17] and TER [Sno+06]. BLEU [Pap+02] is a popular N-gram-based metric that uses a modified precision metric to compare the machine translation (MT) output with the reference translation. The precision is computed by measuring the similarity of N-gram ($n = 1, \ldots, 4$) at the word level. METEOR [BL05] is another metric based on N-grams and is dependent on semantic features to overcome the semantic weakness of BLEU by considering the synonym overlap through a shared WordNet synset of words. These semantic features improve the quality of the correlation of the output and the reference of the system. Along with exact standard word (or phrase) matching, METEOR has additional features, i.e., stemming and paraphrasing. ChrF++ [Pop17] shows the use of character n-gram precision and recall (F-score) for automatic evaluation of MT outputs. ChrF++ has shown a good correlation with human rankings of different MT outputs. Moreover, ChrF++ is language and tokenization independent. TER [Sno+06] measures the minimum number of edits required to change a system output so that it matches exactly a reference translation. The edits include deletions, insertions, substitutions, and shifts of words, as well as capitalization and punctuation. TER score is calculated by dividing the number of edits by the average referenced words. We use BENG [Mou+20] to automatically evaluate our approach. BENG is an evaluation tool for text generation that abides by the FAIR principles and is built upon the successful benchmarking platform GERBIL [Usb+15].

9.3.3 Experimental Setup

As we can see in Figure 9.1, our approach consists of two stages. The first stage is the rule-based verbalizer and the second stage contains two parts. The first part is based on standard encoder-decoder architectures and the second part is based on the few-shot learning method by fine-tuning an LLM such as T5. However, the first stage feeds the two parts of the second stage. For example, ① means that the data is from the LIMES silver data generated by the first stage of NMV-LS, which is the rule-based verbalizer. ① feeds both two parts of the second stage of our pipeline architecture. For evaluating the first part of the second stage in our approach (i.e., standard encoder-decoder architectures), we conducted three sets of experiments for both English and German to answer the following research question:

Q_1. Does the complexity of LS impact the performance of our NMV-LS in case of training standard encoder-decoder architectures?

For evaluating our second part of the second stage in our approach (i.e., few-shot learning using the T5 model), we conducted one set of experiments for English to answer the following research questions:

Q_2. Does fine-tuning an LLM improve the verbalization of our NMV-LS system?

Q_3. Does fine-tuning a LLM help to generalize different LSsfor verbalization?

Q_4 How large is the impact of using human-annotated data on the quality of verbalization compared to using silver data?

Experiment Set 1, English Language (107k dataset). We evaluated a GRU /LSTM/-BiLSTM-2 layers encoder-decoder on an English dataset consisting of 107k pairs (each pair contains an LS and its verbalization in English or German), split into 70% for training, 20% for validation, and 10% for testing. For all experiments, we set the parameters as follows: The learning rate is {0.1, 0.01, and 1}, the dropout is 0.1, the embedding dimensionality is 256, the epochs number is {100, 1000, and 10000}, the clipping value is 0.25, the SGD optimizer with negative log-likelihood loss function, and the maximum length of a sentence is {107 and 187 tokens}. The max length of a sentence means that model can filter out all the pairs that have a length greater than the max length. For LSTM/BILSTM, the batch size is 256. The selection of parameters is manually adjusted. We run all GRU on colab and LSTM/BiLSTM on a local server with 1 GPU, 16 CPUs, and 32 GB of memory. We use the `Pytorch` library to implement our model. The results are listed in Table 9.2. For these results, we set the learning rate to 0.01 in case of using GRU and to 0.1 in the case of using LSTM/BiLSTM. We conducted additional experiments with the learning rate set to 1 to study the impact of the learning rate on the results using LSTM/BILSTM. The results are provided in Table 9.4.

Experiment Set 2, German Language. We evaluated the GRU /LSTM/BILSTM-2 layers encoder-decoder on the German dataset containing only 73k pairs. LSsare also complex in terms of atomic LSs. For instance, an LS can contain up to 6 atomic LSsA_i combined using operators $\sqcup$ and $\sqcap$. The results of the experiments are shown in Table 9.3. The results in Table 9.3 are obtained with the learning rate set to 0.01

for GRU and to 0.1 for LSTM/BiLSTM with a batch size of 265. We conducted more experiments with the learning rate set to 1 to study the impact of the learning rate on the results. The results are presented in Table 9.5.

Experiment Set 3, Transformer. We implemented our transformer model using the `Pytorch` framework with the default parameters, i.e., the number of epochs is 30, the batch size is 256, and the max sentence length is {107, 187}. The results are listed in Table 9.6.

Experiment Set 4, Few-shot learning on T5. To address the issues raised by employing conventional architectures, such as overfitting and limited vocabulary size as we have seen in previous experiments, we implemented few-shot learning of text generation on the T5 model with a small number of training samples. This experiment is designed with four distinct sets of few-shot training data and three distinct sets of testing data, as shown in Table 9.7. In the first experiment, we fine-tuned the T5 model using a training dataset of 10k pairs, each consisting of an LS and its English verbalization from LIMES silver data ①. In the second experiment, we fine-tuned T5 using the previous training dataset in combination with 70 pairs of LS and their human-annotated verbalizations from the LIMES silver data ②. In the third experiment, the training dataset from the second experiment is combined with human-annotated manipulated LIMES LS ③. By modifying the formula, Manipulated LIMES LS is defined differently from the previous LS. Additionally, we fine-tuned the T5 model on the training dataset in an effort to determine whether the model can improve the verbalization of manipulated LIMES LS. In the last experiment, we fine-tuned the T5 model using the training data from the previous experiment in

Tab. 9.2.: BLEU, METEOR, ChrF++, and TER scores for the English language, evaluated on the 107k dataset; the learning rate is 0.01 for GRU and 0.1 for LSTM/BILSTM.

Model	Length	Iter.	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
GRU	107	100	35.92	0.36	0.36	0.66	0.69
GRU	107	1000	41.05	0.41	0.39	0.71	0.63
GRU	187	1000	22.07	0.22	0.22	0.44	0.56
GRU	107	10000	**99.22**	**0.99**	**0.78**	**0.99**	**0.01**
GRU	187	10000	88.81	0.89	0.60	0.93	0.05
LSTM	107	100	82.61	0.83	0.65	0.92	0.27
LSTM	187	100	77.31	0.77	0.58	0.87	0.40
BiLSTM	107	100	85.37	0.85	0.64	0.91	0.26
BiLSTM	187	100	79.23	0.79	0.59	0.89	0.34

Tab. 9.3.: BLEU, METEOR, ChrF++, and TER scores for the German language evaluated on the 73K dataset. The learning rate is 0.01 for GRU and 0.10 for LSTM/BILSTM.

Model	Length	Iter.	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
GRU	107	100	41.84	0.42	0.43	0.74	0.66
GRU	107	1000	49.75	0.5o	0.47	0.79	0.59
GRU	187	1000	54.01	0.54	0.40	0.71	0.38
GRU	107	10000	**99.98**	**1.00**	**0.90**	**1.00**	**0.00**
GRU	187	10000	79.52	0.80	0.54	0.84	0.32
LSTM	107	100	60.40	0.60	0.44	0.70	0.45
LSTM	187	100	76.67	0.77	0.63	0.86	0.49
BiLSTM	107	100	81.90	0.82	0.59	0.86	0.21
BiLSTM	187	100	81.30	0.81	0.59	0.85	0.30

Tab. 9.4.: BLEU, METEOR, ChrF++, and TER scores for the English language evaluated on the 107K dataset with the learning rate set to 1.00.

Model	length	Iter.	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
LSTM	107	100	83.01	0.83	0.61	0.86	0.22
LSTM	187	100	68.06	0.68	0.67	0.89	0.55
BiLSTM	107	100	**94.45**	**0.94**	**0.70**	**0.96**	**0.08**
BiLSTM	187	100	86.18	0.86	0.66	0.89	0.16

combination with SILK LS ④. All experiments are built using the Pytorch lightning framework, with the following hyperparameters such as the number of epochs being five and the learning rate being 3e-5 and used beam search decoding to generate verbalization LS with parameters, e.g., *max length*=256, *num beams*=15, *no repeat ngram size*=6. In addition, *t5-base* is utilized as a pre-trained model. All mode LS based on few-shot learning are evaluated using the Table 9.7 test set, which is designed to investigate the effect of each training dataset on the model's ability to improve the generalization quality of LS verbalization.

Tab. 9.5.: BLEU and METEOR, ChrF++, and TER scores for the German language evaluated on the 73K pairs dataset with the learning rate set to 1.

Model	Length	Iter.	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
LSTM	107	100	87.19	0.87	0.66	0.90	0.13
LSTM	187	100	96.67	0.97	0.82	0.99	0.06
BiLSTM	107	100	91.74	0.92	0.71	0.93	0.07
BiLSTM	187	100	**99.58**	**1.00**	**0.85**	**1.00**	**0**

Tab. 9.6.: BLEU, METEOR, ChrF++, and TER scores for German and English evaluated on the 73K and 107K pairs datasets using Transformers.

Data	Length	Iter.	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
107K (En)	107	30	90.89	**0.91**	**0.67**	**0.98**	**0.12**
107K (En)	187	30	**90.92**	**0.91**	**0.67**	**0.98**	**0.12**
73K (De)	107	30	89.98	0.90	0.66	0.97	0.15
73K (De)	187	30	79.11	0.79	0.60	0.93	0.29

Tab. 9.7.: BLEU, METEOR, ChrF++, and TER scores for the English language using the Fine-tuned T5 model leveraging few-shot learning.

Train set	Test set	BLEU	BLEU-NLTK	METEOR	ChrF++	TER
①	LIMES original LS	76.27	0.76	0.54	0.87	0.15
①	SILK LS	34.26	0.35	0.26	0.54	0.71
②	LIMES original LS	77.91	0.78	0.54	0.89	0.13
②	LIMES Manipulated LS	45.76	0.46	0.37	0.68	0.55
③	LIMES Manipulated LS	63.64	0.64	0.43	0.80	0.48
③	SILK LS	34.93	0.35	0.27	0.54	0.67
④	SILK LS	36.58	0.37	0.34	0.59	0.62

9.3.4 Results and Analysis

To answer Q_1, we set the maximum length of a sentence to be either 107 words (tokens) or 187 words (tokens) based on the statistics in Table 9.1. This means that we filter out all verbalized sentences that have a length greater than 107 words for experiments where the maximum length of a sentence is 107 words. We also removed all verbalized sentences that exceed 187 words for experiments where the maximum sentence length was set at 187 words. In Table 9.2, we can observe that NMV-LS using GRU achieves a better BLEU score, up to 99.22 (Table 9.2). In Table 9.3, we can observe that the BLEU score is up to 99.98, obtained from our model using GRU when the length of a verbalized sentence is also less than or equal to 107. In Table 9.4, the BLEU score is 94.45 using BiLSTM with a max length of 107. Furthermore, Table 9.3 and Table 9.5 show that the NMV-LS model achieves better scores when the length of a verbalized sentence is 187. For example, the BLEU score in the $73k$ German dataset is 76.67 using LSTM and 99.58 using BiLSTM (Table 9.3 and Table 9.5). The reason is that the $73k$ German dataset contains complex LSs, and 28.40% of their verbalizations have sentence lengths greater than 100 words, and these sentences will be filtered out. In turn, this will affect the training process, especially since the size of the dataset is only $73k$ pairs, resulting in decreased performance. From all these observations, we conclude that the complexity of LS plays a crucial role in the performance of our NMV-LS model. Furthermore, GRU is more sensitive to the complexity of LS than LSTM/BILSTM. LSTM/BiLSTM can handle very complex LSsand improve performance.

To answer Q_2, we analyzed the results in Table 9.7. First, Limes original LS (i.e., it means that LS generated by Limes) is used to evaluate the second part of NMV-LS. Our findings indicate that our technique is capable of generating verbalization at the human level and outperforms previous approaches. For example, the BLUE score is 76.27, ChrF++ is 0.87, and METEOR is 0.54. Note that we fine-tuned our model with only ①. In other words, we only used the silver data Limes generated by the first stage of NMV-LS (i.e., rule-based verbalizer) to fine-tune our model. This answers Q_2.

To answer Q_3, we implemented the fine-tuned model in ① Limes silver data and evaluated on Silk LS as the most extreme case because Limes & Silk have different rules and grammars to build their LSs. The goal is to study to what extent our model can be generalized to verbalize LSs in different formats and from different systems (e.g., Silk). The results in Table 9.7 show that our model achieves a BLUE score of 34.26. Another case is that we fine-tuned NMV-LS using ② and tested on Limes manipulated LSs. In this case, NMV-LS scores 45.76 BLUE, 0.68 ChrF++, and 0.37. Another case is fine-tuning our model on training data ③ and evaluating it on Silk. From the results, there is no improvement compared to the result generated by the model fine-tuned on ① and tested on Silk. This can be justified as both ① and ③ do not contain any information about Silk. To further investigate this, we added a few samples of Silk LSs to create ④. To this end, we use ④ for fine-tuning NMV-LS and then evaluate on Silk. This improved performance by 1.65. We see these results, in all cases, as a big milestone toward generalizing our approach to verbalize LSs produced by other systems.

To answer Q_4, we implemented a couple of cases. In the first case, we fine-tuned NMV-LS using ② and evaluated it on the Limes original LSs. The results in Table 9.7 indicate that human-annotated verbalization of an LS improves the verbalization very slightly. For example, the BLUE score is 77.91 and is 1.65% higher than the BLUE score produced using ①. In the second case, we fine-tuned NMV-LS by applying ③ and evaluated it on Limes-manipulated LSs. This improved the performance by 17.88 in the BLUE score. We believe that this improvement is led by including Limes-manipulated LSsin the training data ③. This answers Q_4.

Conclusion and Future Work 10

10.1 Conclusion

In this thesis, we addressed several research gaps and challenges in the domain of LD over KGs with human in the loop, presenting contributions in terms of scalability, effectiveness, and explainability of LD approaches. Addressing **Research Gap 1: Geospatial Link Discovery over Knowledge Graphs at Scale**, we developed novel algorithms tailored for LD frameworks that emphasize scalability. Our contributions include the following: the *Line Simplification Approach* (Chapter 4), where we applied simplification as a preprocessing step for LD and achieved substantial improvements in runtime; the *Intersection Matrix Approach* (Chapter 5), an extension of the RADON strategy that achieves up to 35% speedup; and COBALT (Chapter 6), an approach combining R-Tree indexing with content-based measures that maintains an F-measure between 70% and 90%. To address **Research Gap 2: The Absence of Holistic Models for Linked Open Data**, we introduced NELLIE (Chapter 7), a modular pipeline architecture to build a holistic model for LOD by linking instances and ontologies within each KG in LOD, performing KG matching, linking, fusion, embedding, and link prediction. Addressing **Research Gap 3: Explainable Link Discovery**, we developed approaches for enhancing LD explainability. Our contributions here are twofold: a *Multilingual Verbalization and Summarization Approach* (Chapter 8) that produces complete and understandable verbalizations, though fluency decreases with complexity; and NMV-LS (Chapter 9), a language-based LS verbalization system that uses encoder-decoder and few-shot learning architectures, showing promising translation quality. In general, this thesis significantly advances the state of the art in LD over Geospatial KGs, improving efficiency, effectiveness, and explainability of LD approaches. Furthermore, it addresses key research gaps with implications for real-time applications, KG integration, and explainable AI.

In the subsequent paragraphs, we have distilled our findings and observations into a concise summary. This encapsulation highlights the core conclusions we have drawn from our comprehensive analysis of the thesis.

Link Discovery over Geospatial RDF: Line Simplification Approach (Chapter 4) We presented a study of the application of simplification as a preprocessing step for LD approaches for linking RDF resources with geospatial representation. Our evaluations showed that the LD approaches benefit significantly from the use of simplified geometries, with substantial improvements in F-measure and runtime, making them highly suitable for real-time applications such as question answering, where runtime is the key performance factor and result completeness comes second.

Link Discovery over Geospatial RDF: Intersection Matrix Approach (Chapter 5) In this chapter, we introduced RADON2, a strategy to scale the original approach, RADON, by computing the intersection matrix for each pair of resources once and then using it for all possible topological relations associated with the resources. We found that this approach can achieve up to 35% speedup in runtime.

Cobalt: A Content-Based Similarity Approach For Link Discovery over Geospatial Knowledge Graphs (Chapter 6) We proposed COBALT, an approach to topological relations discovery that combines R-Tree indexing with content-based measures to scale up the topological relations discovery process. Our experiments showed that COBALT can achieve a significant acceleration while maintaining an F-measure between 70% and 90%.

NELLIE: Never-Ending For Linked Open Data (Chapter 7) We presented NELLIE, a modular pipeline architecture developed to allow the collection of data necessary to build a holistic model for LOD by linking the instances and ontologies within each KG in LOD. NELLIE represents a milestone towards building a holistic model for LOD, performing KG matching, KG linking (including both ontology and instance matching), KG fusion, embedding, and link prediction.

Multilingual Verbalization and Summarization for Explainable Link Discovery (Chapter 8) We extended our previous work by introducing a multilingual template-based approach to verbalizing LS. Our approach produced both a direct literal verbalization of the content of the LS and a more natural aggregated version of the same content in English, German, and Spanish. The generated verbalizations were found to be complete and easily understandable, enabling non-expert users to understand the content of LS. However, the fluency of our approach decreases as the LS becomes more complex.

Explainable Integration of Knowledge Graphs using Large Language Models (Chapter 9) We presented NMV-LS, a language-based LS verbalization system that translates LS into natural language. The system consists of two independent parts: the first is based on standard encoder-decoder architectures, and the second applies the concept of few-shot learning. Both parts have shown promising results in terms of translation quality.

10.2 Future Work

As the field of Linked Data and KGs continues to evolve, the research presented in this thesis is a step towards realizing the full potential of these technologies. However, there are still many avenues for further exploration and improvement. In the interest of advancing the field, the following points elaborate on key future research directions that we aim to pursue.

Link Discovery over Geospatial *RDF*: Line Simplification Approach Our immediate focus will be on developing a specialized simplification algorithm capable of guaranteeing a minimum input F-measure. This is crucial to ensure that the simplified geometries retain essential characteristics. Through extensive experiments, our aim is to identify the optimal geometry simplification algorithm and its parameters for various sets of relations. A significant future aim is to study the scalability of these algorithms, focusing on optimizing an approach for large-scale *RDF* KGs.

Link Discovery over Geospatial *RDF*: Intersection Matrix Approach Our future work seeks to scale this approach to much larger datasets, which will involve rigorous performance testing and the development of advanced parallelization techniques. We also intend to marry this approach with previously developed simplification algorithms, especially those documented in [ASN18a], to achieve greater speed up in computation times.

Cobalt: A Content-Based Similarity Approach For Link Discovery over Geospatial Knowledge Graphs Efficiency improvements in COBALT will be the focus of our future work. We plan to investigate a wide range of indexing algorithms aimed at reducing memory footprint. Additional layers of information, such as location names and organizations that polygons represent, will be integrated into similarity calculations to improve link discovery accuracy.

Multilingual Verbalization and Summarization for Explainable Link Discovery To make LD processes more explainable, we intend to refine our existing verbalization methods. The focus will be on increasing the fluency and readability of the natural language output without altering the underlying LS. Furthermore, we aim to develop algorithms for consistency checks to improve the accuracy of generated text. Neural network-based approaches will also be refined to address identified challenges in verbalization and summarization.

Explainable Integration of Knowledge Graphs using Large Language Models The future scope involves extensive multilingual testing of the second part of NMV-LS, specifically targeting languages such as German, French, and Spanish. This is in line with our goal to make the LS learning algorithms capable of multilingual, real-time verbalization of dynamically learned LS.

NELLIE: Never-Ending Linking for Linked Open Data Future work involves integrating more approaches for each task within NELLIE. Automated KG matching approaches, e.g., *Tapioca* [Röd+16], are also on the radar for the complete automation of NELLIE. Furthermore, we intend to develop benchmark KGs that can holistically assess NELLIE on various tasks, such as link prediction and instance matching. Integration with LLMs is planned to enrich the KGs in LOD with unstructured data such as text.

By thoroughly addressing these future research avenues, we aim to advance the state of the art in areas such as LD, KG integration, and explainability.

Bibliography

[Ahm+23] Abdullah Fathi Ahmed, Asep Fajar Firmansyah, Mohamed Ahmed Sherif, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. "Explainable Integration of Knowledge Graphs Using Large Language Models". In: *Natural Language Processing and Information Systems*. Ed. by Elisabeth Métais, Farid Meziane, Vijayan Sugumaran, Warren Manning, and Stephan Reiff-Marganiec. Cham: Springer Nature Switzerland, 2023, pp. 124–139 (cit. on pp. 1, 139).

[ASN18a] Abdullah Fathi Ahmed, Mohamed Sherif, and Axel Ngonga Ngomo. "On the Effect of Geometries Simplification on Geo-spatial Link Discovery". In: *Proceedings of SEMANTiCS 2018*. 2018 (cit. on p. 155).

[Ahm+21] Abdullah Fathi Ahmed, Mohamed Ahmed Sherif, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. "Multilingual Verbalization and Summarization for Explainable Link Discovery". In: *Data and Knowledge Engineering* 133 (2021), p. 101874 (cit. on pp. 1, 44, 121, 140–142, 144, 145).

[ASN19a] Abdullah Fathi Ahmed, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. "LSVS: Link Specification Verbalization and Summarization". In: *International Conference on Applications of Natural Language to Information Systems*. Springer. 2019, pp. 66–78 (cit. on pp. 1, 43, 44, 121, 128, 140, 145).

[ASN23] Abdullah Fathi Ahmed, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. "NELLIE: Never-Ending Linking for Linked Open Data". In: *IEEE Access* 11 (2023), pp. 84957–84973 (cit. on pp. 1, 42, 91).

[ASN18b] Abdullah Fathi Ahmed, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. "On the Effect of Geometries Simplification on Geo-spatial Link Discovery". In: *Procedia Computer Science* 137 (2018). Proceedings of the 14th International Conference on Semantic Systems 10th – 13th of September 2018 Vienna, Austria, pp. 139–150 (cit. on pp. 1, 49, 70, 83).

[ASN18c] Abdullah Fathi Ahmed, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. "Radon2: a buffered-intersection matrix computing approach to accelerate link discovery over geo-spatial rdf knowledge bases". In: *Ontology Matching: OM-2018: Proceedings of the ISWC Workshop*. 2018, p. 197 (cit. on pp. 1, 63, 70).

[ASN19b] Abdullah Fathi Ahmed, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. "Do your Resources Sound Similar? On the Impact of Using Phonetic Similarity in Link Discovery". In: *K-CAP 2019: Knowledge Capture Conference*. 2019 (cit. on p. 1).

[Ara+11] Samur Araujo, Jan Hidders, Daniel Schwabe, and Arjen P De Vries. “Serimi-resource description similarity, rdf instance matching and interlinking”. In: *arXiv preprint arXiv:1107.1104* (2011) (cit. on p. 36).

[ALH09] Sören Auer, Jens Lehmann, and Sebastian Hellmann. “LinkedGeoData: Adding a Spatial Dimension to the Web of Data”. In: *The Semantic Web - ISWC 2009*. Ed. by Abraham Bernstein, David R. Karger, Tom Heath, et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 731–746 (cit. on pp. 8, 10, 12, 49, 63, 70, 92).

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015 (cit. on pp. 29, 43, 128, 143).

[BS12] Lakshmi Balasubramanian and M. Sugumaran. “A State-of-Art in R-Tree Variants for Spatial Indexing”. In: *International Journal of Computer Applications* 42 (Mar. 2012), pp. 35–41 (cit. on p. 71).

[BAH19] Ivana Balažević, Carl Allen, and Timothy M Hospedales. “TuckER: Tensor Factorization for Knowledge Graph Completion”. In: *Empirical Methods in Natural Language Processing*. 2019 (cit. on pp. 30, 111–113, 115, 116).

[BL05] Satanjeev Banerjee and Alon Lavie. “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments”. In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*. Ann Arbor, Michigan: ACL, 2005, pp. 65–72 (cit. on pp. 145, 146).

[Bar+20] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, et al. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. In: *Information Fusion* 58 (2020), pp. 82–115 (cit. on pp. 43, 44).

[BK11] Robert Battle and Dave Kolas. “GeoSPARQL : Enabling a Geospatial Semantic Web”. In: 2011 (cit. on p. 39).

[Bec+23] Alexander Becker, Abdullah Ahmed, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. “COBALT: A Content-Based Similarity Approach for Link Discovery over Geospatial Knowledge Graphs”. In: *SEMANTiCS*. 2023 (cit. on pp. 1, 69).

[BM04] Dave Beckett and Brian McBride. “RDF/XML syntax specification (revised)”. In: *W3C recommendation* 10.2.3 (2004) (cit. on p. 25).

[Bec+14] David Beckett, Tim Berners-Lee, Eric Prudâ€™hommeaux, and Gavin Carothers. “RDF 1.1 Turtle”. In: *World Wide Web Consortium* (2014) (cit. on p. 25).

[Bec+90] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. “The R*-tree: an efficient and robust access method for points and rectangles”. In: *SIGMOD ’90*. 1990 (cit. on p. 73).

[BHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web”. In: *Scientific American* 284.5 (May 2001), pp. 34–43 (cit. on pp. 7, 23).

[BN09] Jens Bleiholder and Felix Naumann. “Data Fusion”. In: *ACM Comput. Surv.* 41.1 (Jan. 2009) (cit. on p. 42).

[Bor+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. “Translating Embeddings for Modeling Multi-relational Data”. In: Curran Associates, Inc., 2013 (cit. on p. 42).

[Bow84] BR Bowring. “The direct and inverse solutions for the great elliptic line on the reference ellipsoid”. In: *Bulletin géodésique* 58.1 (1984), pp. 101–108 (cit. on p. 41).

[Bra+19] Daniel Braun, Kira Klimt, Daniela Schneider, and Florian Matthes. “SimpleNLG-DE: Adapting SimpleNLG 4 to German”. In: *Proceedings of the 12th International Conference on Natural Language Generation*. Association for Computational Linguistics. Tokio, Japan, 2019. published (cit. on p. 126).

[BK17] György Büttner and Barbara Kosztra. *CLC2018 Technical Guidelines*. European Environment Agency, 2017 (cit. on p. 77).

[Cha+21] Jaydeep Chakraborty, Hamada M. Zahera, Mohamed Ahmed Sherif, and Srividya K. Bansal. “ONTOCONNECT: Domain-Agnostic Ontology Alignment using Graph Embedding with Negative Sampling”. In: *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2021, pp. 942–945 (cit. on p. 37).

[Che+21a] Jiaoyan Chen, Pan Hu, Ernesto Jimenez-Ruiz, et al. “Owl2vec*: Embedding of owl ontologies”. In: *Machine Learning* 110.7 (2021), pp. 1813–1845 (cit. on p. 37).

[Che+21b] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, et al. “Augmenting ontology alignment by semantic embedding and distant supervision”. In: *European Semantic Web Conference*. Springer. 2021, pp. 392–408 (cit. on pp. 36, 106).

[Cho+14a] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. 2014. arXiv: `1409.1259 [cs.CL]` (cit. on p. 43).

[Cho+14b] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: `1406.1078 [cs.CL]` (cit. on pp. 43, 143, 144).

[CDV93] Eliseo Clementini, Paolino Di Felice, and Peter Van Oosterom. “A small set of formal topological relationships suitable for end-user interaction”. In: *International Symposium on Spatial Databases*. Springer. 1993, pp. 277–295 (cit. on pp. 31, 32).

[CSE94] Eliseo Clementini, Jayant Sharma, and Max J Egenhofer. “Modelling topological spatial relations: Strategies for query processing”. In: *Computers & graphics* 18.6 (1994), pp. 815–822 (cit. on pp. 31, 32, 51, 70, 71).

[DH96] Hercules Dalianis and Eduard Hovy. “Aggregation in natural language generation”. In: *Trends in Natural Language Generation An Artificial Intelligence Perspective*. Ed. by Giovanni Adorni and Michael Zock. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 88–105 (cit. on p. 126).

[Dev+18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018) (cit. on pp. 17, 95).

[Dod02] George Doddington. “Automatic evaluation of machine translation quality using n-gram co-occurrence statistics”. In: *Proceedings of HLT*. 2002, pp. 138–145 (cit. on pp. 123, 130).

[Don+17] Yinpeng Dong, Hang Su, Jun Zhu, and Bo Zhang. *Improving Interpretability of Deep Neural Networks with Semantic Information*. 2017. arXiv: `1703.04096 [cs.CV]` (cit. on p. 43).

[DP73] David H Douglas and Thomas K Peucker. “Algorithms for the reduction of the number of points required to represent a digitized line or its caricature”. In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 10.2 (1973), pp. 112–122 (cit. on pp. 51, 55, 78).

[DHS+73] Richard O Duda, Peter E Hart, David G Stork, et al. *Pattern classification*. Vol. 2. Wiley New York, 1973 (cit. on p. 41).

[EMH94] M. J Egenhofer, D. M Mark, and J. Herring. *The 9-Intersection: Formalism and Its Use for Natural-Language Spatial Predicates (94-1)*. UC Santa Barbara: National Center for Geographic Information and Analysis, 1994 (cit. on pp. 70, 71).

[EF91] Max J Egenhofer and Robert D Franzosa. “Point-set topological spatial relations”. In: *International Journal of Geographical Information System* 5.2 (1991), pp. 161–174 (cit. on p. 31).

[EHZ18] Hesham M Eraqi, Jens Honer, and Sebastian Zuther. “Static Free Space Detection with Laser Scanner using Occupancy Grid Maps”. In: *arXiv preprint arXiv:1801.00600* (2018) (cit. on p. 40).

[F C21] Luciano da F. Costa. *Further Generalizations of the Jaccard Index*. 2021. arXiv: `2110.09619 [cs.LG]` (cit. on p. 95).

[GR04] Francisco Godoy and Andrea Rodriguez. “Defining and Comparing Content Measures of Topological Relations”. In: *Geoinformatica* 8.4 (Dec. 2004), pp. 347–371 (cit. on pp. 33, 73, 74).

[Gut84] Antonin Guttman. “R-Trees: A Dynamic Index Structure for Spatial Searching”. In: *SIGMOD Rec.* 14.2 (June 1984), pp. 47–57 (cit. on p. 71).

[HB11] Tom Heath and Christian Bizer. “Linked data: Evolving the web into a global data space”. In: *Synthesis lectures on the semantic web: theory and technology* 1.1 (2011), pp. 1–136 (cit. on pp. 23, 24).

[HG97] Paul S Heckbert and Michael Garland. *Survey of polygonal surface simplification algorithms*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1997 (cit. on pp. 51, 52).

[Hit27] Frank L Hitchcock. "The expression of a tensor or a polyadic as a sum of products". In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189 (cit. on p. 111).

[HKR09] Pascal Hitzler, Markus Krtzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. 1st. Chapman & Hall/CRC, 2009 (cit. on p. 24).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780 (cit. on p. 143).

[Hua+19] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. "Knowledge graph embedding based question answering". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM. 2019 (cit. on p. 42).

[HKR93] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. "Comparing images using the Hausdorff distance". In: *IEEE Transactions on pattern analysis and machine intelligence* 15.9 (1993), pp. 850–863 (cit. on p. 41).

[Inc11] Open Geospatial Consortium Inc. *OpenGIS Implementation Standard for Geographic information - Simple feature access*. Ed. by John R. Herring. 1.2.1. 2011 (cit. on p. 30).

[IJB11] R. Isele, A. Jentzsch, and C. Bizer. "Efficient Multidimensional Blocking for Link Discovery without losing Recall". In: *WebDB*. 2011 (cit. on pp. 11, 13, 18, 26, 36, 92, 103, 121, 140).

[IAK21] Vivek Iyer, Arvind Agarwal, and Harshit Kumar. "VeeAlign: Multifaceted Context Representation Using Dual Attention for Ontology Alignment". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10780–10792 (cit. on p. 37).

[JC11] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. "LogMap: Logic-Based and Scalable Ontology Matching". In: *The Semantic Web – ISWC 2011*. Ed. by Lora Aroyo, Chris Welty, Harith Alani, et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 273–288 (cit. on pp. 13, 25, 36, 92, 102).

[Jin+21] Xiongnan Jin, Sungkwang Eom, Sangjin Shin, Kyong-Ho Lee, and Chaoqun Hong. "DORIC: discovering topological relations based on spatial link composition". In: *Knowledge and Information Systems* 63.10 (Oct. 2021), pp. 2645–2669 (cit. on pp. 13, 39, 70).

[KB13] Nal Kalchbrenner and Phil Blunsom. "Recurrent Continuous Translation Models". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, 2013, pp. 1700–1709 (cit. on p. 43).

[KF94] Ibrahim Kamel and Christos Faloutsos. “Hilbert R-Tree: An Improved R-Tree Using Fractals”. In: *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB ’94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 500–509 (cit. on p. 73).

[KF93] Ibrahim Kamel and Christos Faloutsos. “On Packing R-Trees”. In: *Proceedings of the Second International Conference on Information and Knowledge Management*. CIKM ’93. Washington, D.C., USA: Association for Computing Machinery, 1993, pp. 490–499 (cit. on p. 73).

[Kan+16] Minoru Kanehisa, Miho Furumichi, Mao Tanabe, Yoko Sato, and Kanae Morishima. “KEGG: new perspectives on genomes, pathways, diseases and drugs”. In: *Nucleic Acids Research* 45.D1 (Nov. 2016), pp. D353–D361. eprint: `https://academic.oup.com/nar/article-pdf/45/D1/D353/8846820/gkw1092.pdf` (cit. on pp. 12, 92).

[KP18] Seyed Mehran Kazemi and David Poole. “SimplE Embedding for Link Prediction in Knowledge Graphs”. In: *Advances in Neural Information Processing Systems*. 2018 (cit. on p. 112).

[Kly04] Graham Klyne. “Resource description framework (RDF): Concepts and abstract syntax”. In: *http://www. w3. org/TR/2004/REC-rdf-concepts-20040210/* (2004) (cit. on p. 24).

[KTR09] Hanna Köpcke, Andreas Thor, and Erhard Rahm. “Comparative evaluation of entity resolution approaches with FEVER”. In: *Proc. VLDB Endow.* 2.2 (2009), pp. 1574–1577 (cit. on pp. 123, 130, 131).

[KKK12] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. “Strabon: A Semantic Geospatial DBMS”. In: vol. 7649. Nov. 2012 (cit. on p. 39).

[LL03] Taewon Lee and Sukho Lee. “OMT: Overlap Minimizing Top-down Bulk Loading Algorithm for R-tree.” In: *CAISE Short paper proceedings*. Vol. 74. 2003, pp. 69–72 (cit. on p. 73).

[Leh+12] Jens Lehmann, Tim Furche, Giovanni Grasso, et al. “DEQA: Deep Web Extraction for Question Answering”. In: *Proceedings of ISWC*. 2012 (cit. on p. 50).

[LLE97] Scott Leutenegger, Mario Lopez, and Jeffrey Edgington. “STR: A Simple and Efficient Algorithm for R-Tree Packing”. In: May 1997, pp. 497–506 (cit. on p. 73).

[Lin+15] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. “Learning entity and relation embeddings for knowledge graph completion”. In: *Twenty-ninth AAAI conference on artificial intelligence*. 2015 (cit. on p. 42).

[Los+11] Matthias Loskyll, Jochen Schlick, Stefan Hodek, et al. “Semantic service discovery and orchestration for manufacturing processes”. In: *ETFA2011*. IEEE. 2011, pp. 1–8 (cit. on p. 8).

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 1412–1421 (cit. on p. 137).

[Meh+15] Nijat Mehdiyev, Julian Krumeich, David Enke, Dirk Werth, and Peter Loos. "Determination of rule patterns in complex event processing using machine learning techniques". In: *Procedia Computer Science* 61 (2015), pp. 395–401 (cit. on p. 8).

[Men18] Thomas Mendel. "Area-Preserving Subdivision Simplification with Topology Constraints: Exactly and in Practice". In: *2018 Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM. 2018, pp. 117–128 (cit. on p. 40).

[MMB12] Pablo N. Mendes, Hannes Mühleisen, and Christian Bizer. "Sieve: Linked Data Quality Assessment and Fusion". In: *Proceedings of the 2012 Joint EDBT/ICDT Workshops*. EDBT-ICDT '12. Berlin, Germany: Association for Computing Machinery, 2012, pp. 116–123 (cit. on pp. 13, 42, 92).

[MT19] Michalis Mountantonakis and Yannis Tzitzikas. "Large-Scale Semantic Integration of Linked Data: A Survey". In: *ACM Comput. Surv.* 52.5 (Sept. 2019) (cit. on pp. 13, 92).

[Mou+20] Diego Moussallem, Paramjot Kaur, Thiago Ferreira, et al. "A General Benchmarking Framework for Text Generation". English. In: International Workshop on Natural Language Generation from the Semantic Web 2020, WebNLG+ ; Conference date: 18-12-2020 Through 18-12-2020. 2020, pp. 27–33 (cit. on pp. 141, 146).

[Mou+19] Diego Moussallem, Axel-Cyrille Ngonga Ngomo, Paul Buitelaar, and Mihael Arcan. "Utilizing Knowledge Graphs for Neural Machine Translation Augmentation". In: *Proceedings of the 10th International Conference on Knowledge Capture*. 2019, pp. 139–146 (cit. on pp. 44, 128).

[MWN18] Diego Moussallem, Matthias Wauer, and Axel-Cyrille Ngonga Ngomo. "Machine Translation using Semantic Web Technologies: A Survey". In: *Journal of Web Semantics* 51 (2018), pp. 1–19 (cit. on p. 140).

[Nen+17] Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. "A survey of current link discovery frameworks". In: *Semantic Web* 8.3 (2017), pp. 419–436 (cit. on pp. 9, 10, 13, 47, 92).

[NL13] A. N. Ngomo and K. Lyko. "Unsupervised learning of link specifications: deterministic vs. non-deterministic". In: *OM*. 2013 (cit. on p. 105).

[NA11] Axel-Cyrille Ngonga Ngomo and Sören Auer. "Limes-a time-efficient approach for large-scale link discovery on the web of data." In: *IJCAI*. 2011, pp. 2312–2317 (cit. on pp. 11, 13, 18, 36, 47, 54).

[Ngo+14a] Axel-Cyrille Ngonga Ngomo, Sören Auer, Jens Lehmann, and Amrapali Zaveri. "Introduction to Linked Data and Its Lifecycle on the Web". In: *Reasoning Web. Reasoning on the Web in the Big Data Era: 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings*. Ed. by Manolis Koubarakis, Giorgos Stamou, Giorgos Stoilos, et al. Cham: Springer International Publishing, 2014, pp. 1–99 (cit. on pp. 8, 24).

[Ngo+14b] Axel-Cyrille Ngonga Ngomo, Sören Auer, Jens Lehmann, and Amrapali Zaveri. "Introduction to linked data and its lifecycle on the web". In: *Reasoning Web International Summer School*. Springer. 2014, pp. 1–99 (cit. on p. 37).

[Ngo11] Axel-Cyrille Ngonga Ngomo. "A Time-Efficient Hybrid Approach to Link Discovery". In: *Proceedings of OM@ISWC*. 2011 (cit. on pp. 92, 121).

[Ngo13] Axel-Cyrille Ngonga Ngomo. "ORCHID - Reduction-Ratio-Optimal Computation of Geo-Spatial Distances for Link Discovery". In: *Proceedings of ISWC 2013*. 2013 (cit. on pp. 10, 37, 49, 50, 69).

[Ngo+14c] Axel-Cyrille Ngonga Ngomo, Sören Auer, Jens Lehmann, and Amrapali Zaveri. "Introduction to linked data and its lifecycle on the web". In: *Reasoning Web. Semantic Technologies for Intelligent Data Access*. Springer Berlin Heidelberg, 2014, pp. 1–90 (cit. on p. 26).

[NL12] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. "EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming". In: *The Semantic Web: Research and Applications*. Ed. by Elena Simperl, Philipp Cimiano, Axel Polleres, Oscar Corcho, and Valentina Presutti. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 149–163 (cit. on pp. 26, 36, 43, 122, 130).

[Ngo+21] Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, Kleanthi Georgala, et al. "LIMES - A Framework for Link Discovery on the Semantic Web". In: *KI-Künstliche Intelligenz, German Journal of Artificial Intelligence - Organ des Fachbereichs "Künstliche Intelligenz" der Gesellschaft für Informatik e.V.* (2021) (cit. on pp. 13, 47, 92, 93, 95, 103, 140).

[NVJ20] Hoang Long Nguyen, Dang Thinh Vu, and Jason J. Jung. "Knowledge graph fusion for smart systems: A Survey". In: *Information Fusion* 61 (2020), pp. 56–70 (cit. on p. 42).

[NIL12] Khai Nguyen, Ryutaro Ichise, and Bac Le. "SLINT: A Schema-Independent Linked Data Interlinking System". In: *Proceedings of the 7th International Conference on Ontology Matching - Volume 946*. OM'12. Boston, MA: CEUR-WS.org, 2012, pp. 1–12 (cit. on p. 36).

[NRP] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. "Holographic Embeddings of Knowledge Graphs". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI'16, pp. 1955–1961 (cit. on p. 42).

[NTK11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. "A Three-Way Model for Collective Learning on Multi-Relational Data." In: *ICML*. Vol. 11. 2011 (cit. on p. 42).

[NMS10] Mathias Niepert, Christian Meilicke, and Heiner Stuckenschmidt. “A Probabilistic-Logical Framework for Ontology Matching.” In: *AAAI*. Citeseer. 2010, pp. 1413–1418 (cit. on pp. 13, 36, 92).

[Niu+12] Xing Niu, Shu Rong, Haofen Wang, and Yong Yu. “An Effective Rule Miner for Instance Matching in a Web of Data”. In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. CIKM ’12. Maui, Hawaii, USA: Association for Computing Machinery, 2012, pp. 1085–1094 (cit. on p. 36).

[Pap+21] George Papadakis, Georgios Mandilaras, Nikos Mamoulis, and Manolis Koubarakis. “Progressive, Holistic Geospatial Interlinking”. In: *Proceedings of the Web Conference 2021*. WWW ’21. Ljubljana, Slovenia: Association for Computing Machinery, 2021, pp. 833–844 (cit. on pp. 13, 38, 70, 78).

[Pap+02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. “BLEU: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting on Association for Computational Linguistics*. 2002 (cit. on pp. 145, 146).

[Pop17] Maja Popović. “chrF++: words helping character n-grams”. In: *Proceedings of the Second Conference on Machine Translation*. 2017, pp. 612–618 (cit. on p. 146).

[Raf+19] Colin Raffel, Noam Shazeer, Adam Roberts, et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *CoRR* abs/1910.10683 (2019). arXiv: `1910.10683` (cit. on pp. 14, 44, 144).

[RJB17] A. Ramos-Soto, J. Janeiro-Gallardo, and Alberto Bugarín. “Adapting SimpleNLG to Spanish”. In: Association for Computational Linguistics, 2017, pp. 142–146 (cit. on p. 127).

[Rei07] Ehud Reiter. “An Architecture for Data-to-Text Systems”. In: *Proceedings of the Eleventh European Workshop on Natural Language Generation*. ENLG ’07. Germany: Association for Computational Linguistics, 2007, pp. 97–104 (cit. on p. 43).

[RD00] Ehud Reiter and Robert Dale. *Building natural language generation systems*. New York, NY, USA: Cambridge University Press, 2000 (cit. on pp. 19, 43, 122, 123, 142).

[Röd+16] Michael Röder, Axel-Cyrille Ngonga Ngomo, Ivan Ermilov, and Andreas Both. “Detecting Similar Linked Datasets Using Topic Modelling”. In: *The Semantic Web. Latest Advances and New Domains*. Ed. by Harald Sack, Eva Blomqvist, Mathieu d’Aquin, et al. Cham: Springer International Publishing, 2016, pp. 3–19 (cit. on p. 156).

[Sal+15] Muhammad Saleem, Muhammad Intizar Ali, Aidan Hogan, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. “LSQ: The Linked SPARQL Queries Dataset”. In: *The Semantic Web - ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*. Bethlehem, PA, USA: Springer-Verlag, 2015, pp. 261–269 (cit. on p. 8).

[San+18] Georgios M. Santipantakis, Christos Doulkeridis, George A. Vouros, and Akrivi Vlachou. "MaskLink: Efficient Link Discovery for Spatial Relations via Masking Areas". In: *CoRR* abs/1803.01135 (2018). arXiv: 1803.01135 (cit. on p. 39).

[Sch+11] Andreas Schultz, Andrea Matteini, Robert Isele, Christian Bizer, and Christian Becker. "LDIF-linked data integration framework". In: *Proceedings of the Second International Conference on Consuming Linked Data-Volume 782*. CEUR-WS. org. 2011, pp. 125–130 (cit. on pp. 13, 42, 92).

[SHB16] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 1715–1725 (cit. on p. 137).

[SAH18] Yilang Shen, Tinghua Ai, and Yakun He. "A New Approach to Line Simplification Based on Image Processing: A Case Study of Water Area Boundaries". In: *ISPRS International Journal of Geo-Information* 7.2 (2018), p. 41 (cit. on p. 40).

[She+17] Mohamed Ahmed Sherif, Kevin Dreßler, Panayiotis Smeros, and Axel-Cyrille Ngonga Ngomo. "RADON - Rapid Discovery of Topological Relations". In: *Proceedings of The Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*. 2017, pp. 175–181 (cit. on pp. 13, 38, 50, 54, 55, 64, 66, 70, 78).

[SN15] Mohamed Ahmed Sherif and Axel-Cyrille Ngonga Ngomo. "A systematic survey of point set distance measures for link discovery". In: *Semantic Web Journal.(Cited on page 18.)* (2015) (cit. on pp. 37, 40, 50, 54).

[SNL15] Mohamed Ahmed Sherif, Axel-Cyrille Ngonga Ngomo, and Jens Lehmann. "Automating RDF Dataset Transformation and Enrichment". In: *Proceedings of 12th Extended Semantic Web Conference*. Ed. by Fabien Gandon, Marta Sabou, Harald Sack, et al. Cham: Springer, 2015, pp. 371–387 (cit. on pp. 13, 50, 92).

[SNL17] Mohamed Ahmed Sherif, Axel-Cyrille Ngonga Ngomo, and Jens Lehmann. "WOMBAT - A Generalization Approach for Automatic Link Discovery". In: *14th Extended Semantic Web Conference, Portorož, Slovenia, 28th May - 1st June 2017*. Springer. Springer, 2017, pp. 103–119 (cit. on pp. 26, 27, 36, 43, 50, 93, 103, 106, 122, 123, 131, 142).

[She+23] Mohamed Ahmed Sherif, Ana Alexandra Morim da Silva, Svetlana Pestryakova, et al. "IngridKG: A FAIR Knowledge Graph of Graffiti". In: *Scientific Data* (2023) (cit. on p. 1).

[SK16] Panayiotis Smeros and Manolis Koubarakis. "Discovering Spatial and Temporal Links among RDF Data." In: *LDOW@ WWW*. 2016 (cit. on pp. 38, 50).

[Sno+06] Matthew Snover, B. Dorr, R. Schwartz, and L. Micciulla. "A Study of Translation Edit Rate with Targeted Human Annotation". In: 2006 (cit. on p. 146).

[SVL14a] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. arXiv: `1409.3215 [cs.CL]` (cit. on pp. 43, 143).

[SVL14b] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger. 2014, pp. 3104–3112 (cit. on p. 145).

[Thi+20] Elodie Thiéblin, Ollivier Haemmerlé, Nathalie Hernandez, and Cassia Trojahn. “Survey on complex ontology matching”. In: *Semantic Web* 11.4 (2020), pp. 689–727 (cit. on p. 25).

[Tro+16] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. “Complex embeddings for simple link prediction”. In: *International Conference on Machine Learning*. 2016 (cit. on pp. 30, 42, 111).

[Tuc64] Ledyard R Tucker. “The Extension of Factor Analysis to Three-Dimensional Matrices”. In: *Contributions to Mathematical Psychology* 110119 (1964) (cit. on p. 111).

[Ung+14] Christina Unger, Corina Forascu, Vanessa Lopez, et al. “Question answering over linked data (QALD-4)”. In: *Working Notes for CLEF 2014 Conference*. 2014 (cit. on p. 9).

[Usb+15] Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, et al. “GERBIL: General Entity Annotator Benchmarking Framework”. In: *Proceedings of the 24th International Conference on World Wide Web*. WWW ’15. Florence, Italy: International World Wide Web Conferences Steering Committee, 2015, pp. 1133–1143 (cit. on p. 146).

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: `1706.03762` (cit. on p. 144).

[VW95] Mahes Visvalingam and Peter J Williamson. “Simplification and generalization of large scale data for roads: a comparison of two filtering algorithms”. In: *Cartography and Geographic Information Systems* 22.4 (1995), pp. 264–275 (cit. on p. 55).

[VW93] Maheswari Visvalingam and James D Whyatt. “Line generalisation by repeated elimination of points”. In: *The Cartographic Journal* 30.1 (1993), pp. 46–51 (cit. on p. 52).

[Wan+17] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. “Knowledge graph embedding: A survey of approaches and applications”. In: *IEEE Transactions on Knowledge and Data Engineering* 29.12 (2017), pp. 2724–2743 (cit. on p. 42).

[Yan+15] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. “Embedding Entities and Relations for Learning and Inference in Knowledge Bases”. In: *International Conference on Learning Representations*. 2015 (cit. on pp. 30, 111).

[ZL05] Chuanrong Zhang and Weidong Li. “The roles of web feature and web map services in real-time geospatial data sharing for time-critical applications”. In: *Cartography and Geographic Information Science* 32.4 (2005), pp. 269–283 (cit. on pp. 10, 70).

[Zha+18a] Shu-kai Zhang, Guo-you Shi, Zheng-jiang Liu, Zhi-wei Zhao, and Zhao-lin Wu. “Data-driven based automatic maritime routing from massive AIS trajectories in the face of disparity”. In: *Ocean Engineering* 155 (2018), pp. 240–250 (cit. on p. 40).

[ZZ16] Mengyi Zhao and Songmao Zhang. “Identifying and validating ontology mappings by formal concept analysis.” In: *OM@ ISWC*. 2016, pp. 61–72 (cit. on p. 102).

[Zha+18b] Mengyi Zhao, Songmao Zhang, Weizhuo Li, and Guowei Chen. “Matching biomedical ontologies based on formal concept analysis”. In: *Journal of biomedical semantics* 9.1 (2018), pp. 1–27 (cit. on pp. 102, 106).

[Zho+18] Lu Zhou et al. “A Journey From Simple to Complex Alignment on Real-World Ontologies.” In: *DC@ ISWC*. 2018, pp. 93–101 (cit. on p. 25).

[ZK21] Christian Zinke-Wehlmann and Amit Kirschenbaum. “Geo-L: Topological Link Discovery for Geospatial Linked Data Made Easy”. In: *ISPRS International Journal of Geo-Information* 10.10 (2021) (cit. on p. 39).

List of Figures

List of Tables

List of Listings

Declaration

Declaration: Translation from German:

I hereby declare that I prepared this thesis entirely on my own and have not used outside sources without declaration in the text. Any concepts or quotations applicable to these sources are clearly attributed to them. This thesis has not been submitted in the same or substantially similar version, not even in part, to any other authority for grading and has not been published elsewhere.

Original declaration text in German:

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

Paderborn, August 12, 2024

Abdullah Fathi Ahmed Ahmed