Scalable Data Integration for Linked Data

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

DISSERTATION

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(DR. RER. NAT.)

im Fachgebiet Informatik

vorgelegt von
M. Sc. Informatik Markus Nentwig
geboren am 15. April 1986 in Lutherstadt Wittenberg

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Erhard Rahm (Universität Leipzig)
2. Prof. Dr. Peter Christen (Australian National University)

Die Verleihung des akademischen Grades erfolgt mit Bestehen der Verteidigung am 24. Juni 2020 mit dem Gesamtprädikat *magna cum laude*.

UNIVERSITÄT
LEIPZIG

Acknowledgments

This dissertation is the result of the past seven years while working as a research assistant in the Database Group at Leipzig University. For most of this time, my work was funded by the German Research Foundation (Deutsche Forschungsgemeinschaft) in the LOD Link Discovery project. I would like to thank Prof. Dr. Erhard Rahm for the opportunity to do all the research that ultimately led to this thesis. Thank you for the professional support and guidance during my time at the Database Group.

A significant part of the dedication to my work is due to my colleagues. I would like to thank all of them for the wonderful time we spent together, the constructive character of many professional and philosophical discussions as well as the joint activities. In particular, I would like to thank Prof. Dr. Anika Groß, who has given me the necessary encouragement at many points to be able to pursue the ambitious goal of this dissertation. Further thanks goes to Martin Junghanns and André Petermann, who influenced my thematic direction towards distributed processing on graph data structures. I would also like to thank Victor Christen and Alieh Saeedi for the inspiring discussions on research topics and the feedback on my dissertation. Another thank you goes to our secretary Andrea Hesse, who besides her actual work always made sure that there is a nice social atmosphere in the working group.

I owe special thanks to my parents Ursula and Peter Nentwig as well as my sisters Susi and Juliane and my brother Johannes – who always lovingly supported me. Last but not least, a huge thank you goes to Josephine Rast for the effort in proofreading, all the little things that brightened up my day, and her good sense when I need a little extra pressure to get ahead. Certainly that includes our son Moritz, who also gave me a lot of strength to finish the work.

Leipzig, 18. Dezember 2019 Markus Nentwig

Abstract

Linked Data describes an extensive set of structured but heterogeneous data sources where entities are connected by formal semantic descriptions. In the vision of the Semantic Web, these semantic links are extended towards the World Wide Web to provide as much machine-readable data as possible for search queries. The resulting connections allow an automatic evaluation to find new insights into the data. The Semantic Web benefits from the use of standardized web technologies and semantic vocabularies such as ontologies to allow computer systems to associate data from different sources. Identifying these semantic connections between two data sources with automatic approaches is called link discovery. We derive common requirements and a generic link discovery workflow based on similarities between entity properties and associated properties of ontology concepts. Many link discovery approaches with extensive techniques have been proposed to determine links between two sources. But most of the previous approaches disregard the fact that in times of Big Data, an increasing volume of data sources poses new demands on link discovery. In particular, the problem of complex and time-consuming link determination escalates with an increasing number of intersecting data sources. To overcome the restriction of pairwise linking of entities, holistic clustering approaches are needed to link equivalent entities of multiple data sources to construct integrated knowledge bases. In this context, the focus on efficiency and scalability is essential. For example, reusing existing links or background information can help to avoid redundant calculations. However, when dealing with multiple data sources, additional data quality problems must also be dealt with. This dissertation addresses these comprehensive challenges by designing holistic linking and clustering approaches that enable reuse of existing links. At first, the LinkLion portal will be introduced to provide existing links for new applications. These links act as a basis for a physical data integration process to create a unified representation for equivalent entities from many data sources. Unlike previous systems, we execute the complete data integration workflow via a distributed processing system.

The development of the LinkLion platform permits numerous link discovery frameworks to manage calculated or inferred links and make them available for reuse. The easy-to-use web portal allows the upload of new links for arbitrary semantic relations as well as the addition and maintenance of provenance information. Moreover, the links between knowledge bases can be accessed via open interfaces such as a SPARQL endpoint in any other application. We then propose a holistic clustering approach to form consolidated clusters for same real-world entities from many different sources. The reuse of existing links from the LinkLion repository reduces unnecessary entity comparisons. At the same time, we exploit the semantic type of entities to improve the quality of the result. The process identifies errors in existing links and can find numerous additional links. A preliminary evaluation of geographic real-world Linked Data shows the effectiveness of the approach. Additionally, the entity clustering has to react to the high dynamics of the data. In particular, this requires scalable approaches for continuously growing data sources with many entities as well as additional new sources. Previous entity clustering approaches are mostly static, focusing on the one-time linking and clustering of entities from few sources. Therefore, we propose and evaluate new approaches for incremental entity clustering that supports the continuous addition of new entities and data sources. A detailed performance evaluation with real and synthetically customized datasets shows the effectiveness and scalability of the incremental clustering approaches.

To cope with the ever-increasing number of Linked Data sources, efficient and scalable methods based on distributed processing systems are required. Thus we propose distributed holistic approaches to link many data sources based on a clustering of entities that represent the same real-world object. The implementation is realized on Apache Flink. In contrast to previous approaches, we utilize efficiency-enhancing optimizations for both distributed static and dynamic clustering. This includes the use of blocking strategies to reduce the search space. A compact representation of the clusters by fused representatives allows a further reduction of unnecessary comparisons. Also, the approach detects errors in existing links and many new links. The extensive evaluation of the distributed clustering strategies shows high effectiveness for datasets from multiple domains as well as scalability on a multi-machine Apache Flink cluster. Furthermore, a comprehensive comparative evaluation of various general-purpose and specialized distributed clustering approaches is performed considering both match quality as well as scalability for a varying number of machines and data sizes.

Overall, the presented holistic clustering approaches produce results of high quality and are scalable for large sets of entities and sources. The approaches allow for interconnecting entities from multiple Linked Data sources while providing a compact reusable representation to build consolidated knowledge graphs. The solution enables a holistic data integration that serves interesting applications and analytics based on the combination of data from a variety of Linked Data sources.

Contents

List of Figures

List of Tables

Part I

Foundations

1 Introduction

This opening chapter discusses the background to relevant subjects and gives an outline for the structure of the dissertation. Section 1.1 motivates the topic with an introduction to data integration and the relation towards the Semantic Web as well as the phenomenon of Big Data. Requirements are derived from the identified challenges to perform scalable data integration for Linked Data. On this basis, scientific contributions are specified in Section 1.2. Finally, Section 1.3 gives an overview of the remaining work.

1.1 Motivation

Global corporations like Google, Microsoft or IBM make investments to create substantial knowledge bases and to integrate information from many different data sources [46, 59, 196]. One motivating example is IBM Watson, a question-answering system to support human experts in specialized tasks using many interconnected data sources [59]. These sources include dictionaries, encyclopedias and literary writings, which are being enriched with additional semantic descriptions. A commonly known use case for Watson is the successful challenge of human experts in the "Jeopardy!" show. Besides other applications, Watson is also applied to health care scenarios, e. g., supporting the discovery of novel therapeutics [29]. Search engines are another example of combining information from different sources, in this case, to provide answers to complex user queries. By detecting the semantic context of the query, the quality of the results can be improved by presenting relevant additional information. A search query for a current movie results in the expected list-based outcome. But in addition, the user is also informed of nearby showtimes, movie ratings and background information on the actors. The basic principle for both examples is the combination of knowledge from different data sources to derive additional information for users. Within the healthcare use case, IBM Watson includes data sources covering patents, genes, drugs, chemical compounds and published studies; for the movie use case, search engines pair Wikipedia information with movie rating pages, and location information enables the search engine provider to show nearest screenings for the requested movie.

1.1.1 Data Integration

This consolidation of knowledge from different data sources to find additional insights is summarized under the term *data integration* [114]. Starting in the 1980s, the idea of integrating databases emerged with the need for querying data located on physically isolated computer systems [173]. The data distribution typically leads to increased autonomy and heterogeneity of the data sources, which generates problems for data integration [43, 115]. The autonomy of a data source describes its dependence on other sources. A high degree of autonomy enables the possibility of structuring the contained

Table 1.1: Person entities from two data sources with challenges for data integration, e. g., missing, wrong or misspelled values (Leibntz instead of Leibniz, Stagira (Greece) instead of Leipzig (Germany)), semantic ambiguities on instance and schema level. Arrows indicate entities representing equivalent objects.

(a) Example person entities in source A.

ID	name	surname	birthPlace	type
$A1$	Gottfried Wilhelm	Leibniz	Leipzig (Germany)	person
$A2$	Friedrich	Leibniz	-	person
		…		

equivalence relation

(b) Example person entities in source B.

ID	label	place_of_birth	type
$B0$	Gottfried W. Leibntz	Stagira (Greece)	philosopher
$B1$	Friedich Leibniz	Leipzig (Germany)	philosopher
$B2$	Aristotle	-	philosopher
	…		

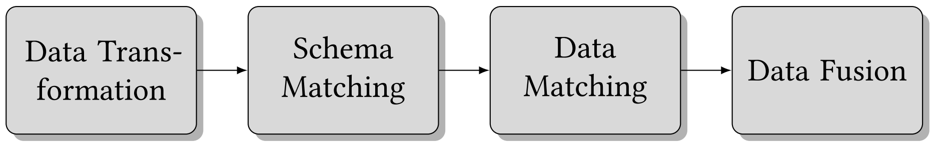


Figure 1.1: General workflow for physical data integration.

data individually. The downside of this individuality is increased heterogeneity between data sources, for example, by using different data models or technical requirements. The need for data integration for heterogeneous data sources causes plenty of problems. We illustrate some of these problems in Table 1.1 based on person entities for data sources A and B. Thus, we show data quality problems, such as inconsistent or misspelled data entries as well as general heterogeneity problems [158].

When creating workflows to integrate data from multiple data sources, two basic approaches are distinguished. Physical data integration focuses on combining data from different sources to establish a new representation, e. g., the creation of data warehouses or knowledge graphs. Alternatively, virtual data integration retrieves the required data from the sources and combines them at runtime as required, e. g., for federated query processing [43, 155]. Typical use cases for the integration of a large number of data sources frequently rely on physical data integration [155]. This integration primarily takes place at the instance level, for example, by matching or clustering entities from different data sources.

In this work, we use a physical data integration approach to create an integrated knowledge representation [45, 155]. The central components of the workflow are data transformation, schema matching, data matching and data fusion (see Figure 1.1). In the first place, there is a *data transformation* (or data preprocessing) process to extract relevant data from multiple data sources [31, 44]. In preparation for subsequent steps, data quality problems are resolved at this point by applying data cleaning approaches [158]. *Schema matching* is the following process to discover identical structural entities for multiple, heterogeneous data sources (see [157] for a survey). For example, the column names from Table 1.1a can be mapped to their equivalent in Table 1.1b, such as (ID $\leftrightarrow$ ID), (name + surname $\leftrightarrow$ label) and so on. After integrating the schema information, the actual records from the data sources are aligned in a process called *data matching* (also instance matching or entity resolution) to find equal real-world entities and to classify them accordingly [31]. The correct alignment of entities generally involves the comparison of relevant attribute values; in Table 1.1, this leads to the matching entities (A1 $\leftrightarrow$ B0) and (A2 $\leftrightarrow$ B1). The final task of *data fusion* merges matching entities into a single and unified entity representation [31].

1.1.2 Semantic Web

The complex problem of data integration has been of interest to researchers for many decades. In other research areas, data is also combined using data integration processes (e. g., see Figure 1.1). In this section, we show the application of the data integration process to support the development of the Semantic Web – a supplement to the existing World Wide Web.

One of the main reasons for the success of the World Wide Web in the 1990s is the usage of hyperlinks to connect web documents. Additional structured features for web documents (metadata), such as information on authorship as well as technical information like date of creation or expiry date, have always been an integral part of the Web [14]. Through the use of new technologies and the establishment of standards, the World Wide Web has since been continually adapted to the changing requirements and needs of users. Instead of static websites, dynamic content is used. Search engines have replaced web directories for a long time, and communication and collaboration take place in social networks and online encyclopedias. Both consumed as well as produced data have become more and more heterogeneous and extensive, such as text, sound and video data from interconnected devices. To counter this increased complexity and as a complement to the classical Web of *Documents*, the Web of *Data* (also Semantic Web) was initiated by the W3C to annotate data with useful metadata and make it machine-readable [187]. Corresponding description languages such as RDF, XML or OWL aim at ensuring that semantic connections can be derived from the data and that data can be linked together.

Data published in compliance with Semantic Web specifications, such as machine readability and use of open formats, is referred to as Linked Data [19]. If the data is also available under the public domain or an open license [150], the term Linked Open Data is common. To realize the principles of Linked Data, adding links to data sources is fundamental. Links are semantic relations that can be identified across entities, e. g., equivalence or part-of relationship. These links can exist both within data sources and between different data sources. The visualization of the resulting interlinked data sources results in a graph structure – the Linked Open Data Cloud[1]. By focusing on the links between the data sources, the main objective of Linked Data is pointed out. New links between entities are intended to make information more accessible across data sources. Compared to data matching, where equivalence between objects is typically determined, Linked Data supports arbitrary semantic relations. Searching for these semantic relations is called link discovery. Link discovery focuses on the creation of binary links between two data sources with efficient and effective methods (see [132] for a survey).

[1]Linked Open Data Cloud `https://lod-cloud.net/`

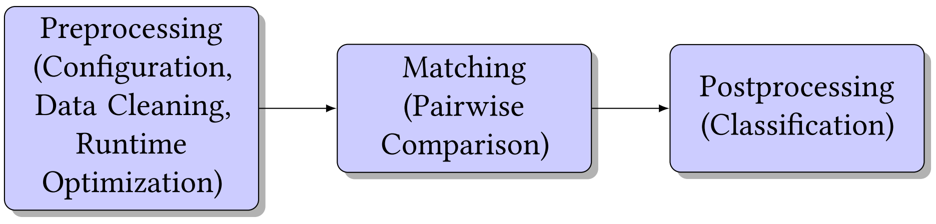

Preprocessing (Configuration, Data Cleaning, Runtime Optimization)
Matching (Pairwise Comparison)
Postprocessing (Classification)

Figure 1.2: General link discovery workflow.

A generalized workflow for link discovery with the three typical components is shown in Figure 1.2. *Preprocessing* handles data problems before the matching step. The configuration specifies which attributes should be compared in the matching step. Good data quality helps to achieve good matching results; therefore, the use of data cleaning methods can be useful [158]. Runtime optimization is the critical step in preprocessing. The aim is to avoid as many comparisons between dissimilar entities as possible, e. g., by applying filtering or blocking methods [31, 156]. Filtering relies on early pruning of dissimilar entities, e. g., by using indices. In contrast, blocking splits the data into several partitions and compares all entities within these partitions. The *matching* step performs the pairwise comparison of the entities based on the defined configuration. More elaborate configurations can also define how different attribute similarities are combined. This may be necessary if individual attributes are not sufficient for a distinction. The similarity values from the matching step are used as decision support for a classification problem. This classification is part of the *postprocessing*, along with an optional refinement of the results. For example, to achieve high quality, a classification can include that threshold values for similarities must be exceeded. The resulting set of links, according to the semantic relation, forms a so-called mapping.

These mappings can either be integrated into existing knowledge bases or made available for further use in portals, such as Datahub[2] or BioPortal[3]. Thus, existing links can be reused as an essential basis for data matching to avoid time-consuming complete recomputations. Link portals can also store additional provenance information related to mappings. Using meta information, such as creation date or matching strategy, can help identify falsely created links as well as inconsistent data. As the number of data sources in the Web of Data increases, high connectivity between the data sources becomes more laborious. Therefore, new sources calculate links to selected Linked Data sources. As a result, some data sources are only indirectly connected. By reusing links, equivalent entities over multiple data sources can be discovered despite the indirect connection [76].

[2]Datahub platform https://datahub.io/
[3]BioPortal platform https://bioportal.bioontology.org/

To enable the physical integration of entities into knowledge graphs, the determination of binary links from the link discovery is not sufficient. Instead of mappings with binary links, related entities can be clustered as an alternative result representation. These clusters can then combine the information of the contained entities as a form of physical data integration (data fusion). The link discovery should therefore be complemented by a clustering process that simplifies the creation of integrated knowledge bases for multiple data sources. After the creation of clusters, particularly large clusters point to well-linked entities. The information given by these consolidated clusters can thus be significant for further analysis or improvement of quality.

1.1.3 Big Data

Data integration plays a vital role in many research areas, including the Semantic Web. Due to semantic links, the addressing of heterogeneity between data sources becomes increasingly relevant. With the current trend towards digitization in many areas of life, it is possible to link many more data sources. Interlinked data sources promise new opportunities for data analysis and decision making. For this purpose, ever-larger amounts of heterogeneous data are accumulated and often summarized under the term Big Data. A description of the ”V” dimensions of Big Data clarifies the differences to previous data storage, processing and analysis [45]. This leads to the introduction of methods and techniques to perform data integration tasks on Big Data.

A study by the market research company IDC in 2018 [160] shows that data growth continues to increase strongly. The worldwide data volume increased from less than $5\,\mathrm{ZB}$[4] in 2012 to already $33\,\mathrm{ZB}$ in 2019. By 2025, according to estimates, $175\,\mathrm{ZB}$ must be stored and made usable. Thus, the ability to handle massively growing amounts of data (*volume*) in both individual source systems and across a vast number of systems is a significant task. Since many new data-generating systems produce unstructured data, the inherent heterogeneity (*variety*) also increases. With the term *velocity*, both the increased speed of data generation as well as the increased speed of data processing is described. *Veracity* is the fourth component often used to describe Big Data, along with volume, variety, and velocity. It describes the necessity to ensure the quality of the data as much as possible and is directly related to the three V’s mentioned above [197]. Achieving high data quality is a complex task [158], even more with an increasing number of heterogeneous data sources that are subject to continuous change.

Various frameworks in the Big Data ecosystem improve the handling of enormous amounts of data. In this context, Apache Hadoop[5] and the associated platform was a

[4] 1 ZB (Zettabyte)$= 1 \times 10^9$ TB (Terabyte).
[5] Apache Hadoop `https://hadoop.apache.org/`

significant development. Hadoop implements the programming model MapReduce [39], which enables the distributed execution of algorithms on large amounts of data. The functions *Map* and *Reduce* allow the expression of user-defined program logic. *Map* creates key/value pairs using the data values on each partition, which are then aggregated according to the resp. key using the *Reduce* function. Besides the MapReduce implementation, the Hadoop framework contains a collection of open-source software libraries, such as HDFS or YARN, to simplify the storage and processing of Big Data. Hadoop abstracts all details of data distribution and load balancing in complex computations and ensures reliable execution even in the event of hardware errors. The free availability of the software means that all relevant cloud providers offer Hadoop solutions for executing compute- and data-intensive workflows on computer systems of any size. However, disadvantages of MapReduce, such as the materialization of intermediate results [112], led to the development of new distributed processing systems. These systems are either specialized for applications like data warehousing or data analytics, or general-purpose processing platforms with additional support for, e. g., stream processing or graph processing. For the latter, Apache Spark [194] and Apache Flink [27] are relevant systems to enable the execution of arbitrary user-defined programs.

1.1.4 Challenges and Requirements

The thematically broad introduction outlines the need for optimized data integration processes for Linked Data. The processing and analysis of growing amounts of heterogeneous data can benefit from findings and techniques of the Big Data environment. Consequently, this dissertation deals with solutions for the integration of different Linked Data sources using Big Data technologies.

A central component of many data integration processes is data matching, which identifies equivalent entities between data sources. For Linked Data, link discovery can be used to determine equivalence relations as well as any other semantic relations. The determination of new links is of central importance for Linked Data and many other applications in the Semantic Web. For this reason, this thesis focuses on the application of Big Data frameworks to make the link discovery process more efficient and effective. At the same time, we investigate how Big Data frameworks can be used beyond link discovery for the associated data integration tasks. Due to the preliminary discussion, the following important requirements for data integration in the Semantic Web should be adhered to.

Reuse of Existing Data Matching Results A major strength of the Semantic Web is the existence of semantic links between data sources. These links can be valuable for data integration in general and data matching in particular. Approaches for determining new links should therefore offer methods that support the reuse of existing links. For example, the transitivity of indirectly linked entities can be used to determine new links. The benefit of reuse-based methods should be evaluated based on qualitative metrics.

To enable the reuse of links in the first place, the availability of links must be increased. Public data portals should be used to allow researchers to exchange results. Provenance information related to the creation process, such as date, methods used, and resulting similarity values, should also be stored. Thus, the origin of links can be used in future linking processes, e. g., by inferring new statements from existing data with logical reasoning.

Holistic Clustering and Support for Dynamic Data Sources Previous link discovery approaches mostly perform pairwise comparisons of data sources. When entities from multiple data sources are compared, the number of comparisons rises significantly. Consequently, there is a strong need for efficient and effective holistic approaches to link multiple data sources. For this purpose, methods should combine attribute values of equivalent entities from different data sources. The aim is to create a holistic cluster with unified attributes that can be enriched with metadata from the data sources. The cluster representation gives the possibility for central maintenance for each real-world object. Also, the reuse of existing links should be possible for holistic methods. This makes it possible to use attribute values as well as semantic types to remove incorrect links and to determine links that have been missing so far.

Furthermore, approaches should offer a convenient way to add new entities or data sources to existing clustered results. Thus, a continuous adjustment of dynamic data sources should be made possible without the need to recalculate all mappings between sources.

Scalable and Distributed Processing of Data Integration Workflows The scalability of linking approaches becomes an important problem with the increasing number of data sources and entities per source. For an efficient data integration process that is scalable to large amounts of data, two conditions have to be satisfied. First, by using Big Data frameworks, computation workload can be distributed on a cluster of machines for parallel processing. These distributed processing frameworks should be used to perform efficient linking and clustering. Secondly, runtime optimizations should be implemented to limit the number of comparisons between dissimilar entities. The partitioning of the data into disjoint blocks results in synergy effects combined with the distributed processing framework. These effects should be used within the linking and clustering approaches.

1.2 Scientific Contributions

The initial analysis of challenges for data integration within Linked Data led to the following scientific contributions. All contributions are both peer-reviewed and published in journals or proceedings of conferences resp. workshops.

Comparison of Current Link Discovery Approaches State-of-the-art link discovery systems are categorized by common requirements such as effectiveness and efficiency to assess strengths and weaknesses. Based on these findings, we derive a generic architecture for the link discovery process. The background for this analysis is a survey article accepted for publication in the Semantic Web Journal in 2015 and then published in 2017 [132]. In the context of this dissertation, a limitation to four tools takes place to represent a current state of development. For these tools, an overview of functional features is provided, e. g., configuration approach, runtime optimization and support for parallel processing.

Platform for Reuse of Links and Provenance The free and open-source web portal LinkLion increases the visibility for link discovery results and enables researchers to share computed mappings. Alternatively, existing mappings can be reused and analyzed. Ontology vocabulary is used to facilitate the handling of provenance information. For new mappings, researchers can add information such as the used algorithm and link discovery system. Besides viewing links, mappings and metadata on the portal relevant data can be acquired by querying via SPARQL or REST API, or directly by downloading the raw files. The web portal and its functionality were presented in a practical demonstration at the ESWC 2014 and published as part of the corresponding conference proceedings [134].

Holistic Entity Clustering Link discovery approaches typically compare entities from two data sources. When integrating many data sources, this process is difficult to apply, thus limiting the scalability of link discovery. We therefore propose Split-Merge as a holistic approach for linking many data sources to integrate equivalent real-world objects into clusters. By efficiently using existing match results, new links can be found while correcting erroneous links at the same time. The approach was accepted for presentation at the DINA workshop in 2016 and subsequently published as part of the ICDM 2016 workshop proceedings [131].

Distributed Clustering for Linked Data The parallel execution of linking workflows promises increased effectiveness using state-of-the-art Big Data frameworks. For this reason, we propose an implementation of the holistic entity clustering on the distributed processing system Apache Flink and show the scalability of the approach. The implementation shows that efficiency can be further increased by blocking strategies, compliance with consistency rules and exploitation of data properties. The presentation of the approach took place at the ODBASE conference in 2017 and was published in the respective conference proceedings of the OTM 2017 [130].

Comparative Evaluation of Scalable Clustering Approaches Integrating many data sources to create clustered objects is especially challenging for Big Data applications. For this reason, we compare various clustering approaches for Apache Flink, which use holistic methods to combine entities from different data sources. A uniform workflow allows a comparative evaluation of the presented approaches across different domains. The journal article was published in CSIMQ in 2017 [162].

Incremental Entity Clustering For the handling of dynamically growing Linked Data sources, we present clustering methods for the incremental extension of existing knowledge bases without completely recalculating the results. The clustering can manage completely new data sources as well as new entities from multiple sources. As a result, entities can be added to existing clusters or form new clusters. To achieve scalability for large amounts of data, the distributed data processing system Apache Flink abstracts the actual parallelization. A comparison with current static clustering methods complements the study on incremental clustering. Evaluation results show scalability and high quality of the resulting clusters for different domains. The incremental clustering concept was presented at the DINA workshop in 2018 and published in the ICDM workshop proceedings [133].

Resulting Software Systems and Open Source Implementations The approaches for holistic and incremental clustering are combined into a single data integration system for multiple Linked Data sources. In addition to the central data matching step with link discovery and clustering, preprocessing (data cleaning) and post-processing (data fusion) is also performed. All work steps of the data integration process are performed in the distributed processing system Apache Flink.

All implementations of the proposed approaches in this dissertation are available as open-source software on GitHub, e. g., the techniques for scalable holistic and incremental entity clustering[6] and the LinkLion repository[7]. In addition, the LinkLion web portal is available for the reuse of link discovery results[8].

1.3 Structure of Thesis

This dissertation consists of three main parts. The introductory Part I contains two further chapters:

Chapter 2 gives an introduction to the topic of Semantic Web. For this purpose, the most important topical areas and the corresponding technical structure are presented, which are used to implement applications and research activities. With these technological basics, the data model is defined for application in the further course of the work.

Chapter 3 places a special focus on the topic of link discovery as a central element of the Semantic Web to pursue the idea of Linked Data. Based on the requirements for a reliable link discovery process, we present a generalized workflow. Finally, the comparison of relevant approaches for link discovery is carried out.

Part II – Reuse of Link Discovery Results and Clustering Strategies – presents strategies to increase the efficiency and effectiveness of link discovery procedures.

Chapter 4 describes the LinkLion data portal for storing and maintaining link discovery results. The published data can be enriched with provenance information to enable continued analysis on the links. The permanent online access allows the use of the built-in Linked Data browser as well as SPARQL queries and download of raw data.

Chapter 5 presents a holistic approach for clustering entities from many data sources. Additional relationships between entities can be identified and at the same time erroneous data can be detected. In detail, clusters are refined by split and merge operations. The evaluation uses a real dataset from the geographical domain.

[6]GitHub Clustering https://github.com/freeclimbing/mapping-analysis
[7]GitHub LinkLion https://github.com/AKSW/LinkLion
[8]LinkLion Portal http://www.linklion.org/portal/

Chapter 6 complements holistic clustering with two incremental approaches to process dynamically changing data sources. The approaches distinguishes a set-based method for adding entities from different sources and a source-specific method. In addition to a comparison of the incremental methods, an evaluation against typical static clustering approaches is carried out.

Part III deals with the Parallelization of Clustering Strategies.

Chapter 7 begins with a brief outline of the development of distributed data processing frameworks to capture basic concepts. These lead to a detailed discussion of state of the art and it's derived features. For the following chapters, the distributed processing framework Apache Flink is used due to good support for graph algorithms and iterative computations.

Chapter 8 deals with distributed data processing to implement the holistic clustering of multiple data sources on very large datasets. We achieve an efficient cluster determination by applying abstract programming models provided by Apache Flink. The evaluation shows scalability and high-quality results for datasets from geography and music.

Chapter 9 shows that it is possible to compare different clustering strategies on distributed data processing systems. Due to very large data volumes, this comparison is elaborate. The presented FAMER system allows a comparison of approaches based on a detailed evaluation for datasets of variable size. With this knowledge, we provide an assessment of the efficiency of the investigated approaches.

The final Part IV – Conclusion and Outlook – then concludes the results of this dissertation and provides an outlook on future work.

2 Background

2.1 Semantic Web

As an extension to the Web of Documents, the idea of a *Semantic Web* was postulated by Tim Berners-Lee et al. in 2001 [15]. The presented vision shows how computer systems automatically process Web content to assist users in specific tasks. As an example, the system sets up doctoral appointments in accordance with the personal calendar and checks facts for complex questions. In this context, the integration of different, heterogeneous knowledge repositories is possible by using semantic data descriptions. As a result, they make it feasible to show correlations between the data. Since then, standards for the Semantic Web, such as RDF Schema, RDF, OWL and SPARQL, have been published and are used for a plethora of applications and research activities [17][1].

2.1.1 Topical Main Areas

The following classification is given by the W3C [85] and shows all relevant areas of the Semantic Web with the appropriate description. Typically, applications as well as research activities can be categorized into one or more areas of the classification.

[1]Extended timeline up to 2016: http://www.dblab.ntua.gr/~bikakis/XML%20and%20Semantic%20Web%20W3C%20Standards%20Timeline-History.pdf

Linked Data

The idea to link web data in a way that machines can exploit the knowledge automatically was formulated in 2006 by Berners-Lee [13]. Together with the idea, he outlines four principles to create the so-called *Linked Data* based on related “things” – expressing any real objects like Leipzig[2] or abstract concepts like calculus[3] [13]:

1. “Use URIs as names for things.”
2. “Use HTTP URIs so that people can look up those names.”
3. “When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL).”
4. “Include links to other URIs. so that they can discover more things.”

According to these principles and driven by the idea to interlink data, the Linking Open Data project [186] promoted the publication of datasets for public use. Up to June 2018, researchers and contributors published 1234 datasets based on the Linked Open Data principles (see Figure 2.1). The resulting graph is called Linked Open Data Cloud[4] and consists of interconnected data sources of different domains such as publications, life sciences, government or cross-domain. As an example, open government initiatives improve the availability and usability of governmental and administration data [79] by providing Open Data platforms[5][6][7].

Vocabularies

Vocabularies (also called ontologies) are used to provide a structure for data in the Semantic Web, e. g., to define relationships or properties of objects. According to Gruber [70], the term ontologies defines a set of basic types of objects to model a knowledge domain, e. g., classes, their relationships and attributes. Also, the basic types allow logical constraints to be added to provide a consistent overall representation.

For this purpose, we give an example from the geographical domain, a route description from one place to another. Diverse classes of objects such as highways, points of interest or forms of land use together with their properties form an ontology and are used

[2]DBpedia Leipzig `http://dbpedia.org/page/Leipzig`
[3]DBpedia Calculus `http://dbpedia.org/page/Calculus`
[4]Linked Open Data Cloud `https://lod-cloud.net/`
[5]Open Data platform USA `https://www.data.gov/`
[6]Open Data platform UK `https://data.gov.uk/`
[7]Open Data platform Germany `https://www.govdata.de/`

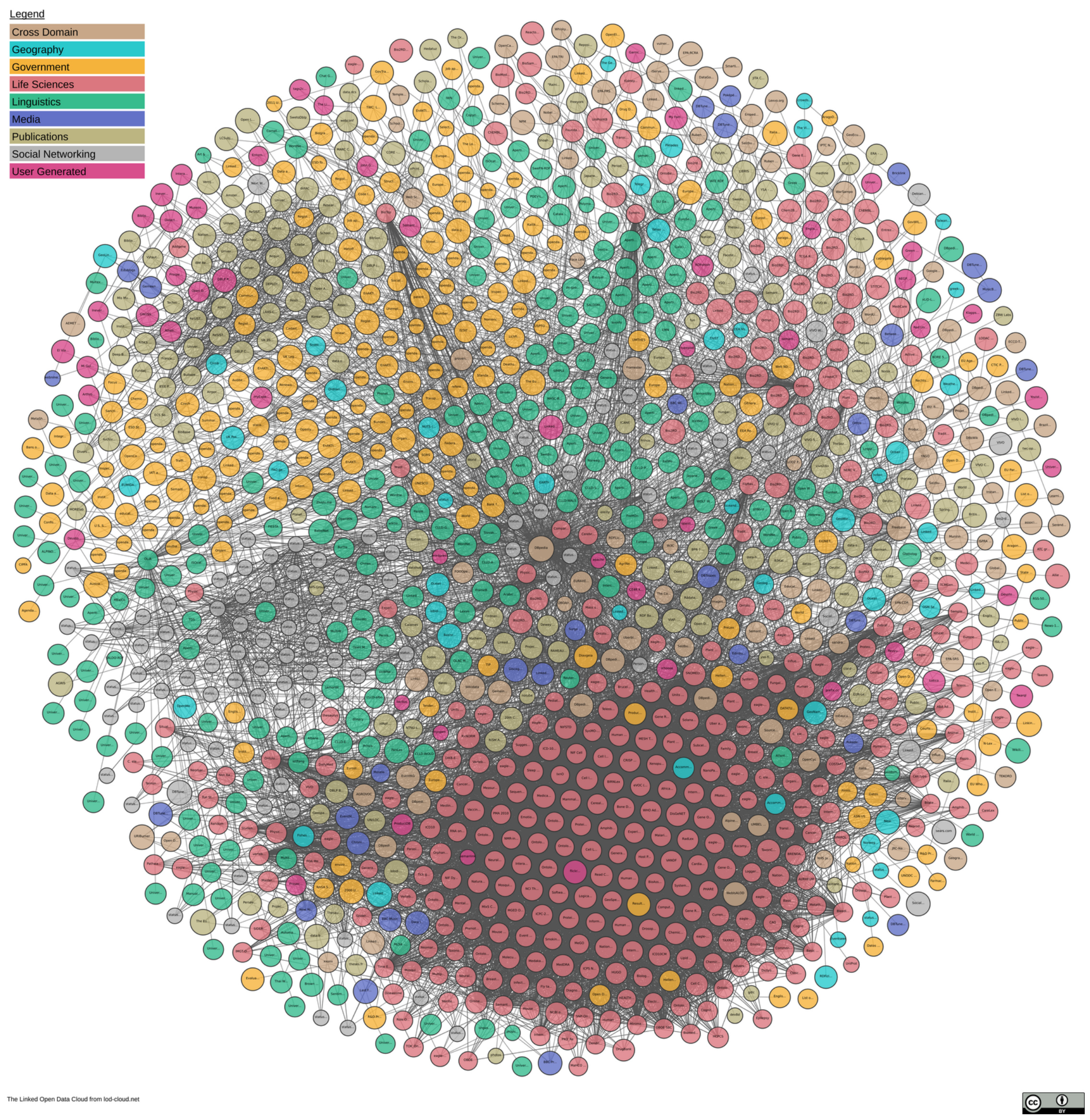


Figure 2.1: Linked Open Data cloud (Source: `https://lod-cloud.net/`).

to find a suitable route [34]. The classes simplify the route calculation and allow compliance with modeled constraints, such as avoiding residential areas on long distances or respecting turning restrictions at intersections.

Due to this fundamental modeling, ontologies are essential for the integration of data sources in the Semantic Web. The term ontology is partly distinguished from the term vocabulary when a large number of complex interconnected classes are paired with formal or logical constraints [70, 85]. The integration of knowledge bases in the life sciences is an example of successful data integration using ontologies. Thus, it is possible to link the side effects of drugs (e. g., SIDER [111]) to general medical terms (e. g., UMLS [20]). As a result, side effects and medical terms can be associated with individual patient data so that medical staff can obtain additional information for treatment [22, 174]. The ontologies RDF, RDFS and OWL (details in Section 2.1.2) provide fundamental concepts for the description of further ontologies, enabling the creation of new vocabularies consistently.

Query

Programmatic mechanisms are used to access data in the Semantic Web, which can be accessed via the SPARQL query language (similar to SQL for relational databases or XQuery for XML). SPARQL employs RDF vocabulary and variables to find triple patterns. The combination of queries from several data sources is possible as well as different result types, e. g., as table format or as RDF [85]. More details on SPARQL can be found in the corresponding Section 2.1.2, which describes the components of the Semantic Web technology stack.

Inference

The term inference is strongly connected with the formal modeling that takes place in ontologies. Resources can implicitly utilize properties based on the affiliation to certain ontology concepts [85]. For instance, given the ontology concepts ’village’ and ’city’, it can be derived that both are a sub-concept of ’settlement’. This assignment can be used to derive further attributes, such as population numbers or the associated country.

This process of deducing new relations from the data is called inference or reasoning. Logical expressions are defined by an ontology using a formal knowledge representation language (OWL) or rule-based mechanisms (RIF, see following Section 2.1.2). Depending on the application, the logical expressions are then evaluated using automatic methods [159]. Besides the objective of deducing new relations, the verification of logical integrity is essential. The verification ensures consistency according to logical expressions for existing as well as new data.

Vertical Applications

Vertical applications describe generic application scenarios that are examined concerning possible purposes and potential problems when using Semantic Web technologies. Generally, domain-specific technologies from the previously mentioned four areas are involved. During implementation, the users give detailed feedback to develop semantic technologies further [85]. A positive example of this is “The Health Care and Life

Science Interest Group"[8]. They investigated the usability of Semantic Web technologies for drug discovery, patient care management and reporting as well as the publication of scientific knowledge from 2005 to 2018. The results of this interest group have led to new standards and technologies jointly developed and used by universities and research companies alike[9].

2.1.2 Technology Stack

The Semantic Web technology stack (also called layer cake) is illustrated in Figure 2.2. It integrates relevant technology blocks to form a single model that can be used to implement semantic user interfaces and applications [85]. The importance of each building block varies for different areas of application (see previous Section 2.1.1). For instance, SPARQL is crucial for performing queries on Linked Data, whereas OWL and RIF are more appropriate for determining logical rules to perform inference. In this section, relevant technology blocks are discussed in more detail.

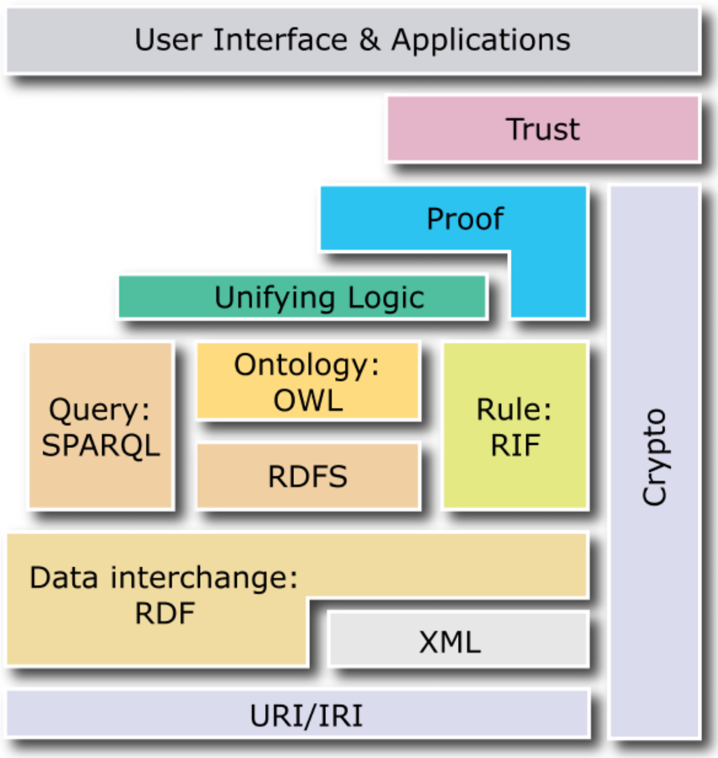


Figure 2.2: W3C Semantic Web technology stack, showing how the technology layers build on each other (Source: [189]).

[8]The Health Care and Life Science Interest Group https://www.w3.org/2001/sw/hcls/

[9]W3C Semantic Web Vertical Applications https://www.w3.org/standards/semanticweb/applications

Resource Identifiers

The application of unique identifiers allows efficient operations on individual data records. In relational databases, e. g., a unique candidate key is defined for each table, often an automatically generated unique number (ID). Instead, the Semantic Web uses existing Web standards, namely RFC 3987, to describe Internationalized Resource Identifiers (IRIs) [48]. As an extension to the existing URI standard [12], IRI allows a broader range of characters from the Unicode character set.

In other respects, IRIs have the same basic characteristics as URIs. Accordingly, the properties of URIs (Uniform Resource Identifiers) are described hereafter. A generic URI syntax achieves the *Uniformity* property with certain mandatory and optional components. Using `http://dbpedia.org/resource/Leipzig` as an example, typical components are shown without claiming complete representation of the standard: The schema name '`http`' is followed by the authority '`dbpedia.org`' with the concluding (optional) path '`/resource/Leipzig`'. Both URI and IRI describe *resources*, which are generally used for arbitrary objects to be described, e. g., electronic documents and web services to abstract or real-world objects as well as persons. Finally, the *identifier* describes the property that the URI contains a unique expression to distinguish the resource from other resources.

Resource Description Framework (RDF)

The Resource Description Framework (RDF) [37] describes an abstract data model to express information on the Web. The W3C recommendation for RDF describes the design goals as follows. Based on a simple data model with formal semantics and provable inference, statements should be possible about any resources. Therefore, URI-based vocabulary, XML-based syntax and XML schema datatypes should be supported [37]. The RDF model is the common basis for many Semantic Web technologies. Various forms and dialects express different data serialization formats for storing or exchanging RDF, such as Turtle[10] or JSON-LD[11], but also the query language SPARQL and the schema description language RDFS (see following sections).

The basic form of expression in RDF is a triple consisting of *<subject, predicate, object>*. The so-called statement is reflecting a relationship (given by the predicate) between the subject and the object. Triples, as well as collections of triples, form an RDF graph; subjects and objects are considered as graph vertices and predicates as graph edges [37]. In the remainder of the work, the Turtle serialization format is used to provide easy-to-read

[10]RDF 1.1 Turtle `https://www.w3.org/TR/turtle/`
[11]JSON-LD 1.0 `https://www.w3.org/TR/json-ld/`

examples for statements about things, e. g., by using prefix definitions as abbreviations for URIs. For the rest of the work, uniform prefixes are obtained via a lookup service[12]. A simple example of an RDF graph is shown in Figure 2.3 by expressing the birthplace of philosopher Leibniz. Besides the IRIs used here for subject, predicate and object, literals or blank nodes can also be used as so-called RDF terms. Literals contain textual information such as strings, numbers or dates values attached with a datatype. Blank nodes are disjoint from IRIs and literals and are used as intermediate nodes in a local scope when no unique identifier via IRI or literal is wanted (or possible). There are certain restrictions for placing RDF terms in statements. As Figure 2.3 shows, all positions can be IRIs, but only subjects or objects can be blank nodes and only objects qualify as a position for literals. IRIs and literals refer to resources as they represent a tangible entity in the world. Compared to literals, IRIs have a global scope and the same IRI always refers to the same resource. Typically IRIs can be dereferenced and are thus a good starting point for web interaction [37].

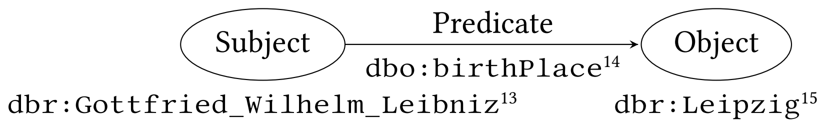


Figure 2.3: Basic example for a single RDF Triple having subject, predicate and object.

The concept reification is used to add new information or statements to existing RDF triples with the appropriate vocabulary[16]. The non-normative description for reification in the RDF standard suggests the use of a new IRI to link the statements together. As an example, a single statement about the birthplace of Leibniz is reified. For this purpose, the new IRI `ex:leibniz` is used to form the following four reified statements.

```
ex:leibniz  rdf:type  rdf:Statement .
ex:leibniz  rdf:subject  dbr:Gottfried_Wilhelm_Leibniz .
ex:leibniz  rdf:predicate  dbo:birthPlace .
ex:leibniz  rdf:object  dbr:Leipzig .
```

With this description of the RDF triple, it is now possible to make statements regarding the triple, e. g., the journalistic source:

```
ex:leibniz  prov:wasGeneratedBy  "Wikipedia".
```

[12]Namespace lookup http://prefix.cc
[13]DBpedia Leibniz http://dbpedia.org/page/Gottfried_Wilhelm_Leibniz
[14]DBpedia ontology birthPlace http://dbpedia.org/ontology/birthPlace
[15]DBpedia Leipzig http://dbpedia.org/resource/Leipzig
[16]RDF 1.1 Semantics https://www.w3.org/TR/rdf11-mt/

In addition to the standard reification, alternative proposals for reification exist [75, 80]. The authors of [80] present multiple methods to attach additional information to existing statements with reification, e. g., to include details on provenance. The proposal of Hartig et al. [75] is an interesting alternative to embed existing RDF triples in new statements. For our example, this leads to the following representation:

```
<<dbr:Gottfried_Wilhelm_Leibniz  dbo:birthPlace  dbr:
  ↪ Leipzig>>  prov:wasGeneratedBy  "Wikipedia".
```

SPARQL

The query language SPARQL (SPARQL Protocol and RDF Query Language) allows structured reading and manipulating queries on RDF data as well as result presentation. A SPARQL endpoint processes these queries according to the SPARQL 1.1 Protocol[17]. Endpoints access the specified dataset and return the results matching the query. Therefore, SPARQL endpoints are the interface between the requesting side and the data store containing the RDF data. An overview of typical data stores with SPARQL support [118] compares multiple options, which rely on either relational or graph database stores.

The required syntax and semantics for the query language are specified in the W3C SPARQL 1.1 Overview standard [178]. Since the extensive SPARQL standard is not entirely relevant for this work, we limit ourselves to a detailed description of the relevant parts, especially the language specification. Basic terms and operators are defined in SPARQL 1.1 Query Language[18]. In principle, the general syntax of a query q is always a quintuple of q(query type qt, variable selection vs, dataset clause dc, where clause wc, solution modifiers sm)[19]. Table 2.1 shows the characteristics and relevant restrictions. The table demonstrates that the dataset clause dc always points to an RDF dataset IRI (when omitted, the default graph is used). Likewise, the where clause wc always needs a graph pattern on which the resulting triples must match. A graph pattern in SPARQL can be defined in different ways, e. g., by combining triple patterns, value constraints and optional graph patterns or by joining the results of multiple graph patterns. A triple pattern is then again an RDF statement[20] where subject, predicate and object can be replaced by query variables to specify which information should be retrieved from the actual RDF dataset. A simple graph pattern is realized in the example SELECT query

[17]SPARQL 1.1 Protocol https://www.w3.org/TR/sparql11-protocol/

[18]SPARQL 1.1 Query Language https://www.w3.org/TR/sparql11-query/

[19]SPARQL 1.1 Query Language 19.8 Grammar

[20]According to the SPARQL standard, the subject of the triple pattern could also be a literal. Practically, triple stores that allow SPARQL queries do not allow literals as subject to correctly implement the RDF standard. It follows that a triple pattern with literal as subject would never return a result.

Table 2.1: Possible SPARQL query forms to retrieve result sets or RDF graphs from RDF datasets according to the general SPARQL syntax.

Query Type (qt)	Variable/IRI selection (vs)	Dataset Clause (dc)	Where Clause (wc)	Solution Modifiers (sm)	resulting in
SELECT	list of variables	RDF Dataset IRI	graph pattern	order, limit, offset	table result set
CONSTRUCT	triple pattern	RDF Dataset IRI	graph pattern	order, limit, offset	graph
DESCRIBE	list of variables/IRIs	RDF Dataset IRI	graph pattern	order, limit, offset	graph
ASK	-	RDF Dataset IRI	graph pattern	order, limit, offset	boolean

(shown in Listing 2.1) by specifying two triple patterns and two variables (`?idea` and `?label`) in combination with the value constraint to the language `"de"`. Furthermore, the result set can be restricted or ordered by solution modifier sm, e. g., shown in Listing 2.1 for order and limit.

Listing 2.1: Example query to show general syntax of SPARQL language, retrieve german labels for notable ideas of Leibniz

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?idea ?label
FROM <http://dbpedia.org>
WHERE {
  dbr:Gottfried_Wilhelm_Leibniz dbo:notableIdea ?idea .
  ?notableIdea rdfs:label ?label .
  FILTER (lang(?label) = 'de')
}
ORDER BY ?idea
LIMIT 5
```

Depending on the selected query type qt, input parameters vs are selected. While SELECT and DESCRIBE require a list of query variables (optionally, DESCRIBE needs a list of IRIs), CONSTRUCT requires a triple pattern. The query type selection also determines the result format, e. g., Table 2.2 shows the SQL-like table result for the SELECT query. For the sake of clarity, the result set is limited to five result tuples in

Table 2.2: Resulting table for example SELECT SPARQL query.

idea	german label
dbr:Alternating_series_test	”Leibniz-Kriterium”@de
dbr:Best_of_all_possible_worlds	”Die beste aller möglichen Welten”@de
dbr:Calculus	”Infinitesimalrechnung”@de
dbr:Characteristica_universalis	”Universalsprache”@de
dbr:Fermat’s_little_theorem	”Kleiner fermatscher Satz”@de

the query. In contrast, CONSTRUCT queries return a newly created RDF graph based on the given pattern and DESCRIBE returns an RDF document whose structure is determined by the SPARQL query processor. Finally, ASK can be used if a binary yes/no response is sufficient.

In addition to these query forms, combined sub-queries can also be formulated, and typical grouping and aggregate functions can also be expressed. Even more complex analytical questions can be answered with SPARQL 1.1, e. g., the search for arbitrary graph patterns in RDF graphs. The SPARQL standard is completed by miscellaneous extensions, some of which are still under development. Possible extensions are the update operator to support manipulating RDF data[21], federated queries over different SPARQL endpoints[22] or automatic detection of SPARQL services via HTTP [178].

RDF Schema and Web Ontology Language

Both RDF Schema (RDFS) [23] and the Web Ontology Language (OWL) [188] are semantic extensions for RDF by using the underlying data model with its fundamental components. RDFS provides a vocabulary that can be used to describe properties of classes, entity sets and the relations between them. Thus RDFS offers the possibility to specify simple ontologies such as FOAF[23] or Schema.org[24]. Both RDFS and OWL can be used to publish semantic schema information. With OWL, a formal description language for the Semantic Web, another set of basic terms for the creation of ontologies is defined. OWL supplements existing constructs that are specified by RDF and RDF schema and uses the expressiveness of RDF. Information specified in OWL is also valid RDF.

In the area of Linked Data, the description of ontologies using RDFS or OWL implies the structured description of things, groups of things and relations between them. In this context, things described with logical statements are restricted or choices are given,

[21]SPARQL 1.1 Update https://www.w3.org/TR/sparql11-update/
[22]SPARQL 1.1 Federated Query http://www.w3.org/TR/sparql11-federated-query/
[23]FOAF Vocabulary Specification 0.99 http://xmlns.com/foaf/spec/
[24]Schema.org core schema https://schema.org/docs/schema_org_rdfa.html

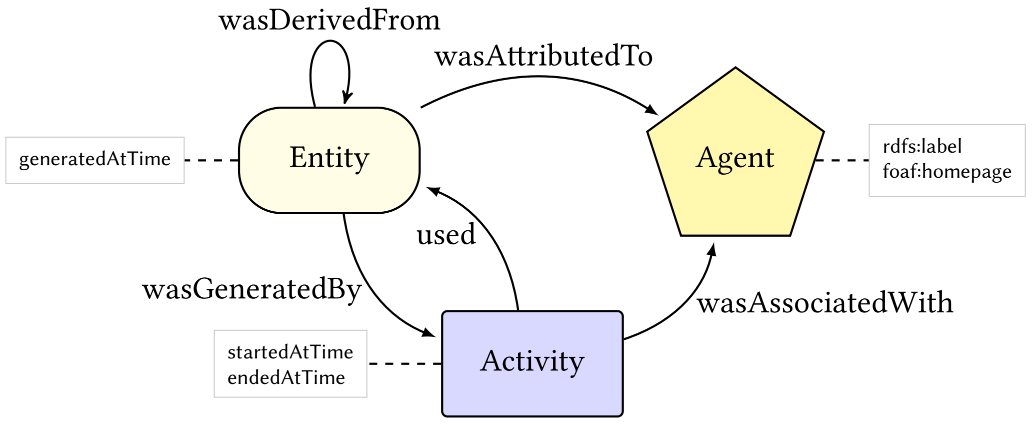


Figure 2.4: High level overview on PROV records and how they are linked to each other and some possible attributes (grey boxes). Concepts and properties without prefix are part of the PROV vocabulary. Figure modified based on [10].

e. g., a person who has the attribute male as gender cannot be female at the same time. These formalized and structured terms can then be used to gain new insights by logical inference (reasoning), using description logic (DL) in different expressions depending on the purpose. Also, the check of logical statements for consistency can be realized with OWL-based vocabulary[25].

RIF

The Rule Interchange Format (RIF) enables declarative rules and their exchange between different rule systems, e. g., with RDF and OWL ontologies [102]. RIF is not relevant in this thesis and will not be elaborated.

Trust, Logic and Unifying Logic

The complex of trust and proof deals with the question of how trustworthy statements in the Semantic Web are. As with previously described technology blocks such as OWL, where underlying layers (such as RDF, IRI) are used, trust and proof also rely on those layers. Both the layer trust and proof require further standardization work. However, with the PROV Standard [69], there are already approaches to facilitate the traceability of decisions. Additional information about entities, activities, people and interactions between those can be stored together with the actual data to understand the origin of information [10]. A basic example of interacting PROV concepts together with properties such as date/time of generation is provided in Figure 2.4. The use of PROV allows changes to the data to be tracked more easily, e. g., by providing the inference rule for

[25]OWL 2 Web Ontology Language Primer (Second Edition) https://www.w3.org/TR/2012/REC-owl2-primer-20121211/

adding new relations. Together with (not yet standardized) cryptography techniques, e. g., for authentication or non-deniability of events, it is assumed that the establishment of trust in the Semantic Web is feasible [126]. For further work, we neglect components like cryptography and trust and limit ourselves to the logging of available provenance information.

2.1.3 Conclusion and Distinction to this Thesis

In the last two sections, the description of the main application areas of the Semantic Web leads to the introduction of associated technology blocks. Since the application areas are typically data-driven, Linked Data is of particular importance. The objective for existing as well as new Linked Data sources is the creation of additional semantic links to increase the usability of interlinked datasets. To express this relevance, Auer et al. [7] describe an extensive Linked Data life cycle, as shown in Figure 2.5.

Besides techniques for *storing* and *querying* RDF data [56], various collaboration and networking techniques [5, 101] support the *authoring* of new knowledge bases. The following components deal with the *interlinking* and *classification* of datasets, which corresponds to the research topic of this dissertation (highlighted in Figure 2.5). Interlinking datasets is a significant task to advance data integration in the Semantic Web. Link discovery frameworks are employed to determine additional connections while delivering high-quality results with minimal configuration effort [132]. Based on these links, both entities of the data sources and concepts of the associated ontologies can be classified [7]. This enables knowledge bases to be enriched, e. g., by adding axioms to ontology concepts [26] or to physically integrate entities to create unified knowledge bases [155]. Published Linked Data sources and knowledge bases can then be qualitatively analyzed. Surveys comparing quality analysis approaches provide an overview of the metrics and dimensions used to improve data quality [135, 195]. A further improvement of data quality in the Web of Data can be driven by services that continuously analyze datasets concerning quality metrics [40]. Together with the quality analysis, repair strategies are usually applied, which simultaneously monitor changes to the data and make them traceable. Ultimately, new interfaces and visualizations allow RDF data to be searched, explored [16, 120] or extracted for new applications [7].

The presentation of the components in the Linked Data life cycle makes it easy to see that modifications to individual components can have an influence on the entire process. For example, recognizing and removing incorrect links between data sources can help to improve a subsequent linking step. As a result, components of the life cycle are often considered in combination to improve the overall Linked Data quality [7].

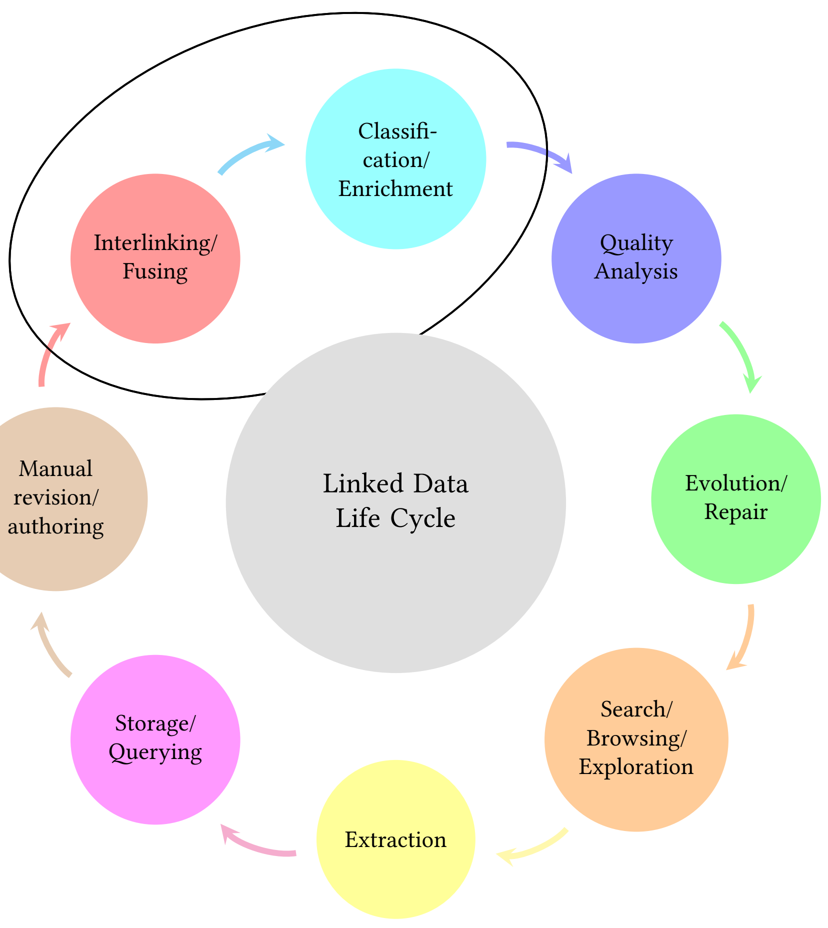


Figure 2.5: Linked Data life cycle (Source: [7]) with spotlighting (black ellipse) of the focused components in this thesis.

This dissertation combines the components interlinking/fusing and classification/enrichment to enable the integration of data sources to form a holistic knowledge representation. In addition, strategies for quality analysis and repair are used to improve linking results. To close the circle, existing links are also made available via the LinkLion web portal. Together with provenance information, these results can be reused for new applications, e. g., for a follow-up linking process.

2.2 Data Model

This section deals with the conception of a general data model for the subsequent integration of Linked Data sources. After presenting relevant definitions in Section 2.2.1, the data model is applied to example RDF statements in Section 2.2.2.

2.2.1 Definitions

We consider a set of k data sources $\mathcal{S} = \{S_1, \ldots, S_k\}$, each containing an arbitrary number of entities $\mathcal{E}$ such as e_1, e_2. Entities are referenced by a URI and have additional property values such as a label, semantic type information or domain-specific properties, e. g., 'place of birth'. For better readability, URIs are typically shortened to IDs like `(1)` or `(2)` (e. g., for e_1 and e_2) in examples and tables. Entities have different semantic types $\mathcal{T} = \{T_1, \cdots, T_m\}$, that can be used for a basic classification, such as 'person' or 'settlement'. Because different data sources may use various vocabularies to identify semantic types, they must be unified before they can be used further.

Two entities of different sources can be connected by a `owl:sameAs` link stating that they represent the same real-world object. Besides, a similarity value sim can be given for a link to indicate the strength of the relation, e. g., the highest similarity 1 to state equality. On this basis, a binary equivalence mapping $\mathcal{M}_{i,j} = \{(e_1, e_2, sim)|e_1 \in S_i, e_2 \in S_j, sim[0,1]\}$ is defined for all `owl:sameAs` links between two sources S_i and $S_j (1 \leq i, j \leq k, i \neq j)$. In our case, however, we consider the set of united equivalence mappings and therefore define $\mathcal{L} = \bigcup_{i,j=1}^{k} \mathcal{M}_{i,j}$ and thus simply refer to the set of all links.

To achieve the goal of integrated data sources, we determine clusters of entities linked by equivalence relationships and denote them as $\mathcal{C}$. Ultimately, for each semantic type T_i $(1 \leq i \leq m)$, a set of n_i clusters $\mathcal{C}_i = \{c_{i,1}, \ldots, c_{i,n_i}\}$ is created. Each cluster $c_{i,j}$ $(1 \leq i \leq m, 1 \leq j \leq n_i)$ only includes matching entities of type T_i (denoting the same real-world object). Furthermore, different clusters always correspond to distinct entities. Clusters $c_{i,j}$ also have a cluster representative $r_{i,j}$, which is derived from the properties of all the entities in the cluster. Thus, a unified entity representation is provided. The set of all representatives $r_{i,j}$ is called $\mathcal{R}$ in the following. In the remainder of the work, we will use these representatives $\mathcal{R}$ to compare new entities with existing clusters to avoid a more expensive comparison with all cluster members.

2.2.2 Transformation to Data Model for Clustering

The defined concepts are used to transfer the RDF representation into the data model, followed by a discussion of the advantages. We return to the already mentioned example of the philosopher Leibniz and show the representation in various data sources. Starting with DBpedia, RDF statements in Listing 2.2 show properties of Leibniz such as name, place of birth and semantic type.

Listing 2.2: RDF statements DBpedia

```
dbr:Gottfried_Wilhelm_Leibniz dbo:birthPlace "Leipzig" .
dbr:Gottfried_Wilhelm_Leibniz rdfs:label
    ↪ "Gottfried␣Wilhelm␣Leibniz" .
dbr:Gottfried_Wilhelm_Leibniz rdf:type dbo:Person .
```

Listing 2.3: RDF statements Freebase

```
fb:m.0372p fb:place_of_birth "Leipzig" .
fb:m.0372p rdfs:label "Gottfried␣Wilhelm␣Leibniz" .
fb:m.0372p rdf:type fb:people/person .
```

Listing 2.4: RDF statements Wikidata

```
wd:Q9047 wd:place_of_birth "Leipzig" .
wd:Q9047 rdfs:label "Gottfried␣Wilhelm␣Leibniz" .
wd:Q9047 rdf:type wd:human .
wd:Q75925 rdfs:label "Friedrich␣Leibniz" .
wd:Q75925 rdf:type wd:human .
wd:Q1094559 rdfs:label "Leibniz-Keks" .
wd:Q1094559 rdf:type wd:trademark .
```

By stating the same properties for the two sources Freebase and Wikidata in Listings 2.3 and 2.4, characteristic features and differences can be determined. The attribute values for the place of birth and the philosopher's name are identical in all sources, but different vocabularies (`dbo:person`, `fb:people/person` and `wd:human`) are used for the semantic type `rdf:type`. These heterogeneous attribute names and values must be unified in the course of the data integration process. In this example, the term 'Person' is used uniformly for the mentioned semantic types.

Table 2.3: Mapping of URIs for entities and consecutive determined IDs.

Entity ID	Entity URI
1	`dbr:Gottfried_Wilhelm_Leibniz` (DBpedia)
2	`fb:m.0372p`[26] (Freebase Gottfried Wilhelm Leibniz)
3	`wd:Q9047` (Wikidata Gottfried Wilhelm Leibniz)
4	`wd:Q75925` (Wikidata Friedrich Leibniz)
5	`wd:Q1094559` (Wikidata Leibniz-Keks)

In Listing 2.4, two further entities are given for Wikidata – Friedrich Leibniz (father of the philosopher) and the Leibniz-Keks (cookie named after G. W. Leibniz). Together with the three entities describing the philosopher, the mapping to the data model is carried out and relevant corner cases are shown.

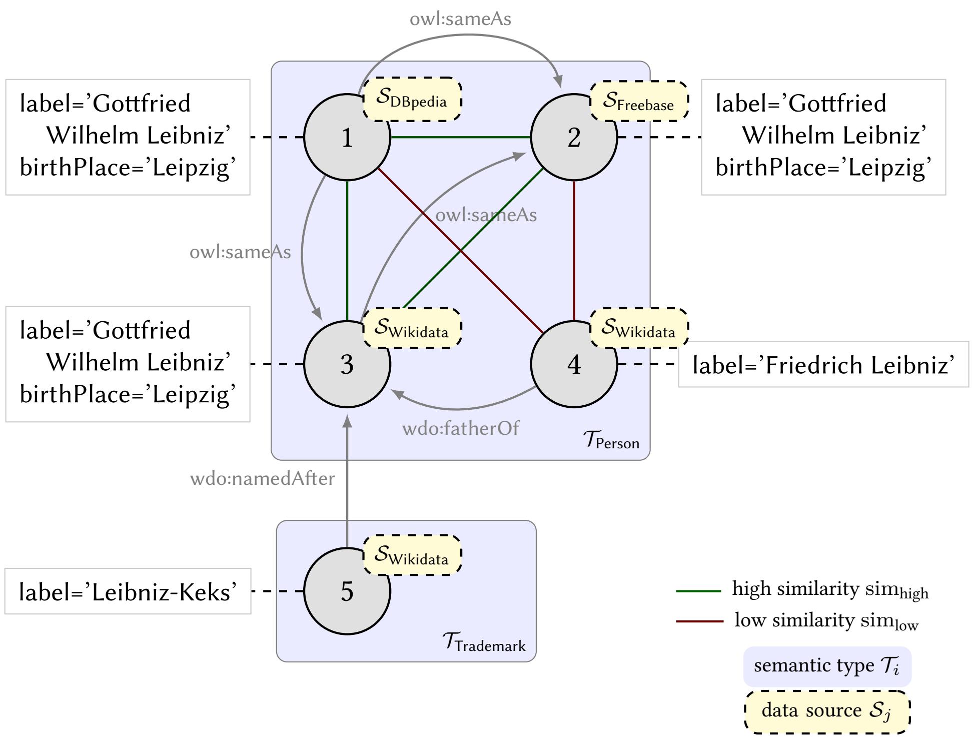
The set of data sources can be described as $\mathcal{S} = \{\text{DBpedia}, \text{Freebase}, \text{Wikidata}\}$. The five entities are grouped in Table 2.3 and each one is labeled with a consecutive ID. A manual unification of the semantic types results in $\mathcal{T} = \{\text{Person}, \text{Trademark}\}$. To create binary mappings, the links between different sources are calculated for all entities with type 'Person' as seen in Equations (2.1) to (2.3). Due to the same property values, the entities describing Leibniz have a very high similarity sim_{high}, which is stored in the links $(1, 2)$, $(1, 3)$ and $(2, 3)$. The low similarity sim_{low} given for the links $(1, 4)$ and $(2, 4)$ reflects that two different persons are described. For further use within a clustered representation, we join the binary mappings to the total set of all links $\mathcal{L}$ in Equations (2.4) to (2.7).

$$\begin{aligned} \mathcal{M}_{\text{DBpedia, Freebase}} &= \{(1, 2, \text{sim}_{\text{high}})\} && (2.1)\\ \mathcal{M}_{\text{DBpedia, Wikidata}} &= \{(1, 3, \text{sim}_{\text{high}}), (1, 4, \text{sim}_{\text{low}})\} && (2.2)\\ \mathcal{M}_{\text{Freebase, Wikidata}} &= \{(2, 3, \text{sim}_{\text{high}}), (2, 4, \text{sim}_{\text{low}})\}. && (2.3) \end{aligned}$$

$$\begin{aligned} \mathcal{L} &= \bigcup_{i,j=1}^{k} \mathcal{M}_{i,j} && (2.4)\\ &= \{(1, 2, \text{sim}_{\text{high}})\} \cup \{(1, 3, \text{sim}_{\text{high}}), (1, 4, \text{sim}_{\text{low}})\} && (2.5)\\ &\quad \cup \{(2, 3, \text{sim}_{\text{high}}), (2, 4, \text{sim}_{\text{low}})\} && (2.6)\\ &= \{(1, 2, \text{sim}_{\text{high}}), (1, 3, \text{sim}_{\text{high}}), (1, 4, \text{sim}_{\text{low}}), (2, 3, \text{sim}_{\text{high}}), (2, 4, \text{sim}_{\text{low}})\}. && (2.7) \end{aligned}$$

This situation is illustrated graphically in Figure 2.6, with semantic types and calculated links (green and red) highlighted. For improved comprehension, the original relationships between entities (`owl:sameAs`, `wd:fatherOf` and `wd:namedAfter`) are also displayed. In this way, the `owl:sameAs` links reconfirm the exemplarily calculated (green) links. Also, this shows that entities from a single data source are not compared (assumption of duplicate-free data sources), e. g., (3) and (4). Likewise, entities with different semantic types are never compared, so there is no similarity calculation

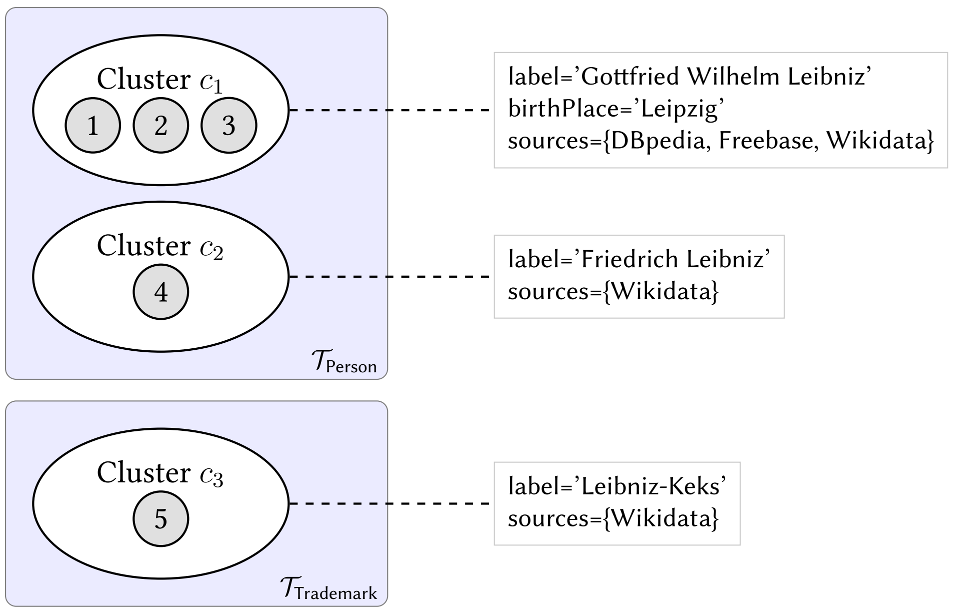
[26]Freebase is no longer in production, but URIs are still valid and can be used via data dumps.



owl:sameAs
label='Gottfried Wilhelm Leibniz' birthPlace='Leipzig'
$\mathcal{S}_{\text{DBpedia}}$
1
$\mathcal{S}_{\text{Freebase}}$
2
label='Gottfried Wilhelm Leibniz' birthPlace='Leipzig'
owl:sameAs
owl:sameAs
label='Gottfried Wilhelm Leibniz' birthPlace='Leipzig'
$\mathcal{S}_{\text{Wikidata}}$
3
$\mathcal{S}_{\text{Wikidata}}$
4
label='Friedrich Leibniz'
wdo:fatherOf
$\mathcal{T}_{\text{Person}}$
wdo:namedAfter
$\mathcal{S}_{\text{Wikidata}}$
label='Leibniz-Keks'
5
$\mathcal{T}_{\text{Trademark}}$
high similarity sim_{high}
low similarity sim_{low}
semantic type $\mathcal{T}_i$
data source $\mathcal{S}_j$

Figure 2.6: RDF statements transferred to graphical representation of data model.

for (5) despite the agreement in parts of the label. Figure 2.7 shows the clustered representation for the example. Besides a standardization of the property values, metadata on the contained entities and already covered data sources are listed.



Cluster c_1
1
2
3
label='Gottfried Wilhelm Leibniz' birthPlace='Leipzig' sources={DBpedia, Freebase, Wikidata}
Cluster c_2
4
label='Friedrich Leibniz' sources={Wikidata}
$\mathcal{T}_{\text{Person}}$
Cluster c_3
5
label='Leibniz-Keks' sources={Wikidata}
$\mathcal{T}_{\text{Trademark}}$

Figure 2.7: Clustered representation with unified property values.

3 Link Discovery

Link discovery is of considerable importance for the Semantic Web to advance the idea of Linked Data (see Section 2.1.1). Therefore, identifying new semantic relationships between entities of different data sources plays a central role for numerous applications, e. g., federated queries [74, 154, 165] or answering complex questions [117, 168]. It is related to the problem of entity resolution (also deduplication, reference reconciliation or object matching), which is being investigated extensively [31, 49, 109]. For example, comparable techniques are used to increase efficiency and determine a similarity between objects. However, significant differences such as large amounts of data and semantic heterogeneity have led to the development of link discovery frameworks.

A notable distinction is the support for varying semantic relations. Entity resolution methods are limited to search for equivalence relations. Instead, link discovery allows determining arbitrary semantic relations. Due to many interrelated resources, link discovery has to deal with high heterogeneity. Additionally, resources are often described by properties from ontologies. These properties or underlying ontology classes can then be linked to each other, which can lead once again to an increase in heterogeneity. By contrast, most entity resolution approaches focus on homogeneous datasets of relatively simple, structured objects, described by a set of single-valued attributes (e. g., benchmark datasets in [110]). For this reason, link discovery methods often rely on a combination of instance and ontology matching.

Accordingly, there are surveys for data linking in the Semantic Web that describe different techniques and frameworks [58, 191]. Based on these surveys and by analyzing additional frameworks, we published a comprehensive link discovery survey [132]. In addition to a definition and requirements for link discovery, frameworks are compared using a generalized workflow. This survey was reassessed in this thesis to present a range of current link discovery frameworks. While pointing out desirable features, it also identifies flaws that offer the potential for improvement. Furthermore, we show the relevance of link discovery in the context of data integration. Following the structure from the survey, Section 3.1 will provide a formal description of the concept of link discovery as a basis for determining semantic relations.

To further support the development of Linked Data, new data must be added and links to existing sources have to be calculated. However, the number of links between different data sources is often low because the process of creating them is both error-prone and time-consuming [167]. For instance, it takes less than 20 minutes to compare two data sources where each source consists of 1000 entities if a comparison is computed in 1 ms. With 1 million resources in each data source, the number of comparisons adds up to 1^{12}, which corresponds to more than 30 years of calculation time. Thus, the naive comparison of all resources of the two data sources is not anymore feasible. The reason for this is the quadratic complexity for evaluating the Cartesian product between two sets of entities. The calculation of new semantic relations for Linked Data requires reliable automatic processes that are efficient and effective. For this reason, we define requirements (Section 3.2) and a generalized workflow (Section 3.3) that lead to a stable link discovery process as far as possible. The comparison (Section 3.4) is then performed using link discovery frameworks that are actively under development. Besides the summary, differentiation to the remaining work is given in Section 3.5.

3.1 Problem Definition

The *link discovery* problem can be described as follows: Given two sets of resources S and T and a relation R (e. g., `owl:sameAs`[1] or `dbo:producer`[2]), find all pairs $(s, t) \in S \times T$ such that $\mathrm{R}(s, t)$ holds. The identified pairs are referred to as *links*. Depending on the data sources, different data properties are used to determine the links. A combination of data properties used for link discovery is called link configuration or specification. The set of resulting links is called a *mapping*: $M_{S,T} = \{(a_i, \mathrm{R}, b_j) | a_i \in A, b_j \in B\}$. Optionally a similarity score ($\mathrm{sim} \in [0, 1]$) computed by a link discovery tool can be added to the entries of mappings to express the confidence of a computed link. In this case, links can be represented as quadruples $(a_i, \mathrm{R}, b_j, \mathrm{sim}(a_i, b_j))$.

[1]Equivalence `http://www.w3.org/2002/07/owl#sameAs`

[2]Producer of a creative work `http://dbpedia.org/ontology/producer`

3.2 Requirements

To retrieve improved results for data integration for the Semantic Web, we define requirements for linking entities that are used in the remainder of the thesis.

Effectiveness An essential criterion for linking entities is a high-quality result, which is usually determined by the measures precision, recall and F-Measure compared to a known perfect result. Therefore, a linking system is better suited if it can determine both a high number of correct links in the set of calculated links (precision) and high coverage of links in the set of all relevant links (recall). In other words, the result should only contain links between entities that belong together in reality, according to the relation used. For an effective determination of links, match techniques of varying complexity or a combination of different techniques are used. Besides, different semantic link types should be supported.

Efficiency The processing of extensive datasets should be quickly feasible and approaches to scalability must be recognizable. The naive comparison of all possible pairs of two entity sets (Cartesian product) results in a problem of quadratic complexity. Increased efficiency is mainly achieved by reducing the search space, i. e., avoiding unnecessary comparisons by exploiting the distribution of the properties of entities. Another possibility is the efficient use of parallel computing units. This includes parallelization on a single computer (processor cores) as well as on distributed computer systems (clusters) or graphic cards (GPUs).

Configuration and Tuning Effort Different link specifications are required to achieve high effectiveness and efficiency, depending on the data. The combination of various similarity measures, entity attributes and other configuration parameters such as similarity thresholds create complex specifications. Manually specifying such configurations is very difficult and time-consuming so that automated approaches should primarily reduce this effort. This automation can be achieved by learning-based methods, e. g., by supervised approaches using training data of matching and non-matching pairs of resources. Alternatively, the link discovery framework can analyze the datasets, e. g., to select suitable similarity measures or properties to evaluate. It should also be ensured that the automated approaches minimize the configuration overhead, e. g., by avoiding additional parameters or a low amount of necessary training data.

Powerful Infrastructure The link discovery workflow should be executable on different platforms, preferably with parallel processing. Furthermore, frameworks should offer extensive possibilities to configure all parts of the workflow. In particular, link discovery tools should come with flexible libraries with different similarity functions, support different performance optimizations and provide different possibilities to access data sources. Overall, the user interface should be easy-to-use and configurable via graphical (web) interfaces. Mechanisms for collaborative work in groups or crowd-sourcing should be provided for easier labeling of training data or generation of gold standards. Overall, frameworks should be designed domain-independent. Still, it should be possible to flexibly customize it for more specific link discovery tasks, e. g., linking geographical resources or knowledge from the life sciences.

Static and Dynamic Link Discovery In addition to the classical one-time execution of link discovery on static data sources, more and more applications rely on the continuous integration of data sources. Therefore, link discovery approaches should be able to process both static and dynamically changing data. For example, already calculated similarity values can be reused to recognize changed relationships with incremental methods.

3.3 Generalized Workflow

Current link discovery frameworks rely on various substeps that lead to the generic workflow represented in Figure 3.1. The visualized objects can be divided into resources (gray) and processes (blue), with cylindrical resources representing either data sources or data sinks. Optional work steps or resources are marked with dashed borders.

The generic workflow itself consists of three main steps: preprocessing, matching and postprocessing. Preprocessing deals with two important tasks: finalizing the linking specification with parameters from the configuration and improving runtime efficiency, e. g., by reducing the search space for following similarity computations. The central phase consists of instance matching and the optional ontology matching to generate links according to the specified relation between source and target dataset. While the matching phase is entirely automatic, there may be user interaction for preprocessing, e. g., to label training data for learning-based linking and postprocessing, e. g., to verify computed links with lower confidence. The output of the workflow is the set of links representing the mapping between the source and target datasets. We now describe the components of the individual phases in detail and show differences in implementation.

The link discovery workflow starts with the preprocessing, where relevant resources are loaded. This includes the two datasets, parameters for the configuration and optional

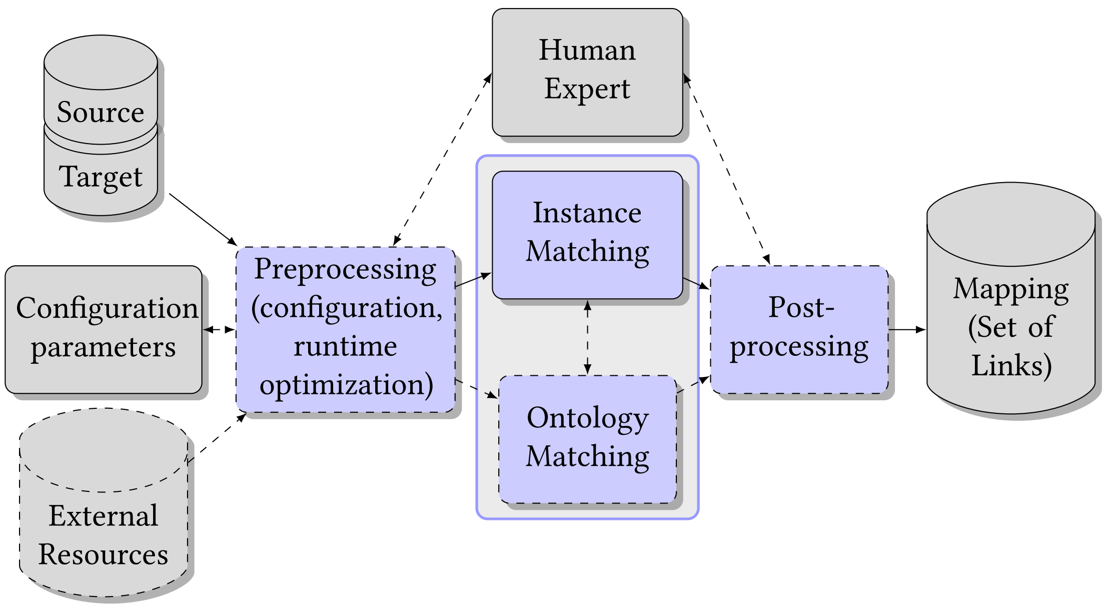


Figure 3.1: General workflow link discovery frameworks (steps with dashed borders are optional).

background knowledge resources. The datasets are provided in RDF/OWL format either as file dump or retrieved via query-based data access from a SPARQL endpoint. At this point, a selection of the relevant (ontology) class in the data source can already be made. This selection ensures, for example, that settlements of a geographical data source are not compared with persons from a general-purpose data source such as DBpedia. After importing configuration and resources, further preparatory steps like data cleansing are executed, e. g., stop words removal or abbreviation resolution.

3.3.1 Configuration

The task of the configuration step is to complete the linking specification by using the input parameters. This linking specification indicates which similarity measures should be used to compare the properties or semantic context of resources in the matching phase. Depending on the available properties in the data source, multiple similarity values can be aggregated using numerical approaches, e. g., weighted average. The application of minimum thresholds is also common to exclude unlikely results [109]. In contrast, rule-based approaches use match rules that are combined by logical operators to make a link decision. For instance, when comparing settlements, it can be required for a match that the settlement names have a trigram similarity of ≥ 0.9 and a low distance of the geographic coordinates [109]. Workflow-based approaches, which are not commonly used, are based on iterative calculations of different similarity values. For example, an initial similarity computation determines link candidates based on specific properties. These results are then restricted via a minimum similarity threshold. Finally, more complex computations are carried out according to proposals such as [136, 179].

The definition of complex linking specifications, such as combined match approaches on different data properties, can be provided manually or with automatic methods. A (semi-)automatic determination of the linking specification is carried out either by applying adaptive link discovery methods or learning-based methods. *Adaptive methods* analyze the input data and deduce parts of the linking specification, e. g., by applying logical rules [78, 143]. For (semi-)supervised *learning-based methods*, training data is typically used. This implies a dependency on appropriate training data where pairs of entities have been labeled as matching or non-matching in a prior step. Based on this training data, a classification model is created using techniques such as decision tree, SVM or genetic algorithms. Reduced effort in the creation of training data can be achieved by using active learning, where the model is influenced by specific feedback on controversial decisions. Learning-based approaches may also be unsupervised, thereby avoiding the need for training data. However, these approaches may still require the specification of critical parameters such as suitable similarity or distance measures and threshold values [143, 147].

3.3.2 Runtime Optimization

The runtime can already be considerably reduced by selecting the appropriate configuration parameters, e. g., by restricting it to the relevant entity class. However, the effect of an optimized pairwise entity comparison can have an even larger influence on the runtime. With naive methods, the quadratic complexity results in an unreasonably large number of comparisons. Therefore, this section discusses strategies to limit the number of necessary comparisons. In the literature, approaches for link discovery, ontology matching, (privacy preserving) entity resolution and duplicate detection are presented to realize a search space reduction with the common goal of runtime optimization [31, 47, 49, 152, 156]. Methods for runtime optimization are usually divided into two categories – blocking and filtering [182].

Blocking methods distribute the entities over several partitions (blocks) to compare all entities within these employing a (more complex) similarity calculation [152, 156]. This form of search space reduction is dealt with in more detail in the literature [47, 49, 152]. A further distinction in blocking techniques can be observed in the composition of the blocks. Some surveys [47, 49] explicitly distinguish between blocks with mutually exclusive subsets such as standard blocking [57] and overlapping subsets such as sorted neighborhood method [81], q-gram indexing [30] or meta blocking [151].

Filtering methods use the similarity measure or similarity threshold of the linking specification to exclude comparison pairs that do not meet the similarity conditions [132, 182]. For example, filtering methods may only retain candidate pairs when compared property values share a certain amount of tokens to exceed a threshold for token-based string similarity measures such as Jaccard or Dice [3, 9]. Additionally, the position or length of the string tokens [192] or characteristics of the metric space such as triangle inequality [139] can be exploited to filter unsatisfactory candidates. According to the defined link specification, filtering methods detect all matching pairs, whereas blocking methods may miss matches if the entities are unfavorably divided into blocks. For this reason, blocking methods are sometimes referred to as approximate methods and filtering as exact methods [30, 152]. Since index data structures are usually used both for the classification of entities into blocks and for grouping into filtered sets of entities, the notion indexing is also used to describe search space reduction techniques [31].

Both blocking and filtering methods require parameters in order to operate successfully. For blocking techniques, the choice of the blocking key(s) to determine the partitions is of particular importance. Depending on the specific method, other parameters are relevant, e. g., the length of q-grams for q-gram based indexing or the size of the sequential window for window-based methods. Since filtering methods use the similarity measure given by the linking specification, the corresponding parameters are also used for the subsequent matching step.

3.3.3 Match approaches

After runtime optimization, the main phase determines the similarities for the remaining candidate pairs. Therefore, similarity measures between defined resource property pairs are calculated according to the link specification. Link discovery frameworks usually support a combination of different match techniques (matchers) to cover the requirements according to the domain. At this point, background knowledge can also be used, e. g., to use language- or domain-specific dictionaries, thesauri or abbreviation lists [132].

The literature discriminates schema- or ontology-based matching and instance-based matching, whereby combinations of both methods are also possible [157]. This discrimination is also the case for link discovery, but with the focus on instance matching between entities based on most discriminative attributes [132]. Reasons for this include a large number of entities and, accordingly, the substantial heterogeneity of entity attribute values. Since access to ontology information is also available via the metadata of the instances, ontology matching offers the potential to increase effectiveness further.

In detail, match operators can be divided into element- and structure-level matcher, whereby instance matching is limited to element-level techniques [50, 157]. *Element-level* matchers are far more common and calculate similarity measures on individual atomic property values (e. g., strings, numbers) or domain-specific data types (e. g., dates, geographical coordinates). *Structure-level* matchers are usually more sophisticated by deriving a similarity value for resources from their contextual similarity. For example, the context of adjacent resources or higher-level ontology concepts can be included to determine a similarity between entities.

A promising link discovery approach is to utilize already existing links and mappings to find new links. Based on the transitivity of the equality relation one can compose several links to derive new `owl:sameAs` links. Thus, links between previously unconnected datasets are possible. Effective strategies for such a composition of mappings and links have been proposed and evaluated in [76]. The initial situation describes two datasets S and T, which can only be reached indirectly via links to intermediate datasets. The calculation of all paths between S and T can then be used to select the most promising candidates. Alternatively, graph-based algorithms like shortest path are used to find promising link candidates between multiple data sources. To enable the reuse of links for novel link discovery approaches, public mapping repositories such as BioPortal [149] or LinkLion (Chapter 4) can be used for storage and maintenance.

3.4 Comparison of Frameworks

Two critical factors mainly influenced the selection of the link discovery frameworks for the comparison. Candidate frameworks are expected to support the requirements defined in the previous section as much as possible. Also, the frameworks should show an active development, including evaluation results, which further limits the selection. The activity in the source code management system[3] shows ongoing work on the framework. For current evaluation results, successful OAEI benchmark participation[4] is important. In particular, good results for the instance matching or link discovery track are expected at least twice. Table 3.1 shows the four considered frameworks Silk, LIMES, LogMap and AML (successor of AgreementMaker[36]) while providing the first classification to criteria such as support for learning-based algorithms or ontology matching capabilities. Each of them has at least participated in the OAEI benchmark 2017 [1] and 2018 [2][5].

[3]Several commits on utilized platforms like `https://github.com/` within the last year

[4]Ontology Alignment Evaluation Initiative `http://oaei.ontologymatching.org/`

[5]LIMES is not mentioned by name for both OAEI 2017 and 2018, but the algorithm RADON is integrated in LIMES [169] and is therefore used accordingly.

Table 3.1: Compared link discovery frameworks (sorted by year of initial publication).

System / initial publication	Year	Learning-based	OAEI IM participation	Support for pure ontology matching
Silk [184]	2009	✓	✓	-
LIMES [139]	2011	✓	✓	-
LogMap [90]	2011	-	✓	✓
AML [53]	2013	-	✓	✓

The detailed comparison of the functional properties is carried out according to the generalized link discovery workflow. Table 3.2 shows an overview of the identified main features. For the specification of the input format, all four frameworks support RDF data. Additionally, Silk and LIMES can handle CSV files and read data from SPARQL endpoints. In contrast to the increased flexibility and dynamic for data access, the use of (public) SPARQL endpoints can also lead to availability and performance problems. With the original design for ontology matching, LogMap and AML also support reading OWL files. The use of background knowledge or dictionaries like WordNet, Wikipedia or ontologies like UMLS is widespread in the field of ontology matching [84]. However, the tools examined for link discovery usually have limited use for this type of information, including existing links and mappings. Therefore, it is not surprising that only the ontology matching frameworks use background knowledge. For example, AML uses WordNet to determine similarities to synonyms for instance matching [55]. Similarly, LogMap specifies the use of WordNet or UMLS to search for synonyms, but only in their initial publication for ontology matching [90].

The configuration step defines the used semantic relation (link type) and the link specification. In addition to the equivalence relation (`owl:sameAs` links), only Silk and LIMES can handle additional link types given by the user. The link specification states which properties of the entities are used to determine the similarity values and the corresponding weighting. LogMap and AML only accept a manual configuration while Silk and LIMES additionally allow different learning-based strategies for automatic link specification determination. In genetic programming, for example, a predefined fitness function is optimized by adjusting random link specifications. Therefore, the evolutionary principles selection and variance are applied to reach a performance criterion of the fitness function. Silk [87] determines a combination of discriminative properties to learn the link specification from reference links. This process is improved by applying genetic programming. Thus, the selection of different distance measures is optimized, including thresholds and variants of aggregation functions. Both Silk [86] and LIMES [142] combine genetic programming with active learning, i. e., several link specifications that sufficiently fulfill the performance function are evaluated by an oracle to improve the result. The LIMES approach is extended by [144], where correlations between link candidates are used to improve link specifications. In detail, two methods are proposed. First, graph clustering is used to determine correlations for candidates within the class. Secondly, the spreading activation principle is used to correlate candidates across classes that are close to each other. The basic idea is that these candidates could belong to one class. Therefore, the aim is to adjust the boundary of classes if the correlation between candidates is sufficiently high. The semi-supervised active learning and unsupervised learning-based methods in LIMES apply a binary classification to search for suitable link specifications [141, 143]. For example, in [141], potentially matching classes and properties are determined as a starting point for a link specification using the hospital-resident optimization problem. The method refines the candidates via active learning by training user feedback on unclassified property pairs in multiple iterations via a perceptron neural net.

Link discovery frameworks should provide methods to reduce the search space to enable scalable link discovery. With AML, LIMES and LogMap, three of the tools use filtering approaches. AML and LogMap rely on inverted index structures, which are generated in a preprocessing step by splitting labels into their sub-components. The strategy of LIMES is to exploit properties of the metric space, in particular, the triangle inequality. Distance calculations among reference points restrict the search space. When the distance between two reference points is high (and thus the similarity low), the similarity of nearby entities is also minimal. In detail, entities of the data source s are assigned to previously determined reference points. Then, the distances of the entities of source t to the reference points are used to filter out reference points that are too far away (and thus the assigned entities of source s). Accordingly, comparisons are only made between entities that have similar distances to the surrounding reference points. Further optimizations to search space reduction with LIMES are possible, e. g., for geo-spatial [137, 169] data with space tiling or temporal data with reasoning based time interval relations [65]. Silk is the only one of the presented methods that explicitly uses blocking for search space reduction [88]. Multiple blocking keys can be specified and based on custom similarity measures, indices are built for each of them. Afterward, the indices are aggregated into a single multi-dimensional index while preserving the individual information of the properties. The created index is eventually used to link candidates based on entities that share a block or are located in adjacent blocks. Finally, the link specification can be used to determine the similarity of candidates in the succeeding matching step.

In the following instance resp. ontology matching step, frameworks apply different types of matchers to determine similarities between entities. The most common method

Table 3.2: Characteristics of link discovery frameworks. GP - genetic programming, AL - active learning, "-" means not existing, "*" investigated in [83], but not available in current release

	LogMap	Silk	LIMES	AML
Supported link types	`owl:sameAs`	`owl:sameAs`, user-specified others	`owl:sameAs`, user-specified others	`owl:sameAs`
Configuration	manual weighted average	manual (match rules), supervised learning (GP, AL)	manual (match rules), supervised learning (GP, AL), unsupervised (GP)	manual weighted combination
Runtime optimization				
- Blocking	-	multi-dimensional	-	-
- Filtering	indexing	-	space tiling	indexing
String similarity measures	✓	✓	✓	✓
Further similiarty measures	-	numeric, date equality	geographical coordinates, numeric, date equality	geographical coordinates
Structure matcher	similarity of parental ontology concepts and neighborhood	-	-	combined annotation-/property-based similarity
Use of				
- external dictionaries	WordNet synonyms	-	-	WordNet synonyms
- existing mappings	-	-	-	-
Post-processing	inconsistency repair	-	-	-
Data input	RDF, OWL	RDF, SPARQL, CSV	RDF, SPARQL, CSV	RDF, SPARQL
Data output	RDF, OWL	RDF, CSV	RDF, CSV, XML	RDF
Parallel processing	-	MapReduce	(MapReduce)*	-
GUI / API	✓/ -	✓ / ✓	✓ / ✓	✓/ ✓
Web interface / online	✓/ ✓	✓ / -	✓ / -	✓/ -
Download tool/source	✓/ ✓	✓ / ✓	✓ / ✓	✓/ ✓
Open Source project	✓	✓	✓	✓

is to use element-level matchers to compare properties of entities based on their string similarity measures, e. g., edit distance, n-gram, or Jaccard [28]. Support for more specialized data types is available for Silk, LIMES and AML, e. g., numerical data, geographical coordinates or dates. In addition to the use of element-level matchers, LogMap and AML also apply structure-level techniques based on ontology information. LogMap applies an instance matching process that first creates candidate mappings, which are then refined [95]. Similar entities are determined via element-matcher on attribute values. This leads to the creation of a candidate set, which is then reduced by different methods. For example, candidates are eliminated if no match is found between the higher-level ontology concepts of the entities. Other structure-level techniques include the analysis of neighboring ontology concepts. If neighboring concepts also show a match, the candidate mapping is preserved.

The LogMap support for instance matching has been continuously extended [91, 92, 94] to accomplish the tasks within the OAEI. For AML, the instance matching approach is described in [54]. Besides the already mentioned element-level strategies, separate match strategies are used in certain situations to compute similarities on a structural level. If there is a high density of connections between the ontologies involved, a combination of string similarity and structural similarity is used. The case is handled differently if the structure-matcher detects the very same matches in the ontology for a high proportion of entities. Since this indicates a generally small structural difference for the set of entities, only entity-level matcher are used.

Postprocessing of the results only takes place with LogMap and AML. LogMap relies on a repair technique that generates coherent mappings using logical reasoning [93]. AML uses a similar algorithm, which is also based on logical reasoning to reduce the number of inconsistent results [166]. The results produced by the link discovery frameworks can then be stored in output formats such as RDF dialects or CSV.

Only Silk[6] and LIMES [83] provide the option for distributed execution of link discovery workload via Apache Hadoop MapReduce. The use of MapReduce has some disadvantages, like the mandatory materialization of intermediate results. Therefore, extensions such as Apache Spark or Apache Flink are increasingly used for workloads that can be easily parallelized. The usability of the examined link discovery frameworks is very good. They are available as open-source projects and offer interfaces via API or web interface. Thus, other researchers may use the projects together with new extensions.

[6]Silk MapReduce `https://www.assembla.com/spaces/silk/wiki/Silk_MapReduce`

3.5 Conclusion and Distinction to this Thesis

Following the definition of link discovery, resources of two data sources are compared with each other. The examined frameworks are designed for this purpose and therefore support designated techniques that allow effective computation of new links. Complex structure-based match techniques can be found with the tools LogMap and AML, which also have strong ontology matching capacities. Silk and Limes, by contrast, rely highly on instance matching strategies with simple property-based matchers. Both use learning-based methods to identify high-quality linking specifications for heterogeneous data sources. These sophisticated methods optimize the combination of data properties for subsequent matching, such as using a fitness function. Training with suitable data is a relevant factor, but should not be too time-consuming. The potential of reusing existing links or background knowledge has not yet been exploited to a relevant extent by the frameworks. Only LogMap and AML make use of synonym dictionaries. Efficiency is mainly addressed by filtering techniques for specific matchers rather than more general blocking approaches to reduce the search space. Besides single implementations for the MapReduce approach (Silk and LIMES), there is no support for more recent distributed processing frameworks.

However, the further development of Linked Data depends on the inclusion of additional data sources. These data sources have to be compared with many existing sources. This leads to quadratic complexity in relation to the number of data sources and a poor scalability of existing link discovery systems. So far, only a limited number of studies have partially addressed the use of existing link mappings to integrate new data sources and the resulting quality. Furthermore, the information generated during the creation of links, such as creation time, applied framework and matching methods, is not sufficiently evaluated. Such provenance information can be crucial for quality assessment and should be included for data integration. For this reason, this dissertation examines methods for the scalable integration of semantic data sources.

At first, a repository is presented in Chapter 4 to facilitate the reuse of existing link discovery results. With the provision of the platform, the storage, retrieval and maintenance of the links together with the associated provenance information can take place via programmatic interfaces. The stored links with metadata are a useful basis for the work of the following chapters and show the potential of reusing already calculated link mappings.

The presentation of a holistic clustering approach in Chapter 5 enables the integration of knowledge from multiple data sources. The approach relies on existing links and supports techniques to improve the effectiveness, e. g., by identifying incorrect links and by finding additional links. Eventually, this will produce a unified cluster representation of related resources according to the data model in Section 2.2. Chapter 6 then deals dynamically changing Linked Data sources and proposes two strategies to integrate new resources resp. data sources to existing knowledge graphs. The evaluation investigates whether dynamic approaches have advantages or disadvantages compared to static approaches.

Towards the efficient handling of many data sources, we provide a comprehensive discussion of concepts for distributed data processing in Chapter 7. The distributed implementation of both holistic and incremental clustering is presented in Chapter 8. The resulting stand-alone system allows the execution of complex workflows based on the distributed processing system Apache Flink. A detailed evaluation shows the performance for multiple large datasets from different thematic domains. To conclude the topic, a competitive evaluation of multiple scalable clustering methods is provided in Chapter 9. The comparison is executed on Apache Flink to show scalability and quality for multiple domains.

Part II

Reuse of Link Discovery Results and Clustering Strategies

4 LinkLion: A Link Repository for the Web of Data

Preamble

This chapter is based on [134]. LinkLion describes a web platform based on Semantic Web technologies that allow the reuse of link discovery results. By working with established ontologies, provenance information can be included in the results. Additionally, link results can be retrieved and maintained via a SPARQL endpoint or a REST interface. The portal was presented at a practical demonstration at ESWC 2014 and published in the associated conference proceedings.

4.1 Motivation

In addition to being central for question answering across several datasets, links also play a key role in various other domains such as data fusion and federated SPARQL queries. It is a well-known problem that links make up less than 3% of the RDF triples on the Web of Data [139]. This problem is being addressed by link discovery and ontology matching tools and frameworks [103, 138]. However, due to the architectural choices behind the Web of Data, the results of a link discovery framework cannot be added directly to the datasets involved in the link discovery process. Further, the direct addition of links to a knowledge base fails to provide means to track the source of these links for later reference. Moreover, the availability of some endpoints still remains a major issue[1], making the direct addition of linking results to some endpoints unattractive.

We address these drawbacks by presenting the open-source link repository LinkLion. The main goal of LinkLion is to facilitate the publication, retrieval and use of links between knowledge bases. Our repository thus provides dedicated functionality for the upload, storage, querying and download of large sets of links. Currently, it contains 77.8 million triples, which describe 15.5 million links of 12 different types (e. g., `owl:sameAs`, `dbo:spokenIn`, `foaf:made`, `spatial:P`) distributed on 3247 mappings that link 476 datasets. These links were retrieved from the Web as well as computed by tools, e. g., LIMES [138] and Silk [185]. Our repository provides a SPARQL query interface as well as commodity interfaces to access the mappings. In contrast to other portals such as BioPortal[2], LinkLion focuses exclusively on links and provides dedicated functionality for manipulating them. Moreover, we do not limit ourselves to a single domain, such as the life sciences. In the following, we give a brief overview of the repository and show the use cases that will be presented during the demo. The repository can be accessed at `http://www.linklion.org`. The source code of the repository is available on GitHub[3]. We provide a SPARQL endpoint at `http://www.linklion.org:8890/sparql`.

4.2 Implementation

Figure 4.1 shows the overview of LinkLion's architecture. The backend consists of a triple store in which we save data according to the vocabulary shown in Figure 4.2. The ontology[4] was designed with usability and reuse in mind. Especially, we wanted to allow end-users of the portal to select dedicated portions of certain mappings at will. This meant designing an ontology that allowed amongst others (1) retrieving all links that pertain to a particular resource or set of resources, (2) gathering all mappings between datasets of interest as well as (3) getting aggregated information on how particular links came about. We implemented this vision by storing the output of a link discovery tool under an instance of the mapping class. Individual mappings can be described by metadata, including the datasets that they link, the tool (including a version number) used to generate the links and the creation date of the mapping. We refrained from using blank nodes for links. Instead, we gave each link a unique ID. Note that we reused existing

[1]`http://labs.mondeca.com/sparqlEndpointsStatus.html`
[2]`http://www.bioontology.org/BioPortal`
[3]LinkingLodPortal `http://github.com/AKSW/LinkingLodPortal`
[4]Available at `http://www.linklion.org/ontology`.

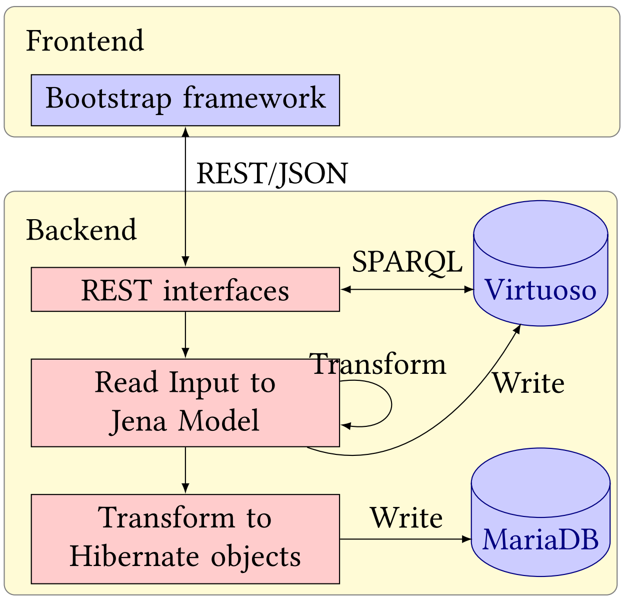


Figure 4.1: Architecture visualization of frontend and backend to store the mappings in the Virtuoso and MariaDB.

vocabularies (especially PROV[5], VoID[6] and DOAP[7]) as much as we could. The use of a triple store pays off as end-users can choose to provide more metadata, such as the link specification used or parameters of the algorithm they used to discover the link without us having to alter our schema. For the sake of scalability, we also provide the core of the data in the triple store as SQL dump. The functionality of the backend is exposed by RESTful interfaces, which allow programmatic access to LinkLion from code written in virtually any modern programming language.

The frontend of our repository provides an easy way to use some of the functionality of LinkLion (see Figure 4.3). First, it allows users to upload new mappings. Users are asked to provide a source file in the N-Triples format[8]. Moreover, the framework which was used to generate the links, as well as the algorithm used, has to be provided (note that we consider humans also to be linking frameworks). The data (and especially the mapping) given by the user is then checked for consistency and uploaded into the underlying triple store. The content of the triple store can be browsed directly from the web page (see Figure 4.4). Especially, the frontend includes search functionality and pagination, which allow end-users to search for mappings that link to or from a dataset of interest.

[5]http://www.w3.org/TR/prov-o/
[6]http://www.w3.org/TR/void/
[7]https://github.com/edumbill/doap/
[8]http://www.w3.org/2001/sw/RDFCore/ntriples/

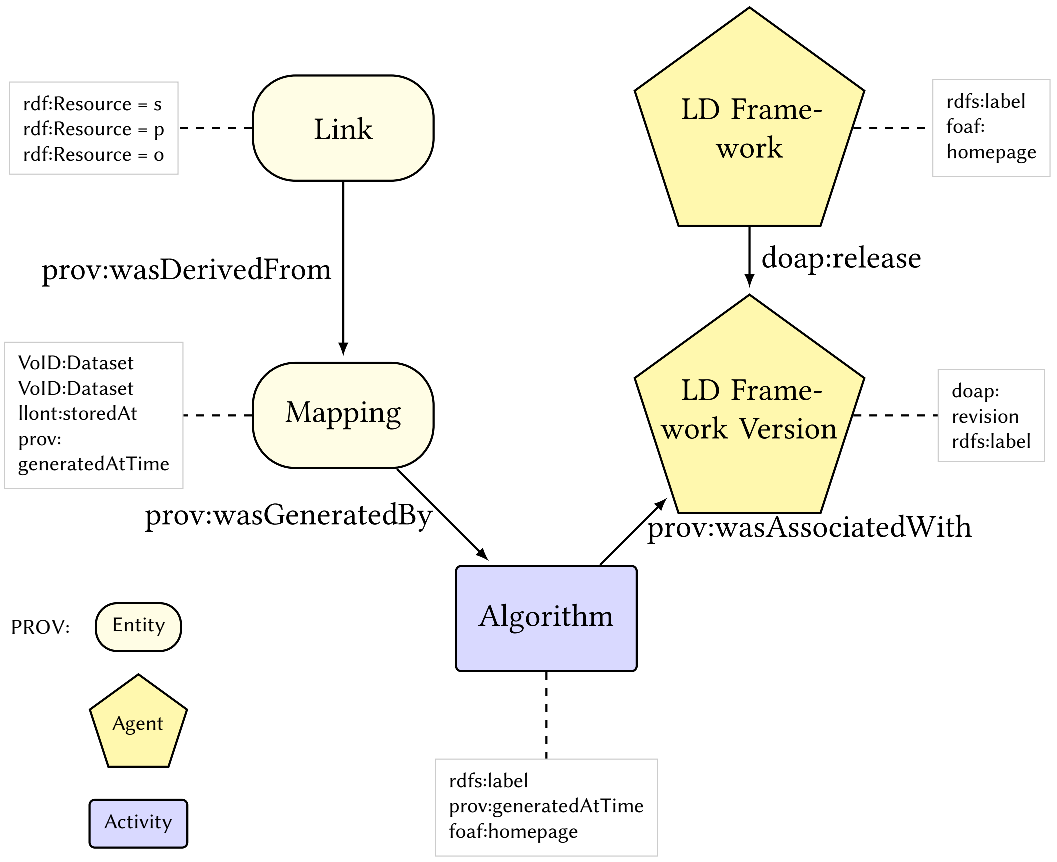


Figure 4.2: Overview of the LinkLion ontology. New classes such as Link, Mapping, Algorithm and LD (link discovery) Framework are specified as subclasses of the PROV vocabulary.

4.3 Use Cases

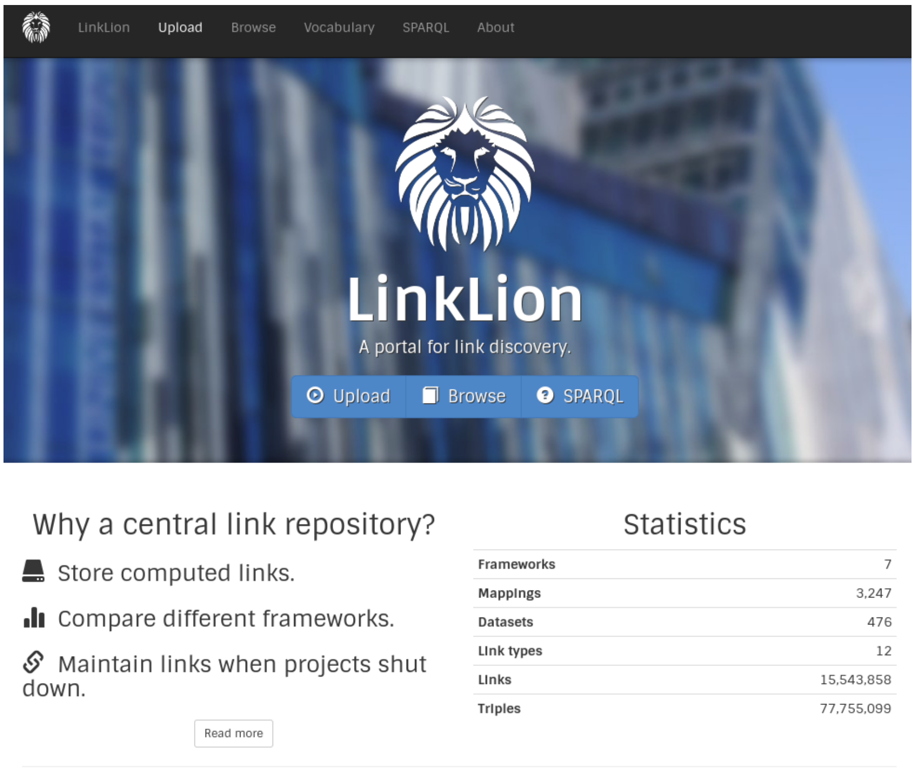
In this section, we present a selection of use cases to motivate users to adopt LinkLion.

4.3.1 Gather all links and mappings to a given resource

Gathering and fusing all information on a resource of interest is of central importance to applications such as question answering systems, Linked Data browsers and quality assessment tools. LinkLion allows to gather all links pertaining to a particular resource (`dbpedia:Thailand` in our example in Listing 4.1) through the following SPARQL query. By using this information, novel repair-based algorithms for link discovery such as Colibri can find errors or inconsistencies in the data [145].

Listing 4.1: Retrieve all links associated to a particular resource.

```
SELECT ?link WHERE { { ?link rdf:subject dbpedia:Thailand .}
             UNION { ?link rdf:object dbpedia:Thailand .} }
```



Why a central link repository?

Store computed links.

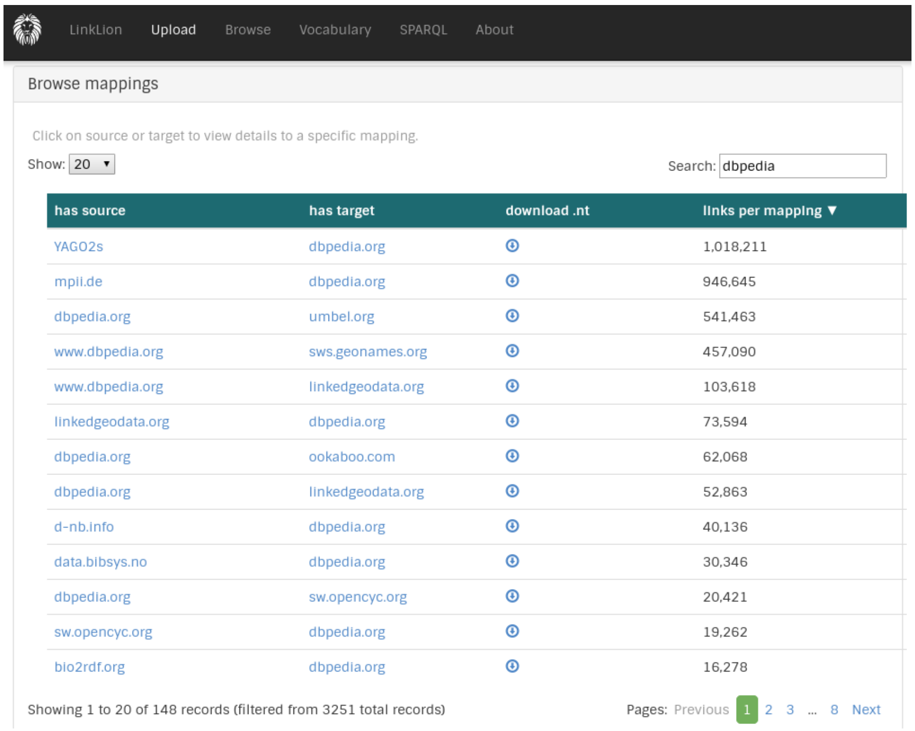
Compare different frameworks.

Maintain links when projects shut down.

Statistics

Frameworks	7
Mappings	3,247
Datasets	476
Link types	12
Links	15,543,858
Triples	77,755,099

Figure 4.3: LinkLion homepage with statistics regarding the content.



Browse mappings

Click on source or target to view details to a specific mapping.

Show: 20

Search: dbpedia

has source	has target	download .nt	links per mapping ▼
YAGO2s	dbpedia.org	⊙	1,018,211
mpii.de	dbpedia.org	⊙	946,645
dbpedia.org	umbel.org	⊙	541,463
www.dbpedia.org	sws.geonames.org	⊙	457,090
www.dbpedia.org	linkedgeodata.org	⊙	103,618
linkedgeodata.org	dbpedia.org	⊙	73,594
dbpedia.org	ookaboo.com	⊙	62,068
dbpedia.org	linkedgeodata.org	⊙	52,863
d-nb.info	dbpedia.org	⊙	40,136
data.bibsys.no	dbpedia.org	⊙	30,346
dbpedia.org	sw.opencyc.org	⊙	20,421
sw.opencyc.org	dbpedia.org	⊙	19,262
bio2rdf.org	dbpedia.org	⊙	16,278

Showing 1 to 20 of 148 records (filtered from 3251 total records)

Pages: Previous 1 2 3 ... 8 Next

Figure 4.4: Frontend view for the mapping browser. The search for DBpedia returns 148 mappings.

The portal also allows gathering all mappings that contain links about a particular resource, e. g., `dbpedia:Thailand`, as shown in Listing 4.2.

Listing 4.2: Get all mappings associated to Thailand.

```
SELECT DISTINCT ?mapping WHERE { ?link prov:wasDerivedFrom ?mapping .
                                { ?link rdf:subject dbpedia:Thailand }
                          UNION { ?link rdf:object dbpedia:Thailand } }
```

4.3.2 Get support for a link

Ensemble learning techniques have been shown to improve the results of manifold machine-learning applications, such as named entity recognition frameworks. Our repository facilitates the use of ensemble learning for combining the results of different link discovery tools. Especially, LinkLion allows us to retrieve (if any) the list of mappings that contain a given link, as well as the algorithms and the frameworks that generated it. In Listing 4.3, the support for the resources `<http://sws.geonames.org/1605651/>` and `dbpedia:Thailand` connected by a `owl:sameAs` link is queried.

Listing 4.3: Fetch metadata for mappings where a specific link is contained.

```
SELECT ?mapping ?algorithm ?framework WHERE {
        ?mapping prov:wasGeneratedBy ?algorithm .
        ?algorithm prov:wasAssociatedWith ?framework .
        ?link prov:wasDerivedFrom ?mapping ;
              rdf:predicate owl:sameAs .
        { ?link rdf:subject dbpedia:Thailand;
                rdf:object <http://sws.geonames.org/1605651/> }
        UNION { ?link rdf:object dbpedia:Thailand;
                      rdf:subject <http://sws.geonames.org/1605651/> } }
```

4.3.3 Link Composition

With the growth of the Linked Data Web, it becomes ever more important to regard link discovery as a holistic process that goes beyond linking a pair of knowledge bases. Algorithms based on composition can exploit sequences of links to enrich their mapping composition graphs [76]. Moreover, algorithms that link several knowledge bases at the same time [145] can achieve higher accuracies. By using LinkLion, composition and concurrent linking algorithms are now enabled to gather the data they require without having to manage all the links by themselves. In the query below, all resources related to `dbpedia:Thailand` over two links are retrieved from the repository.

Listing 4.4: Get resources related to Thailand over two links.

```
SELECT DISTINCT ?resource WHERE { {
        ?link rdf:subject dbpedia:Thailand ; rdf:object ?x .
        ?link2 rdf:subject ?x ; rdf:object ?resource }
        UNION { ?link rdf:object dbpedia:Thailand ; rdf:subject ?x .
                ?link2 rdf:object ?x ; rdf:subject ?resource } }
```

4.4 Conclusion

This chapter presents LinkLion, a repository for links between knowledge bases of the Web of Data. The repository enables users to upload results of a link discovery process and allows them to add information on how the results were created. Therefore, LinkLion provides management and distribution capabilities through both open-access and open-source web interface. Resulting sets of links can be reviewed in the portal and via SPARQL queries; additionally, the results can be downloaded via dumps.

In the following chapters, we demonstrate the high value of link portals like LinkLion. Data contained in LinkLion is therefore used to connect previously unlinked data sources and to detect incorrect links.

5

Holistic Entity Clustering for Linked Data

Preamble

The current chapter is based on [131]. The paper describes a concept for cluster determination, which can identify same real-world entities across multiple Linked Data sources. The intention is to avoid a similarity calculation between all entities. For this purpose, the approach relies on existing link discovery results to create initial clusters. They are then refined by split and merge operations to achieve the desired cluster representation. For this reason, the clustering approach is referred to as SplitMerge in the rest of the work. SplitMerge was presented at the DINA 2016 workshop with the following publication in the ICDM 2016 workshop proceedings.

5.1 Motivation

Linking entities between sources has been a significant effort in recent years to support data integration in the so-called Web of Data. A large number of tools for semi-automatic link discovery has been developed to facilitate the generation of new links (mostly of type `owl:sameAs`) [132]. Linked datasets can be accessed via platforms such as `DataHub`[1] or `<sameAs>`[2] and repositories such as BioPortal [149] or LinkLion [134], where numerous links for many sources are collected to improve their avail-

[1]DataHub platform `https://datahub.io`
[2]sameAs platform `http://sameas.org`

ability and re-usability. The continuous availability of already determined links is important to avoid their repetitive computation for new applications and use cases.

Despite the advances made, there are significant limitations in the achieved linkage of data sources and the current approaches for link discovery. First, the degree of interlinking is still low and automatically generated links are wrong in many cases [52, 139]. Current approaches for link discovery only match two data sources at a time (pairwise linking), resulting in poor scalability to many sources because the number of possible mappings increases quadratically with the number of sources. To be more precise, one needs up to $\frac{k\cdot(k-1)}{2}$ binary match mappings for k data sources. Hence, fully interlinking 200 sources in the Web of Data would require the determination and maintenance of almost 20.000 mappings.

Existing approaches to determine `owl:sameAs` links also focus on entities of the same type while many sources contain entities of different types (bibliographic datasets contain publication and author entities, geographical datasets contain numerous kinds of entities such as countries, cities, lakes, etc.). Furthermore, existing links are hardly utilized when additional links need to be determined. A general problem for the Web of Data is that integrating the information about entities requires users to specify within SPARQL queries the respective data sources and the links to be traversed. This makes it difficult to fully reach and combine the available information about entities.

To address these shortcomings, we propose *SplitMerge* as a clustering-based approach to holistically match entities between many sources. SplitMerge combines matching entities from k sources in one cluster instead of maintaining a high number of binary links. The matching entities of a cluster with their different properties can be easily fused to derive a more comprehensive, integrated entity representation that can be centrally maintained and accessed, e. g., within a knowledge graph [46, 146]. As sketched in [155], the clustering-based approach also facilitates the integration of additional sources and entities. They only need to be matched with the set of already existing clusters rather than adopting a pairwise linking with numerous different sources. The proposed approach to determine initial entity clusters can optionally build on already existing links and it can deal with entities of different semantic types. When existing links are utilized, SplitMerge can identify and eliminate wrong links and it clusters many previously unconnected entities. The fact that we can utilize existing links shows that the proposed clustering-based approach complements the prevalent pairwise linking, e. g., to provide the fused entities and entity clusters within integrated data sources such as knowledge graphs.

We illustrate the approach by the example records in Table 5.1 about real geographical entities of different types from five sources. The black lines on the left indicate existing

Table 5.1: Sample entities from the geographical domain to show holistic entity clustering. Black lines on the left denote same-as links between entities.

id	label	source	type	latitude	longitude
0	Lake Louise (Canada)	NYTimes	-	51.42	-116.23
1	Lake Louise	GeoNames	BodyOfWater	51.41	-116.23
2	Lake Louise, Alberta	DBpedia	Settlement	-	-
3	lake louise alberta	FreeBase	Settlement	51.43	-116.16
4	Lake Louise (Alberta)	DBpedia	BodyOfWater	51.41	-116.23
5	lake louise	FreeBase	BodyOfWater	51.41	-116.23
6	Mystic (Conn)	NYTimes	-	41.35	-71.97
7	N17632379615920	FreeBase	-	41.35	-71.97
8	Mystic	GeoNames	Settlement	41.35	-71.97
9	Black Hill	GeoNames	Mountain	53.54	-1.89
10	Black Hill	DBpedia	Mountain	53.53	-1.88
11	Black Hill	LinkedGeoData	Mountain	53.96	-1.85
12	Black Hill	LinkedGeoData	Mountain	54.69	-2.15
13	katmandu (nepal)	NYTimes	-	27.72	85.32
14	Kathmandu	GeoNames	Settlement	27.7	85.32
15	Kathmandu	DBpedia	Settlement	27.7	85.33
16	Kathmandu	FreeBase	Settlement	27.7	85.37

`owl:sameAs` links that we utilize as input to build the clusters. To achieve a good clustering, we cannot solely rely on existing links but also have to check the similarity between records; for this purpose, we consider the entity labels (names), their semantic type and the geographical coordinates. The table shows that the information about types and coordinates is typically incomplete, making it difficult to achieve good match quality. We can also observe potential errors in the given links. For instance, *Lake Louise* is both a city (type 'settlement') and a lake (type 'body of water'), so that entity (0) with an unknown type should only link to one of the two. The clustering leads to many additional matches compared to the initial links. For example, the cluster of the last four records about *Kathmandu* implicitly represents six matches, i. e., three new ones (`(14)-(15)`, `(14)-(16)`, `(15)-(16)`).

This chapter comprises the following contributions:

- The holistic clustering-based approach SplitMerge for matching entities in the Web of Data is proposed. It avoids the determination and maintenance of a huge number of pairwise mappings.
- We outline an initial method for the holistic clustering of real, possibly incompletely described entities of different semantic types. The approach utilizes existing links and can identify incorrect as well as many additional links. After some

preprocessing, the approach determines initial clusters by computing connected components that are subsequently refined by cluster splits and merges.

- An initial evaluation of SplitMerge shows the effectiveness for matching real entities from geographical data sources.

In the next section, we outline the problem statement. Section 5.3 describes and illustrates the SplitMerge clustering that builds on existing same-as links to determine entity clusters. A preliminary evaluation is presented in Section 5.4. Finally, we discuss related work in Section 5.5 and conclude.

5.2 Problem statement

The data model presented in Section 2.2 is used to implement the holistic clustering-based approach. Table 5.1 shows a running example with data sources $\mathcal{S}$ and semantic types $\mathcal{T}$. The URIs of the entities $\mathcal{E}$ are replaced by short IDs (1 – 15) to provide better readability. The specification of further properties is the basis for the following similarity calculation. Similarities, e. g., determined by a link discovery framework, provide the possibility to judge the strength of the connection between entities. Our main algorithm (Section 5.3) uses a set of existing (or pre-determined) link mappings $\mathcal{L} = \bigcup_{i,j=1}^{k} \mathcal{M}_{i,j} (1 \leq i, j \leq k, i \neq j)$ as input in addition to the set of associated entities $\mathcal{E}$ of the k data sources. The intention of the holistic entity clustering is the computation of a set of clusters $\mathcal{C}_i$ for each semantic type $T_i \in \mathcal{T}$, as described in the data model. Accordingly, different clusters also represent different real-world entities. With the general assumption of duplicate-free data sources, clusters can contain at most k entities. A cluster is therefore source-consistent if it contains at most one entity per source. Each cluster of $z \leq k$ entities represents $\frac{z\cdot(z-1)}{2}$ match pairs (e. g., in a link discovery result set). Thus, the cluster is generally a much more compact representation than with the use of binary links. The unified cluster representatives $\mathcal{R}$ are used to compare clusters with each other. The representative stores the list of encountered data sources to guarantee source consistency.

5.3 Holistic clustering based on existing links

In this section, we outline the SplitMerge approach to determine entity clusters based on existing mappings between the data sources to integrate. A high-level description of the approach is given in Algorithm 1. Figure 5.1 illustrates the main workflow and its application to the sample records from Table 5.1. In addition to the existing set of links and the associated entities from different sources, the input of the algorithm includes domain knowledge about the semantic entity types, a similarity function f_{sim} and similarity thresholds t_s, t_m to determine the similarity of entities and clusters. While our algorithm is generic, it can be customized to specific domains by providing appropriate background knowledge, similarity functions and thresholds. For the considered geographical domain, the similarity function determines a combined similarity from the string (trigram) similarity on normalized labels, the semantic type similarity and the normalized geographical distance. SplitMerge consists of four major steps described in detail in the rest of this section: preprocessing, initial clustering (connected components), cluster decomposition and cluster merge.

Algorithm 1: SplitMerge Clustering

```
Input: Set of entities $\mathcal{E}$ from $k$ sources, link set $\mathcal{L}$, domain knowledge $D$, simFunc $f_{\text{sim}}$,
       thresholds $t_s, t_m$
Output: Set of clusters $\mathcal{C}$
1  $\mathcal{C} \leftarrow \emptyset$
2  $\mathcal{E}, \mathcal{L} \leftarrow$ preprocessing($\mathcal{E}, \mathcal{L}, f_{\text{sim}}, D$)
                                                    /* initial clustering */
3  $\mathcal{C}_{\text{init}} \leftarrow$computeConnectedComponents($\mathcal{E}, \mathcal{L}$)
4  $\mathcal{L}_c \leftarrow$ computeLinkSim($\mathcal{C}_{\text{init}}, f_{\text{sim}}$)
5  $\mathcal{C}_{\text{init}} \leftarrow$refineConnectedComponents($\mathcal{C}_{\text{init}}, \mathcal{L}_c$)
                                                    /* cluster decomposition */
6  foreach $c \in \mathcal{C}_{\text{init}}$ do
7      $\mathcal{C}_{\text{split}} \leftarrow$ groupByType($c, \mathcal{L}_c$)
8      $\mathcal{C}_{\text{split}} \leftarrow$ simBasedRefinement($\mathcal{C}_{\text{split}}, \mathcal{L}_c, t_s$)
9      $\mathcal{C}_{\text{split}} \leftarrow$ createRepresentatives($\mathcal{C}_{\text{split}}$)
10     $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{\text{split}}$
                                                    /* create cluster mapping $\mathcal{CM}$ */
11 $\mathcal{CM} \leftarrow$ computeClusterSim($\mathcal{C}, f_{\text{sim}}, t_m$)
12 while $\mathcal{CM} \neq \emptyset$ do
13     $(c_1, c_2) \leftarrow \mathcal{CM}.getBestMatch()$
14     $c_m \leftarrow$ mergeClusters($c_1, c_2$)
                                                    /* cluster merge */
15     $\mathcal{C} \leftarrow \mathcal{C} \setminus \{c_1, c_2\} \cup \{c_m\}$
16     $\mathcal{CM} \leftarrow$ adaptMapping($\mathcal{CM}, \mathcal{C}, c_m, c_1, c_2, f_{\text{sim}}, t_m$)
17 return $\mathcal{C}$
```

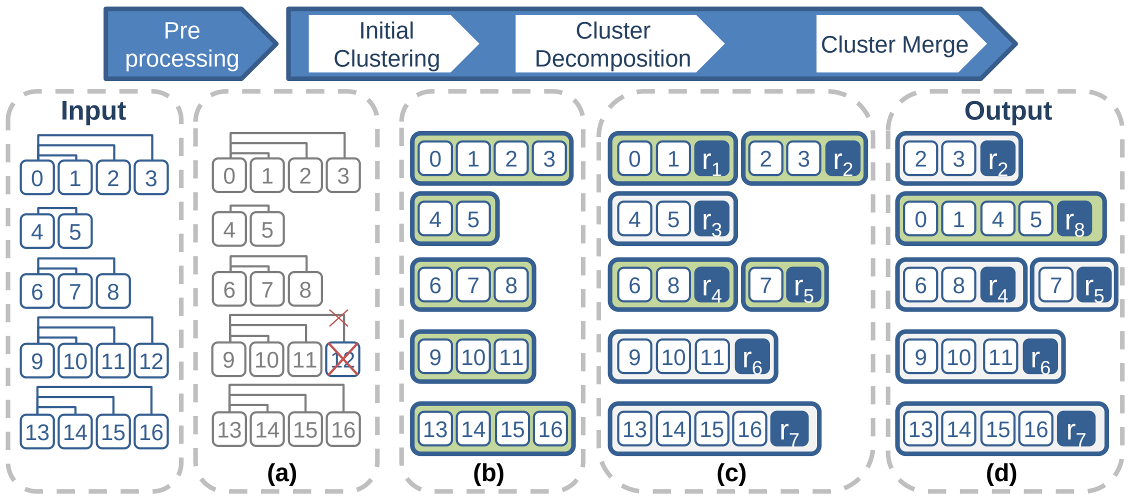


Figure 5.1: Application of the holistic SplitMerge clustering to the running example.

5.3.1 Preprocessing

During preprocessing, we normalize the property values needed for the similarity computation. In our case, we transform the label property to lower case and remove words in parentheses and after delimiters. We further harmonize information about the semantic types of entities and check that the input mappings do not violate the assumption of duplicate-free data sources. Thus, preprocessing already identifies and eliminates inconsistent links to start the clustering process with cleaned input data.

Information about the semantic type of entities differs substantially between sources or may be missing. For instance, DBpedia uses *City* and *Town*, whereas Freebase has a type *citytown* and other related types. To overcome such differences, we use background knowledge about the equivalence and comparability of entity types of different sources to harmonize the type information. For this study, we manually determine this type mapping for our geographical sources. Alternatively, the mapping can be constructed with the help of schema or ontology matching approaches based on linguistic and structural matching techniques[156, 171]. Based on the type mapping, we simplify numerous types to more general ones, e. g., the types *city*, *village* or *suburb* are treated as type *Settlement*. After harmonizing the type information, we remove all links where the linked entities have incompatible types. Note that we do not exclude links to entities with missing type information, e. g., entity (0).

For duplicate-free data sources, each entity should have at most one equivalent entity in any other data source. We therefore check whether the input mappings observe this one-to-one cardinality restriction. In Table 5.1, this is not the case for the Geonames entity (9) about *Black Hill* that links to two LinkedGeoData entities (11, 12). In such cases, we only keep the best link for an entity (based on the similarity function f_{sim}) to obey the one-to-one criterion. In our example, we discard the link `(9)-(12)` (see Figure 5.1 a) since `(9)` is geographically closer to `(11)` than to `(12)` according to the coordinates in Table 5.1.

5.3.2 Initial Clustering

Using the preprocessed entities and mappings, we first identify a set of initial clusters ($\mathcal{C}_{\text{init}}$) by computing all connected components as the transitive closure from the given links (see Algorithm 1, line 3). Each resulting connected component builds an initial cluster c covering all entities that are directly or indirectly connected via a same-as link in $\mathcal{L}$. In our running example, we create five different clusters covering 2-4 entities (see Figure 5.1 b).

If the data sources are not really duplicate-free or if there are errors in the predetermined same-as links, it may happen that the connected components contain more than one entity per data source and more than k entities in total. This incorrect association can occur despite preprocessing, in which it is ensured that each entity of a source is linked to at most one entity of a second source. For example, assume entities a_1 and a_2 from the same source and entities b_1 and c_1 from two different sources. The links $(a_1 - b_1), (b_1 - c_1), (c_1 - a_2)$ obey the 1:1 restriction but result in a connected component of all four entities of three sources. We thus determine within procedure `refineConnectedComponents` (line 5) connected components with more than one entity per data source and eliminate entities to ensure the restriction to enforce the assumption of duplicate-free sources. The entities to be removed are selected based on the similarity to other entities in the cluster. For this purpose, we calculate the similarity between any two entities of a cluster using similarity function f_{sim} and keep the result in the link set $\mathcal{L}_c$ (line 4). Based on these similarities, we keep from each source with multiple entities only the entity with the maximal similarity to the other entities.

5.3.3 Cluster Decomposition

The initially created clusters can contain entities that should actually be separated, e. g., due to wrong input links or because of an insufficiently high transitive similarity between entities. For decomposition, we use two main approaches. First, we split clusters with elements of different or incompatible semantic types. Second, we split clusters containing entities with insufficient similarity to other cluster members. In Algorithm 1, we apply the decomposition approach to each cluster in $\mathcal{C}_{\text{init}}$ (see lines 6 to 10). We first apply the two kinds of cluster decomposition (*GroupByType* and *Similarity-based Refinement*). Finally, we compute a cluster representative for each of the resulting clusters (line 9). The resulting clusters from the decomposition phase ($\mathcal{C}_{\text{split}}$) are successively added to the result set of clusters $\mathcal{C}$ (line 10).

Type-based Grouping

While we eliminate links with incompatible semantic types during preprocessing, there are entities without type information that can lead to clusters with entities of different types during the initial clustering. For our running example, this is the case for the cluster `(0,1,2,3)` (see Figure 5.1 b). Our method *groupByType* splits such clusters into several smaller sub-clusters $\mathcal{C}_{\text{split}}$ with entities of the same type. Entities without semantic types are then added to the sub-cluster of their most similar neighbor using the previously computed link similarities in $\mathcal{L}_c$. For the considered cluster of our example, we first build sub-cluster `(2,3)` for type *Settlement* and the singleton cluster `(1)` of type *BodyOfWater*. The untyped entity `(0)` is assigned to the cluster of the best matching (geographically closer) entity `(1)`, resulting in sub-cluster `(0,1)`.

Similarity-based Refinement

We further split clusters based on the computed intra-cluster similarity between entities (line 8 in Algorithm 1). Algorithms 2a and 2b show the computation of the similarity-based refinement in more detail. Algorithm 2a covers the iteration over the set of clusters $\mathcal{C}_{\text{split}}$ determined by `groupByType` and calls the actual refinement function `simCRefine` (Algorithm 2b).

Algorithm 2a: simBasedRefinement

```
Input: Set of clusters $\mathcal{C}_{\text{split}}$ resulting from type-based grouping, link set $\mathcal{L}_c$, split
       threshold $t_s$
Output: Set of clusters $\mathcal{C}_{\text{result}}$
1 $\mathcal{C}_{\text{result}} \leftarrow \emptyset$
2 foreach $c \in \mathcal{C}_{\text{split}}$ do
3     $\mathcal{C}_{\text{result}} \leftarrow \mathcal{C}_{\text{result}} \cup$ simCRefine($c, \mathcal{L}_c, t_s$)
4 return $\mathcal{C}_{\text{result}}$
```

For each cluster, we determine the entity e with the lowest average similarity asim_{min} (line 1 in Algorithm 2b) of its links to other cluster members using previously calculated similarity values from $\mathcal{L}_c$. If the average similarity is not below an individual threshold t_s, we do not split the cluster but leave the cluster unchanged (line 6). Otherwise (lines 2

Algorithm 2b: simCRefine

```
Input: Cluster $c$, link set $\mathcal{L}_c$, split threshold $t_s$
Output: Set of clusters $\mathcal{C}_{\text{result}}$
1 $e, \text{asim}_{\text{min}} \leftarrow$ getMinAvgSimEntity($c, \mathcal{L}_c$)
2 if $\text{asim}_{\text{min}} < t_s$ then
3     $c \leftarrow c \setminus \{e\}$
4     return createCluster($e$) $\cup$ simCRefine($c, \mathcal{L}_c, t_s$)
5 else
6     return $c$
```

to 4), we separate entity e and recursively call the similarity-based refinement for the reduced cluster to possibly identify further entities to separate. In the merge phase, such separated entities may be added to other more similar clusters. In our running example, the cluster `(6,7,8)` is decomposed into `(6,8)` and `(7)` as shown in Figure 5.1 c, since entity `(7)` has a low similarity with `(6)` and `(8)` due to its completely different label (see Table 5.1).

Cluster Representative

For each cluster in the output $\mathcal{C}_{\text{result}}$ of the previous steps, we create a cluster representative (line 9 in Algorithm 1) to facilitate the computation of inter-cluster similarities in the merge step. The representatives of the final clusters can also be used to efficiently match new entities, e. g., from additional data sources. We create the representative by combining the properties from all entities in a cluster and select a preferred value for each property with multiple values, e. g., based on a majority consensus, the maximal length of string values or pre-determined source priorities. In the representative, we also keep track of the data sources represented in the cluster (this helps to avoid considering merges with entities of already covered data sources). For our use case, we use the longest string value for the preferred label value and prefer geo-coordinates from GeoNames and DBpedia. For the example cluster `(2,3)`, the representative r_2 has label *lake louise alberta*, type *Settlement*, coordinates (51.43, -116.16) and sources (*DBpedia*, *Freebase*).

5.3.4 Cluster Merge

The last step of our holistic clustering approach is the possible merging of clusters below the maximally possible cluster size k. For this purpose, we first apply method *computeClusterSim* (line 11) to determine the similarity between clusters by applying similarity function f_{sim} on the cluster representatives. This operation is likely expensive as it incurs a quadratic complexity w.r.t. the number of clusters. We can reduce the number of comparisons by only considering clusters with fewer than k elements. We further do not consider all cluster pairs that differ in their entity type or that overlap in their sets of covered data sources. The cluster mapping $\mathcal{CM}$ computed for the remaining cluster pairs is restricted to the most similar pairs of clusters with a similarity exceeding the merge similarity threshold t_m.

Cluster merging is an iterative process (lines 12 to 16) that continues as long as there are merge candidates in $\mathcal{CM}$. In each iteration, we select the pair of clusters (c_1, c_2) with the highest similarity from $\mathcal{CM}$ and merge it into a new cluster c_m (lines 13 to 14). This merging also includes the computation of a new representative for c_m. The "old" clusters c_1 and c_2 are removed from $\mathcal{C}$ and the new cluster c_m is added (line 15). We further need to adapt $\mathcal{CM}$ by removing all cluster pairs involving either c_1 or c_2. Then, we extend $\mathcal{CM}$ by similar cluster pairs for the new cluster c_m, if c_m has fewer than k elements. For this purpose, we determine the similarity of c_m with all other clusters of the same type and with different sources. The termination of the merge step is guaranteed since we reduce the number of clusters in each iteration. The number of potential merge candidates is further reduced for increasing cluster sizes. Applying the approach to our example leads to the merging of $(0, 1, r_1)$ and $(4, 5, r_3)$ into the new cluster $(0, 1, 4, 5, r_8)$ (see Figure 5.1 c,d) due to a high label, type and geo-coordinate similarity as apparent from Table 5.1.

For our running example, the proposed approach could holistically cluster matching entities from five data sources, thereby finding previously unknown links and eliminating wrong existing links for improved data quality. The six clusters in the result set (Figure 5.1 d) implicitly represent 17 pairwise entity links compared to 12 initially given links (Table 5.1), from which 3 turned out to be incorrect. In particular, we could now identify matches between previously unconnected sources such as GeoNames and Freebase.

5.4 Evaluation

We evaluate the SplitMerge clustering approach using two datasets. Evaluation dataset 1 (ED1) corresponds to the DI location subtask of the OAEI 2011 Instance Matching benchmark[3] with links of presumed high quality. Evaluation dataset 2 (ED2) has been downloaded from the Linked Open Data repository LinkLion[4]. Figure 5.2 shows the number of

[3]OAEI 2011 IM: http://oaei.ontologymatching.org/2011/instance/
[4]LinkLion: http://www.linklion.org/

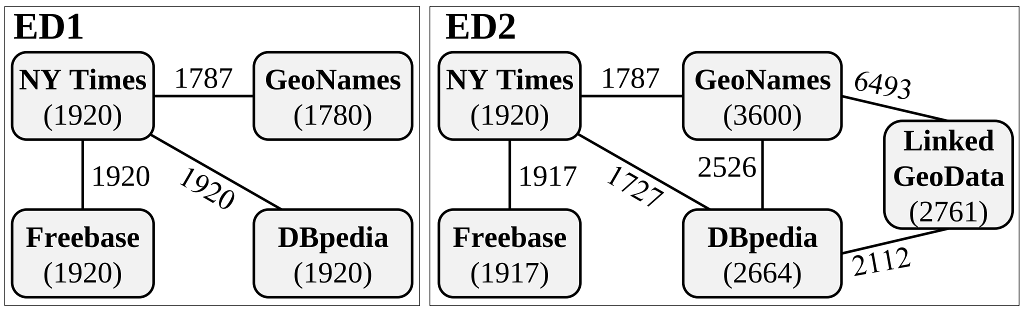


Figure 5.2: Dataset structures for ED1 and ED2 with number of entities and links.

links between the four (ED1) resp. five (ED2) geographical data sources and the number of entities that are interconnected by these links. We retrieved additional entity properties via SPARQL endpoints or REST APIs in the respective sources in 2015. Still, in ED1, geo-coordinates were missing for 1009 entities (13.4%) and the type information even for 2525 entities (33.5%). Similarly, in ED2, 957 entities (7.4%) have no geo-coordinates and 2722 (21.2%) do not cover type information. For instance, NY Times entities do not provide any type information. We use the similarity function described in Section 5.3. For the similarity thresholds t_s, t_m, we tested different settings, thus using a default of 0.7 that showed to produce good results w.r.t. cluster sizes and accuracy.

We first evaluate the resulting cluster sizes for the different phases of SplitMerge applied to these datasets (Table 5.2). During the preprocessing (not shown in the Figure), we already removed seven wrong NYT-GeoNames links based on the one-to-one cardinality restriction for ED1. In ED2, the situation is more complex due to many one-to-one violations (e. g., several entities can be linked among each other between two data sources). We therefore removed 6921 links, including over 5500 links from LinkedGeoData to GeoNames, to hold the one-to-one cardinality for each entity. For instance, there is a group of 94 entities interlinked by 367 links. Each of the 94 entities represents a 'Black Hill' whereof 3 entities occur in DBpedia, 45 in LinkedGeoData and 48 in GeoNames. Within each source, the entities do not represent duplicates but are actually different 'Black Hills'. In particular, there are 364 links between the 45 LinkedGeoData and 48 Geo-Names entities, i. e., many entities are interlinked with several entities in the respective other source. We thus keep only the best link for each entity in preprocessing.

As shown in Table 5.2, the initial clustering leads to clusters of sizes 3 and 4 for ED1, whereas ED2 provides clusters of size 2–5. This satisfies the condition that a cluster should not cover more entities than considered data sources. Applying the type-based grouping and similarity-based refinement results in a significant number of cluster splits and clusters of size 1 and 2 due to incompatible entity types and partially low intra-

Table 5.2: Cluster sizes in workflow phases for evaluation datasets.

ED1

cluster size	initial clustering	decomposition type-based	decomposition sim-based	cluster merge
1	-	50	174	159
2	-	115	153	154
3	140	180	228	229
4	1780	1680	1594	1597

ED2

cluster size	initial clustering	decomposition type-based	decomposition sim-based	cluster merge
1	-	46	362	339
2	711	794	836	831
3	759	773	817	807
4	1067	1011	1063	1075
5	586	580	432	435

Table 5.3: Cluster accuracy for a sample of result clusters in datasets.

ED1

cluster size	|cluster|	|sampled clusters|	cluster accuracy	link accuracy
2	154	6	100.0	100.0
3	229	9	88.89	92.59
4	1597	64	98.44	98.96
	1980	**82**	**97.47**	**98.56**

ED2

cluster size	|cluster|	|sampled clusters|	cluster accuracy	link accuracy
2	831	37	100.0	100.0
3	807	36	100.0	100.0
4	1075	48	97.91	98.61
5	435	19	100.0	100.0
	3148	**140**	**99.29**	**99.36**

cluster similarity. In particular, the similarity-based decomposition points out that several clusters contain dissimilar entities. During the merge phase, some of the smaller clusters can be merged into larger ones leading to more clusters of sizes 3, 4 (and 5). In particular, 15 (23) singleton clusters could be merged into clusters of size 2, 3 (and 4) for ED1 (ED2). Overall, the resulting clusters in ED1 represent 10 423 links with 4803 new links compared to the input link set. In particular, we could cluster many entities from the previously unconnected sources GeoNames, DBpedia and Freebase. For ED2, we had to remove a high number of links that violate the one-to-one cardinality restriction (preprocessing), leading to 9641 input links for the initial clustering. Similar to ED1, we then identify 4411 new links for ED2 in the subsequent workflow phases.

Evaluating the quality of the resulting clusters and thus the linking quality is challenging as it requires a perfect clustering for comparison. Determining such a perfect clustering is inherently difficult even for humans and very time-consuming. We therefore evaluate the quality for a sample of the result clusters in this initial evaluation (see Table 5.3). We randomly selected 4–5% of the clusters according to the distribution of the cluster sizes in the datasets. This selection leads to 82 clusters for ED1 and 140 clusters for ED2. We manually check the accuracy of the created clusters with all implicitly contained links. The *cluster accuracy* denotes the percentage of correct clusters (clusters that only contain entities with same-as semantics) in all sampled clusters, and the *link accuracy* is the proportion of correctly created links. For ED1, all determined links in clusters of size 2 are correct such that the cluster and link accuracy is 100%. For cluster size 3, there is one wrong cluster with two wrong links and one wrong cluster of size 4 with 4 wrong links. The cluster of size 4 should have been actually separated into two clusters of size 2. Overall, the evaluated clusters in ED2 showed to be very accurate. Only one cluster of size 3 contained two incorrect links.

Overall, we achieve a very high cluster accuracy of 97.5% (99.3%) and a link accuracy of 98.6% (99.4%) for ED1 (ED2). This meets our requirements for creating highly accurate clusters that can later be iteratively expanded by adding further entities. So far, we still have clusters of the size 1 and need to check whether those entities build separate clusters or need to be added to other clusters. Based on these results, we will extend our dataset samples and do more extensive evaluations w.r.t. the cluster quality in the future.

5.5 Related Work

Link discovery has been studied intensively and a large number of approaches and prototypes has been developed as surveyed in [132]. Virtually all approaches determine links between only two sources and for entities of one semantic type. A few approaches such as [21, 76] try to utilize existing mappings for deriving additional mappings, e. g., by their transitive composition. Similarly, [145] uses data from multiple Linked Data sources to detect weak or not existing relations based on the transitivity of correct links. In [100], the authors aim at improving the quality of joins on Linked Open Data by determining highly connected entity groups in a set of given links using metrics such as edge betweenness. Although these approaches reuse existing links, the focus is primarily on deriving new or correcting existing pairwise links. By contrast, our goal is a holistic clustering of entities from many data sources and central maintenance of such clusters for easy usability and extensibility. Our approach can deal with entities of different semantic types, finds many additional links, and identifies and eliminates wrong links for improved match quality.

A holistic matching of concepts in Linked Open Data sources has been proposed in [72]. The authors first apply a topical grouping of concepts and then perform pairwise matching of concepts within groups (based on keywords from the concept labels and descriptions) to determine clusters of matching concepts finally. The approach is interesting as it tries to holistically combine conceptual knowledge across data sources, e. g., as useful for the construction of knowledge graphs. However, the approach suffers from scalability and coverage limitations and does not address the clustering of entities as in our scheme. In contrast, the approach of Bellare et al. [11] enables scalable clustering of data sources from the Semantic Web based on distributed processing using Apache Hadoop. The proposed framework allows the clustering of (potentially dynamic) multiple sources based on connected components and subsequent refinement of the clusters. Disadvantages are the restriction to the outdated Hadoop MapReduce platform and the lack of public availability of the source code.

Clustering-based approaches have also been studied for entity resolution [31] outside the Web of Data, however, mainly for only one or two data sources. For two sources, proposed clustering approaches have similarities to our scheme in that they derive the clusters from a binary mapping consisting of pairs of matching entities [77]. For the related problem of deduplication, Costa et al. [35] propose an incremental clustering procedure that solves the assignment of new entities via two individual approaches. Both approaches use hash-based index structures to determine the optimal clustering. In a naive approach, an inverted index is built that contains the word tokens of the attribute values of the entities. The index is used to find similar entities for new elements for which the set-based Jaccard distance should be determined by the word tokens. All candidates that exceed a defined distance threshold are used to perform the cluster assignment for the new element via a voting mechanism. The second approach employs locally-sensitive hashing (LSH) [66] on sets of contiguous entity value substrings of length q (q-grams) to find similar entities. Using min-wise independent permutations to encode the q-grams of each entity to a hash-based fingerprint, similar entities have more agreements in the fingerprints than dissimilar entities. The evaluation shows that the LSH-based approach is more efficient because fewer similarity computations are needed for deduplication. The approach by Costa et al. [35] treats all information of an entity as a single textual attribute value. It is therefore limited in the evaluation of semantic properties and does not support domain-specific attributes.

Some entity resolution approaches construct a similarity graph from the match correspondences and determine subgraph clusters of connected and highly similar entities [71, 153]. These approaches are not only limited to two data sources but also consider only one type of entities. They also do not consider the data quality issues we had to deal with regarding wrong links and missing property values. The clustering approaches of Saeedi et al. [163, 164] also use similarity graphs for entity resolution. Further similarities to the herein presented approach, like support for multiple data sources and holistic clustering of entities, allow a more detailed comparison. For this reason, the approaches of Saeedi et al. are introduced at appropriate points in this dissertation and compared with the evaluation results obtained as follows. In Chapter 6, the results of two static entity clustering approaches [130, 164] are compared to the results for the newly introduced incremental clustering. With the additional support for distributed processing systems in Chapter 8, the subsequent comparison of multiple distributed clustering approaches [162] on Apache Flink is carried out in Chapter 9 together with the relevant discussion of results.

5.6 Conclusion

In this chapter, we proposed the holistic SplitMerge approach for clustering-based link discovery for many data sources. The approach utilizes existing links and can match entities of different semantic types. The determined entity clusters facilitate the integration of more data sources without having to link them to each other data source individually. An initial evaluation for Linked Data from the geographical domain confirmed that SplitMerge holds great promise as it can identify wrong links and many additional links even between previously unconnected sources.

However, this holistic clustering approach on multiple data sources offers the potential for further work. For this reason, a distributed implementation of SplitMerge is evaluated in Chapter 8 while an optimized domain-specific variant of SplitMerge is presented in Chapter 9. Furthermore, data sources are not always static, e. g., sources like DBpedia are regularly updated and new entities are added. Alternatively, data sources that have not yet been considered can be thematically relevant. An extension of the holistic clustering approach to continuously changing data sources is thus discussed in the following Chapter 6.

6 Incremental Clustering on Linked Data

Preamble

The subsequent chapter is based on [133]. The concept of incremental clustering allows a continuous addition of new entities and data sources. This complement to previously often static linking and clustering can be used on multiple data sources. Optimized methods are proposed to cover different use cases. An effective integration of entities from several sources is possible as well as the addition of complete data sources. The incremental clustering concept was presented on the DINA workshop in 2018 and published in the appropriate ICDM workshop proceedings.

6.1 Motivation

Identifying and linking equivalent entities in different sources is the main challenge within the Web of Data – many approaches have been developed to address this problem [132]. Besides pairwise linking of sources, there is an increasing need to integrate equivalent entities from arbitrary sources more holistically, e. g., for entity clustering within knowledge graphs [155]. Such a clustering facilitates the combination of the property values of all clustered entities, e. g., persons, products or cities, for enriched data representation. Entity clustering is a challenging problem since the degree of semantic data heterogeneity and differences in data quality increase with the number of sources from which equivalent entities should be clustered. In addition to high match and cluster quality, high efficiency and scalability are essential for large data volumes

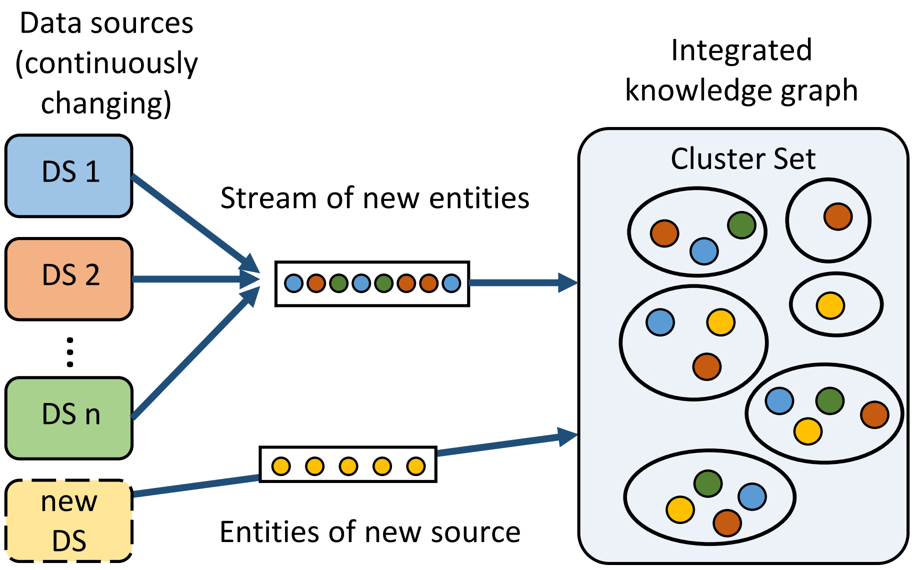


Figure 6.1: Incremental update of entity clusters, e. g., in knowledge graphs.

and many sources. Finally, there is a strong need for dynamic or incremental entity clustering that is not limited to a one-time computation of entity clusters, but that can continuously update entity clusters to cope with changing and new entities, including the incorporation of additional sources (see Figure 6.1).

Previous approaches to entity clustering [77, 163], including our ones [131, 164], are mostly static, i. e., they determine entity clusters a single time from a fixed number of static data sources. The clustering approaches use as input a set of binary `owl:sameAs` links connecting equivalent entities of different sources. These links either exist already in the Web of Data or have to be computed as a preparatory step. The links are commonly organized within a similarity graph where vertices correspond to entities and edges to similarity links. Unfortunately, the overhead to determine similarity graphs and thus for static entity clustering is very high and grows quadratically with both the number and size of the sources[1]. This holds even if one exploits common performance optimizations such as blocking [31] which reduces the number of entity comparisons only by a constant factor k when we partition the entities of a source into k blocks (since an entity only needs to be compared with the entities of one block per source). As a result, the scalability of static entity clustering is likely limited to a smaller number of sources of moderate size.

We thus argue for the use of dynamic or incremental entity clustering that can continuously update entity clusters for new entities and new sources without having to

[1]For n sources with m entities each, we have to compare each entity of a source with $(n-1) \cdot m$ entities given a total of $\frac{n \cdot (n-1)}{2} \cdot m^2$ comparisons if we link every entity pair only once.

completely recompute entity clusters. One approach could be to use a static clustering scheme to create an initial set of clusters and use a separate approach to keep the clusters up-to-date. We favor, however, a single dynamic approach that treats the entities of one of the sources as an initial set of clusters and incrementally adds further entities from the same or other sources as similarly sketched in [155]. This approach also promises a reduced runtime compared to a static clustering of many sources since entity clusters can incrementally be created in several steps without the need for the expensive creation of large similarity graphs.

Specifically, the following contributions are made in this chapter:

- A proposal of incremental approaches to cluster entities from multiple sources where new entities either result in new clusters or the addition to existing clusters. The addition of entities to clusters is set-oriented, so that multiple entities are considered together for an optimized assignment, compared to the isolated addition of individual entities. Furthermore, we provide a separate approach to add entities of a single source.

- An evaluation for the incremental approaches analyzes cluster quality and runtime for three domains and provides a comparison to static entity clustering.

In Section 6.2, we define the problem and provide background information on entity clustering. Section 6.3 explains the proposed approach describing two strategies to realize incremental clustering, which are then evaluated in Section 6.4. Finally, we briefly discuss related work in Section 6.5 and conclude in Section 6.6.

6.2 Problem definition

While data sources may contain entities of many kinds, we focus on clustering entities of a specific type of interest T, e. g., cities, persons or music songs for our evaluation datasets. We assume further that there are no duplicates within data sources but only between data sources. We adhere to the data model from Section 2.2.

An *entity cluster* or just cluster c groups a set of entities $e_1, \dots, e_k$ that are assumed to represent the same real-world entity. For each cluster, we determine a so-called *cluster representative* r fused from the properties of all entities in the cluster. Specific properties of the cluster representatives are used to determine the similarity between a new entity and an existing cluster as a basis for deciding whether the entity should be added to the cluster. We call a cluster *source-consistent* if it contains at most one entity per source, otherwise source-inconsistent since it violates the assumption of duplicate-free sources. The task of *entity clustering* has as input a set of data sources and determines and maintains a set of source-consistent and disjoint entity clusters $\mathcal{C}$ such that all entities within a cluster match and different clusters refer to different real-world objects. This entity clustering should be efficient and scalable to many sources and large data volumes.

For static entity clustering, the input is a fixed set of sources and the clustering is performed only once. For dynamic or incremental clustering, however, the number and contents of data sources can continuously change so that the entity clusters are to be adapted accordingly to reflect the current state of the input data correctly. While entities of a source may be added, changed or deleted, we focus on the addition of new entities as the most complex case for maintaining entity clusters. Similarly, we only consider the addition of new sources but not their removal.

The problem of *incremental clustering* thus has as input an existing (possibly empty) set of source-consistent and disjoint entity clusters as well as a set of new entities. The objective is then to create a new set of source-consistent and disjoint entity clusters that include the new entities. In the general case, we have to find an n:1 assignment between the new entities and the existing clusters. This is the case because an entity is added to at most one of the previously existing clusters (or to a new cluster). However, several entities (of different sources) may be added to the same cluster. A special case is given when all new entities come from the same source (e. g., when a new data source is to be incorporated). Then, a 1:1 assignment is required, in which each existing cluster is extended by at most one new entity to avoid source-inconsistent clusters. The 1:1 assignment problem between two sets of entities has already been studied extensively in the past. Thus, approximate solutions such as stable marriage [63], Hungarian algorithm [128] and the so-called Max-Both approach [42] have been proposed for its solution. A recent study [60] has compared these alternatives and found the overall best effectiveness and runtime efficiency for Max-Both. With this strategy, an element e_i of the first set is assigned to the best matching entity (cluster) c_j of the second set only if e_i is also the top match for c_j, e. g., there is no other entity for which the similarity with c_j is higher. Therefore, we apply the Max-Both approach in our source-specific incremental matching.

6.3 Incremental Clustering Approach

This section describes the approach for the integration of dynamically added entities into existing clusters. A first step creates all possible cluster candidates for integrating the

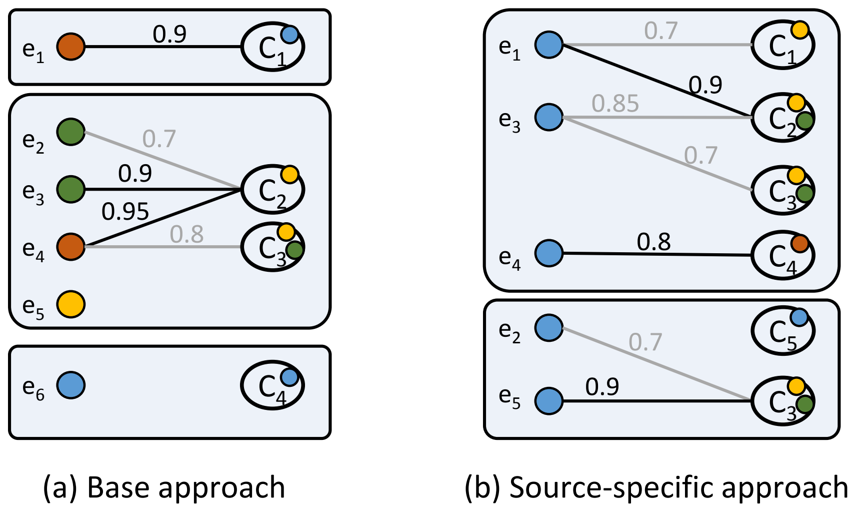


Figure 6.2: Cluster scenarios for base and source-specific approaches (colors imply different data sources, all links are assumed to exceed the minimal similarity threshold).

new entities into existing clusters. Thus, the conditions of source-consistency as well as exceeding a minimum similarity threshold t_{min} according to a domain-specific similarity function f_{sim} must be adhered to.

To limit the effort for finding the cluster candidates and improve the performance of our incremental clustering, we apply standard blocking [31]. Both clusters and new entities are partitioned into several blocks with a blocking function f_{blocking} on selected properties. For example, person entities could be partitioned based on a fixed-length prefix of their surnames so that we only consider clusters for which the cluster representative has the same prefix than the new person entity.

A straight-forward approach would consider each new entity e in isolation and add it to the cluster with the highest similarity above t_{min} that does not yet contain another entity from e's source. If no such cluster exists, a new cluster is created for e. While this is a reasonable approach, it may not be the best one if there are several entities to be added at the same time. For the example in Figure 6.2a, the assignment of entity e_2 to cluster c_2 would mean that no further entity of e_2's source could be added to c_2, thereby preventing that the more similar entity e_3 is assigned to c_2. This problem is especially pronounced if all new entities are from the same source where we have to find a near-optimal 1:1 assignment.

In the following, we therefore present two set-based approaches for incremental entity clustering that try to avoid sub-optimal cluster assignments. We start with the general approach that applies for new entities of different sources and outline then the source-

Algorithm 3: Set-based incremental entity clustering (base approach)

```
Input: Existing clusters $\mathcal{C}_{\text{exist}}$, new entities $\mathcal{E}_{\text{new}}$, similarity function $f_{\text{sim}}$, blocking
       function $f_{\text{blocking}}$, minimum similarity threshold $t_{\text{min}}$
Output: cluster set $\mathcal{C}$
1  $\mathcal{C}_{\text{new}} \leftarrow$ createInitClusters($\mathcal{E}_{new}, f_{blocking}$)
2  $\mathcal{C}_{\text{exist}} \leftarrow$ addBlockingInfo($\mathcal{C}_{exist}, f_{blocking}$)
3  for block i in Parallel do
4      $\mathcal{L}_i \leftarrow$ getClusterCandidates( $\mathcal{C}_{exist}, \mathcal{C}_{new}, f_{sim}, t_{min}$)
5      $\mathcal{L}_{\text{sorted}} \leftarrow$ sortLinkSims($\mathcal{L}_i$)
6      foreach $(c_{new}, c_{exist}, sim) \in \mathcal{L}_{sorted}$ do
7          if $c_{exist} \notin \mathcal{C}_{new}$ then
8              continue()
9          if isSrcConsistent($c_{new}, c_{exist}$) then
10             $c_{\text{exist}}.add(c_{\text{new}})$
11             $\mathcal{C}_{\text{new}}.remove(c_{\text{new}})$
12 return $\mathcal{C}_{exist} \cup \mathcal{C}_{new}$
```

specific approach to determine a 1:1 assignment between new entities and clusters. We assume that the set of existing entity clusters is non-empty. This is not a limitation since the approach can easily be bootstrapped by using the entities of one of the sources as the initial clusters.

6.3.1 Base approach

The pseudocode for set-based incremental clustering for entities of different sources is shown in Algorithm 3. It uses as input the set of new entities $\mathcal{E}_{\text{new}}$, the set of existing clusters $\mathcal{C}_{\text{exist}}$ as well as blocking function f_{blocking}, similarity function f_{sim} and the minimum similarity threshold t_{min}. Initially, a cluster is created for each new entity (cluster set $\mathcal{C}_{\text{new}}$) and assigned to one of the blocks according to f_{blocking}.

The main processing is performed independently (in parallel) for each of the blocks with new entities. First, we compare every new entity with every cluster representative of a block using f_{sim} and create a set $\mathcal{L}_i$ with candidate links. $\mathcal{L}_i$ only contains links $(c_{\text{new}}, c_{\text{exist}}, sim)$ that exceed a threshold t_{min} and where c_{exist} does not yet contain an entity from the source of c_{new}. To avoid sub-optimal cluster assignments, we then sort the $\mathcal{L}_i$ links and process them in descending order of their similarity so that the cluster assignments with the highest similarity can be realized first. A new entity from cluster c_{new} is only added to cluster c_{exist} if it does not lead to a source-inconsistent cluster. The addition to the cluster (function *add*) also involves adapting the cluster representative. Since there are generally several candidate clusters for a new entity, we can ignore all further links for an assigned c_{new}. We achieve this by removing c_{new} from the set $\mathcal{C}_{\text{new}}$ of not yet assigned entities and check at the beginning of each iteration (line 7) whether the current entity still needs to be assigned. The final result consists of the adapted cluster set $\mathcal{C}_{\text{exist}}$ as well as the singleton clusters for new entities that remain in $\mathcal{C}_{\text{new}}$.

Figure 6.2a shows an example for the base approach with three blocks where thick lines indicate the determined assignments. We observe that entity e_3 is assigned to c_2 since it is considered before e_2 due to its higher similarity. The assignment of e_2 is not considered since its source is now already represented in c_2. Entity e_4 has two cluster candidates, but after the assignment to c_2, it is removed from $\mathcal{C}_{\text{new}}$ so that its further cluster candidates like c_3 are ignored.

6.3.2 Source-specific incremental entity clustering

We now deal with incremental clustering of new entities from a single source where we determine a 1:1 assignment to the existing clusters based on the Max-Both approach. The pseudocode in Algorithm 4 uses the same input parameters than before and determines source-consistent candidate links between the new entities and the existing clusters in parallel within partitioned blocks. To implement Max-Both, we determine a set of top links for new entities ($\mathcal{L}_{\text{left}}$) and another link set for clusters ($\mathcal{L}_{\text{right}}$). For each new entity, $\mathcal{L}_{\text{left}}$ contains the link to the most similar cluster and accordingly, $\mathcal{L}_{\text{right}}$ contains for each cluster the link to the most similar entity. The assignment is then only done for links that have the maximal similarity from both sides. The intersection of $\mathcal{L}_{\text{left}}$ and $\mathcal{L}_{\text{right}}$ (line 11) thus leads to the set of resulting links.

For the example of Figure 6.2b, the algorithm assigns only three entities to existing clusters for which the Max-Both condition is satisfied. Note that the base approach would have assigned entity e_3 to cluster c_3 while this assignment is not taken for Max-Both since e_3 has a higher similarity with c_2. In general, Max-Both leads thus to fewer cluster additions than the base approach to achieve high precision with only safe cluster additions.

6.4 Evaluation

We first describe the evaluation datasets and setup and then analyze the effectiveness and runtime efficiency of the proposed incremental clustering approaches.

Algorithm 4: Source-specific incremental entity clustering

```
Input: Existing clusters $\mathcal{C}_{\text{exist}}$, new entities $\mathcal{E}_{\text{new}}$, similarity function $f_{\text{sim}}$, blocking
       function $f_{\text{blocking}}$, minimum similarity threshold $t_{\text{min}}$
Output: cluster set $\mathcal{C}$
1  $\mathcal{C}_{\text{new}} \leftarrow$ createInitClusters($\mathcal{E}_{new}, f_{blocking}$)
2  $\mathcal{C}_{\text{exist}} \leftarrow$ addBlockingInfo($\mathcal{C}_{exist}, f_{blocking}$)
3  for block i in Parallel do
4      $\mathcal{L}_i \leftarrow$ getClusterCandidates( $\mathcal{C}_{exist}, \mathcal{C}_{new}, f_{sim}, t_{min}$)
5      $\mathcal{L}_{\text{left}} = \emptyset$
6      $\mathcal{L}_{\text{right}} = \emptyset$
7      foreach $c_{new_i} \in \mathcal{C}_{new}$ do
8          $\mathcal{L}_{\text{left}}.add(\mathcal{L}_i.getBestLeftCandidate(c_{\text{new}_i}))$
9      foreach $c_{exist_i} \in \mathcal{C}_{exist}$ do
10         $\mathcal{L}_{\text{right}}.add(\mathcal{L}_i.getBestRightCandidate(c_{\text{exist}_i}))$
11     $\mathcal{L}_{\text{max-both}} \leftarrow \mathcal{L}_{\text{left}} \cap \mathcal{L}_{\text{left}}$
12     foreach $(c_{new}, c_{exist}, sim) \in \mathcal{L}_{max\text{-}both}$ do
13         $c_{\text{exist}}.add(c_{\text{new}})$
14         $\mathcal{C}_{\text{new}}.remove(c_{\text{new}})$
15 return $\mathcal{C}_{exist} \cup \mathcal{C}_{new}$
```

Table 6.1: General information for evaluation datasets.

	domain	entity properties	#entities	# sources
DS-G1	geography	name, longitude, latitude	3,054	4
DS-M1	music	artist, title, album,	19,375	5
DS-M2		year, length	1,937,500	5
DS-P1	persons	name, surname,	5,000,000	5
DS-P2		suburb, postcode	10,000,000	10

Table 6.2: Overview of perfect cluster sizes and best configuration values.

	perfect result		results best configuration		
	# clusters	# links	conf(t_{min}, bk)	# correct links	F-measure
DS-G1	820	4,391	conf(0.4, 1)	4,109	0.983
DS-M1	10,000	16,250	conf(0.5, 1)	15,066	0.955
DS-M2	1,000,000	1,624,503	conf(0.7, 1)	1,396,520	0.874
DS-P1	3,500,840	3,331,384	conf(0.7, 6)	2,720,479	0.814
DS-P2	6,625,848	14,995,973	conf(0.7, 6)	11,847,678	0.805

6.4.1 Datasets and Setup

We use five datasets of three domains (geography, music and persons) that have already been used for static entity clustering [130, 163]. Tables 6.1 and 6.2 give an overview of the datasets, including available properties and the number of entities. The smallest dataset DS-G1 comprises real geographic data on settlements from the four knowledge bases DBpedia, GeoNames, Freebase and NYTimes. The datasets for the music and person domains are based on real data, which are then synthetically changed with the DaPo tool [82] to introduce corruptions of property values as well as duplicates across sources. Music datasets DS-M1 and DS-M2 originate from MusicBrainz[2], while the person datasets DS-P1 and DS-P2 are based on publicly available voter data from North Carolina. The largest dataset, DS-P2, has 10 sources and 10 million entities. The datasets with their perfect cluster result can be obtained from our website[3].

Table 6.3 shows the blocking and similarity functions applied in the experiments. For blocking, we apply load-balanced standard blocking using a prefix of a single property (geographic and music datasets) or the concatenated prefixes of two properties (person datasets) as a blocking key. We consider prefixes of different lengths to vary the block sizes and thus the number of necessary comparisons and achievable recall. For similarity computation, we mostly compute the Cosine Trigram similarity for string attributes; for the geographical entities, we additionally consider the normalized geographic distance. For the music datasets, we apply Cosine Trigram on the concatenation of three property values. For DS-P1 and DS-P2, we determine the average of four property similarities. For all datasets, a match requires that the computed similarity values meet a variable minimum similarity threshold $t_{\min}$.

Table 6.3: Blocking and similarity functions used for incremental clustering.

dataset	blocking key	similarity function
DS-G1	prefixLength(name): 1	Trigram (name) + normalized geographical distance
DS-M1 / DS-M2	prefixLength: 1-5 of (artist + title + album)	Trigram (artist + title + album)
DS-P1 / DS-P2	prefixLength(surname): 1-3 + prefixLength(name): 1-3	avg(Trigram (name) + Trigram (surname) + Trigram (suburb) + Trigram (postcode))

[2]Musicbrainz https://musicbrainz.org/

[3]FAMER project https://dbs.uni-leipzig.de/research/projects/object_matching/famer

The quality of the incrementally determined entity clusters is compared with the perfect cluster results that are available for our datasets. Each cluster corresponds to several match links (correspondences) by assuming that each pair of entities within a cluster matches. So the resulting link sets of the computed clusters and the perfect clusters are used to determine the standard metrics precision, recall and the harmonic mean, F-measure, to measure cluster quality. Table 6.2 includes the number of clusters and links of the perfect cluster result as well as the number of correct links and F-measure for the best incremental clustering configurations. Hence, the incremental clustering schemes achieve F-Measure results of more than 95% for the two smaller datasets DS-G1 and DS-M1 and more than 80% for the larger datasets.

The experiments are carried out on a cluster with 16 workers, each of them equipped with an Intel Xeon E5-2430 6x 2.5 GHz, 48 GiB RAM, 2x 4 TiB SATA disks and 1 Gbit Ethernet connection. The machines operate on OpenSUSE 13.2 and Flink 1.5.0. All experiments are carried out three times to determine the average execution time. Together with a description of the implementation on Apache Flink, the scalability of incremental clustering is examined in Chapter 8.

6.4.2 Experimental Results

We analyze the cluster quality and runtime of incremental clustering for our datasets. Mostly, we consider the incremental addition of entire sources and, thus, the use of source-specific incremental clustering. However, we also examine the partial addition of data sources and compare source-specific clustering with the base approach as well as the static entity clustering.

Effect of data source ordering

The proposed incremental approaches are likely dependent on the order in which sources and entities are added to the integrated set of clusters. For example, an incorrect addition of entities to clusters impacts the cluster representatives and thus all further cluster decisions. We thus evaluated different orderings for the incremental addition of data sources for our datasets. We found that the ordering had little impact on the synthetically generated music and person datasets because the sources are largely of the same size and quality.

However, for the real dataset DS-G1, we observed significant differences for the orderings. In general, we could achieve very good clustering quality for this small dataset with the best F-measure of 98.4% (precision 99.98%, recall 96.8%), which meets the best results for static entity clustering [130, 164]. The top result is achieved by the sequence

Table 6.4: Effect of data source ordering for DS-G1.

Source	Freebase		GeoNames		NYTimes		DBpedia	
Position for clustering	1./2.	3./4.	1./2.	3./4.	1./2.	3./4.	1./2.	3./4.
Avg. F-measure	0.971	0.955	0.970	0.964	0.974	0.952	0.944	0.982

(Freebase, GeoNames, NYTimes, DBpedia). We analyzed all possible order permutations and found that source DBPedia should not be used in the beginning since it lacks the geographical coordinates for $57\,\%$ of its records, while the other sources have the coordinates for almost all entities. In Table 6.4 we show for each of the four sources the average F-measure results for all permutations where the source has either been used in the beginning (as first or second source) or at the end (source no. 3 or 4) in the sequence of added sources. We observe that the use of DBpedia in the beginning leads to much lower quality than its use towards the end (average F-measure of $94.4\,\%$ vs. $98.2\,\%$) since the missing information can lead to many wrong match decisions. By contrast, using other sources at the start leads to generally better F-measure than using them last. We conclude that data quality (w.r.t. the properties used for matching) has a high impact and that incremental clustering should start with high-quality data sources.

For comparison, we also used the base approach for incremental clustering rather than the source-specific approach. As expected, the base approach resulted in a somewhat reduced cluster quality with 97.3% F-measure for the best sequence (instead of 98.4%). We further investigated a more dynamic change scenario without adding entire sources. For this, we initially add 80% of the entities of the four sources (randomly selected) followed by two further batches of new entities, each consisting of 10% of the entities of the four sources. The base approach for incremental clustering achieved an F-Measure of 97.0% for this scenario, indicating that the partial addition of sources does not significantly reduce cluster quality.

Comparison of incremental and static clustering

We now compare the runtime and cluster quality for the two incremental approaches with static entity clustering for datasets DS-M2 and DS-P1. We focus the analysis on a specific blocking key (length 4 for DS-M2, length 6 for DS-P1), similarity threshold (t_{min} 0.7) and use the same blocking and similarity functions to determine the similarity graphs for static entity clustering. We consider two state-of-the-art approaches for multi-source entity clustering on the similarity graph, namely the CLIP approach of [164] and the SplitMerge approach from Chapter 5.

Table 6.5: Incremental compared to static clustering for DS-M2 and DS-P1 with blocking key length 4 and 6 resp. with $t_{\text{min}} = 0.7$.

(a) Runtime and quality for DS-M2.

DS-M2	Incremental		Static	
	Base	Source	CLIP	SplitMerge
runtime (s)	4148	1093	1748 + 68	1748 + 659
precision	0.744	0.888	0.848	0.838
recall	0.844	0.826	0.815	0.822
F-measure	0.791	0.856	0.831	0.830

(b) Runtime and quality for DS-P1.

DS-P1	Incremental		Static	
	Base	Source	CLIP	SplitMerge
runtime (s)	614	217	107+101	107+752
precision	0.553	0.811	0.850	0.798
recall	0.836	0.817	0.821	0.851
F-measure	0.665	0.814	0.835	0.824

Table 6.5 shows the obtained runtime and F-measure results where the runtime of static entity clustering consists of the sum of the time to create the similarity graph and the time for clustering itself. We observe that source-specific incremental clustering reaches not only substantially better F-measure than the base incremental approach but also much better runtime due to the fast Max-Both approach to determine 1:1 assignments. The 1:1 assignments of source-specific incremental matching lead to dramatically improved precision over the base approach, especially for DS-P1 the precision increases from 55.3% to 81.1%; that more than outweighs the somewhat reduced recall thereby achieving a substantial improvement in F-Measure (85.6% vs. 79.1% for DS-M2 and 81.4% vs. 66.5% for DS-P1). At the same time, the improved quality is achieved by a much-lowered runtime by factor 3.8 for DS-M2 and factor 2.8 for DS-P1.

In comparison to static clustering, we expect better runtime and a reduced match quality for incremental clustering since we only optimize cluster assignment for a subset of entities at a time. For DS-M2, the time to generate the similarity graph for static entity clustering ($1748\,\text{s}$) is already higher than for the entire source-specific incremental clustering despite the use of blocking key length 4 and therefore small blocks. For DS-P1, we used blocking keys of length 6, which reduces the overhead for the graph generation for the static approach to only 107 s. Therefore, overall runtime for graph generation and CLIP (107 s + 101 s) is about the same as for the source-based incremental approach (217 s), while the static approach SplitMerge is slower by a factor of 4. The base incremental approach is mostly slower than the static approaches. However, the runtime for the incremental approaches is the sum for all source additions, while each incremental addition of entity sets and sources is typically faster than a complete static (re-)computation of clusters.

In terms of match quality, we observe that the base incremental approach is always inferior to the static approaches but that the source-specific incremental approach is actually better for DS-M2 (2.5% higher F-measure) and only slightly worse for DS-P1 (1.0-2.1% lower F-measure than SplitMerge and CLIP). We can thus conclude that the source-specific incremental clustering reaches comparable cluster quality than the static approaches with typically faster runtime and support for the dynamic addition of new entities and sources.

6.5 Related Work

Previous approaches to cluster entities (including connected components and correlation clustering) were mostly static and often considered only entities from a single source [77]. Static clustering of entities from multiple sources has been studied in [11, 131, 163, 164]. An overview of previous research for matching both schemas (ontologies) and entities from multiple sources for holistic data integration is provided in [155].

Relatively little work has been published on incremental entity resolution. Most previous approaches, including [35, 190], follow the straight-forward approach to consider new entities in isolation and add it to the cluster with the highest similarity above a threshold.

The incremental clustering approach of Gruenheid et al. [71] is more sophisticated. They determine and incrementally update a similarity graph to find the best cluster assignments. Their approaches are also able to repair previous cluster decisions. The evaluation focuses on changes for two single datasets of small size, thus leaving open the scalability to large datasets and several data sources, which is the focus of our work.

6.6 Conclusion

This chapter proposed and evaluated incremental approaches to cluster matching entities from multiple sources. In contrast to static entity clustering, the new approaches can continuously integrate additional sources and entities and avoid the expensive computation of similarity graphs for entities from many sources. To optimize the cluster assignments, we consider sets of new entities together. Especially promising is the source-specific incremental clustering determining a 1:1 assignment between entities and existing clusters using a Max-Both strategy. The evaluation for datasets from three domains showed that the proposed approaches are highly effective and efficient and often faster than with static entity clustering. The source-specific approach outperforms the base approach in both cluster quality and runtime and should thus be applied whenever possible.

With the presentation of holistic and incremental approaches, we already showed effective linking and clustering strategies in the last two Chapters 5 and 6. We thus applied methods to improve scalability on large datasets. This includes the avoidance of unnecessary comparison calculations between participating entities. Therefore, blocking strategies and the reuse of existing links from portals such as LinkLion (see Chapter 4) are essential.

Still, the constant growth and new data sources present a challenge to the continuous integration of knowledge bases. For this reason, the following chapters address strategies for distributed execution using modern Big Data frameworks.

Part III

Parallelization of Clustering Strategies

7 Introduction to Distributed Data Processing

This chapter gives an overview of developments regarding distributed data processing. First, the Apache MapReduce system is used to show basic concepts in Section 7.1. Subsequently, state-of-the-art distributed processing systems and the corresponding concepts are presented in Section 7.2. The decision on the distributed processing system for application in this work is then derived based on the collected information.

7.1 Background and Basic Concepts

Distributed processing frameworks apply ideas being already used in early models designed for parallel data processing such as bulk-synchronous processing [181] or parallel message-passing [68]. Thus, modern frameworks apply variations of the following components to enable computations on distributed data.

- Physically distributed processing units for parallel calculation/storage
- Exchange of messages between the processing units
- Possibilities to synchronize between processing units

This also applies to the MapReduce programming model, which was introduced in 2004 [39] and outperformed previous systems in terms of hardware abstraction and ease of use. At this time, it was the first system to support the processing of very large amounts of data on shared-nothing clusters consisting of commodity hardware. The following description of the concepts behind MapReduce provides an initial overview of distributed processing systems. In addition to strengths, weaknesses are also addressed to transition to current distributed data processing systems.

The MapReduce paradigm allows the processing of large amounts of data on concurrent computer systems. Its central concept is the use of the two higher-order functions *Map* and *Reduce* to execute arbitrary user-defined logic. Meanwhile, partitioning of data, scheduling execution, handling hardware failures and communicating between computers is handled by the underlying abstraction layer – the Apache Hadoop platform[1]. Typically, distributed file systems like Apache HDFS are used to maintain data, e. g., to enable desired features like redundancy and scalability. This includes immutability for all stored data collections, since writing a new version is less complex than changing existing data. Starting a MapReduce computation for the common example of counting words for multiple lines of text $(l_1, \ldots, l_n)$, the lines are automatically partitioned and distributed across the available machines. Then, each machine applies the user-defined *map* function (e. g., split words) on the initial key/value pairs (line number/list of words, e. g., $(l_1$, [example, words]$)$) to create intermediate key/value pairs (e. g., (example, 1), (words, 1)). The intermediate result is redistributed so that entries with the same key are on the same physical machine. These key/value pairs are then grouped on the key to apply the user-defined *reduce* function (e. g., sum up the word frequency) to form a final set of key/value pairs (e. g., the individual frequency for each word). Alternatively, the reduce function can combine complex values via custom merge or aggregation functions.

The ease of developing individual programs and providing the MapReduce implementation as part of Apache Hadoop has made a significant contribution to this popularity. The integration of functionality for implementing fault tolerance in the event of hardware failures and exploiting data locality ensures a higher level of performance and successful execution of calculations [39]. In summary, the strengths of MapReduce are

- *Abstraction layer* to handle distribution, storage, communication and hardware failures
- Efficient execution of calculations on *immutable collections of data*
- Mapping of all *user-defined logic to transformation functions* Map and Reduce.

Unfortunately, MapReduce has also disadvantages due to its architecture [112]. For example, the materialization of intermediate results between Map and Reduce function

[1]Apache Hadoop `https://hadoop.apache.org/`

leads to a decrease in efficiency, especially for MapReduce jobs that are executed iteratively or sequentially [108]. In addition, MapReduce cannot utilize global state information that would be useful for iterative algorithms. Thus, iterative processes have to write (and read) intermediate results over and over again from disk. Consequently, unnecessary resources are consumed for these I/O operations [112]. Due to these fundamental limitations, new general-purpose distributed processing frameworks have been developed. In the following section, two current approaches and underlying concepts are presented to point out improvements.

7.2 State of the Art

Current general-purpose distributed processing systems avoid disadvantages of predecessors while pushing new developments. Two well-known representatives of general-purpose distributed processing systems used in both industry and research are Apache Flink [27] and Apache Spark [194]. They are are designed for batch as well as continuous streaming workloads.

Data transformations already known from MapReduce are supplemented by established transformations from the domain of relational databases. Operations such as filter, join, union or group-by together with aggregation functions offer scope for efficient workflow processing. A brief overview on transformation operations is given in Table 7.1 along with minimal examples. Complex workflows therefore combine different transformations, which can be mapped to a directed acyclic graph (DAG). In this graph, vertices reflect transformations that are connected by edges indicating data flow. This form of modeling corresponds to the *dataflow paradigm* [41], which is used in various Big Data frameworks. For this reason, the term distributed dataflow system is also accepted.

When starting a dataflow program, transformations in the DAG are not executed directly due to the so-called *lazy evaluation*. Only the explicit triggering by specific commands, e. g., by writing the results into a data sink, leads to the actual evaluation of all necessary transformations. Depending on the execution environment, an execution plan is then created to make optimal use of the computing resources and to carry out any possible optimizations. Finally, data “flows” across the graph through transformations until it is stored in a data sink as a result.

An unnecessary and too frequent materialization of (intermediate) results is avoided by keeping results of the workflow in *in-memory data structures*. Therefore, datasets are always an immutable collection of in-memory data objects, which are distributed across

Table 7.1: Exemplary dataset transformation operations used in general-purpose distributed processing systems, e. g., for Apache Flink.

Operation	Description
Filter	Return all dataset elements for which the UDF returns `true`. `input.filter(udf: IN -> Boolean)`
(Flat)Map	Map- and FlatMap transformations apply a UDF on each element of the dataset. Map functions emit exactly one resulting element per input while FlatMap functions may emit arbitrary result elements (including none). Input `IN` and output type OUT do not need to be identical. `input.map(udf: IN -> OUT)`

Reduce	By executing a UDF, (possibly grouped) elements are combined to form a single value resp. multiple result values (for each group). One of many possibilities is a dataset grouped by specified keys to reduce/combine the key/value pairs as follows.
	`input.groupBy(keys)`
	`     .reduceGroup(udf: IN -> OUT)`

(Flat)Join	A Join transformation combines elements of two datasets with equal values on (tuple) key(s) and creates exactly one result element for Join and arbitrary result elements for the FlatJoin based on a UDF .
	`leftInput.join(rightInput)`
	`     .where(leftKey).equalTo(rightKey)`
	`     .with(udf: (left,right) -> OUT)`

CoGroup	Similar to the join operator, elements from two datasets are grouped on a key. Matching elements are processed in the UDF by iterating over both result groups.
	`leftInput.cogroup(rightInput)`
	`     .where(leftKey).equalTo(rightKey)`
	`     .with(udf: (left,right) -> OUT)`

Union	Create the union of two datasets having the same type. `leftInput.union(rightInput);`
Aggregate	Aggregations like `sum`, `min`, `max` or UDFs can be applied to grouped datasets `input.aggregate(SUM, key);`

the computer cluster. The results of transformations are always immutable datasets again.

A central job scheduler distributes the operators of the dataflow graph to the individual workers in the cluster. The job scheduler also coordinates the handling of execution errors via checkpoints and recovery options. The data shuffling required for many operators such as join is handled directly between individual workers via a network connection. Results can be materialized into various data sinks, e. g., HDFS, Apache Kafka or custom output formats. In addition to the respective basic functionality, Spark and Flink offer optimized engines for the execution of relational SQL-like operators, machine learning and graph processing algorithms.

In summary, current distributed processing systems have significant advantages over previous systems, such as

- Support for batch and streaming workload,
- Extended set of transformation operators,
- Efficient use of dataflow paradigm,
- Lazy evaluation of transformations in the DAG and
- In-memory data structures without (unnecessary) materialization.

7.2.1 Comparison of Distributed Processing Systems

When comparing the features of Apache Spark and Apache Flink, the two systems are similar in many ways. Both support the characteristics summarized in the last section, whereby the individual realizations differ. For example, Flink and Spark diverge in the implementation of streaming concepts. Apache Flink uses its internal streaming engine and defines transformation operators that can process streaming data at event time. For batch processing, Apache Flink provides the DataSet API as a special case of stream processing, allowing to apply transformation operators to the entire dataset as well as optimizations, e. g., for the execution order of the operators [27]. By contrast, Spark's streaming approach collects streaming data for a short time interval and performs a minimalistic batch computation [194]. The Flink DataSet API enables the optimized execution of operators in the DAG. A cost-based approach selects the best execution plan from several options to optimally arrange grouping and join operators [27]. Apache

Spark provides similar operator optimization functionality with the DataFrame API but is limited to relational operators [4]. Due to this limitation, the complex workflow for data integration and clustering with custom logic cannot be adequately covered by the DataFrame API.

Both systems support additional libraries to provide functionality such as graph processing, machine learning or features regarding relational operators. In complex general-purpose workflows, these additions allow the use of specialized operators. Still, it is necessary to consider the share of specialized operators, since, e. g., systems optimized for graph-processing or machine-learning deliver higher performance with a suitable workload [104, 193]. In performance comparisons of general-purpose distributed processing systems, both Apache Spark and Apache Flink achieve good results [64, 98, 116, 183] for a broad set of algorithms, including graph algorithms. In [98], execution times of graph algorithms like PageRank [24] and the semi-clustering algorithm of Pregel [119] are compared within the graph libraries GraphX (Apache Spark) and Gelly (Apache Flink). For the clustering algorithm, Flink improves the runtime by factor 5.1 for the largest dataset with 3.2×10^6 edges. Similarly, the runtime results for PageRank are consistently faster using Apache Flink. Another study [183] is evaluating six algorithms (including three iterative ones) on Apache Spark, Flink and Hadoop. The authors conclude that both Apache Spark and Apache Flink deliver significant performance gains over Hadoop MapReduce. Compared to Spark over Flink, however, the former is more mature and provides better performance on average. Through native support for iterative algorithms, Apache Flink can achieve up to 3.6x better runtime than Apache Spark for algorithms like PageRank. Li et al. [116] show mixed results for the comparison of different algorithms. In detail, Apache Spark performs better for WordCount, linear regression and ALS, while Apache Flink shows better results for PageRank and SSSP. The frameworks also have critical dependencies towards the corresponding system configuration and the used datasets [98, 183]. This is the case for both Apache Spark and Apache Flink meaning that new releases can bring significant performance changes or necessary configuration adjustments.

The continuous development of both systems allows the introduction of new features and the revision of existing ones. For example, Apache Spark improves the support for streaming workload and adds a stronger integration of deep learning frameworks[2]. In contrast, Apache Flink takes steps to unify the batch and streaming APIs and intro-

[2]Apache Spark release notes version 2.3.0 (https://spark.apache.org/releases/spark-release-2-3-0.html) and 2.4.0 (https://spark.apache.org/releases/spark-release-2-4-0.html)

duces new features for streaming, such as managed state[3] or transactions with ACID support [38].

7.2.2 Apache Flink

The implementation of the data integration workflows in the following Chapters 8 and 9 relies on Apache Flink [27]. This section describes criteria and the Apache Flink features that led to the framework decision and shows the mapping of the data model to the data structure. For the integration of different Linked Data sources, batchwise processing with the Apache Flink DataSet API using general transformations (Table 7.1) is preferred over stream processing. The immutable, in-memory and distributed data structures are called DataSets and provide data for transformations within Flink programs. Transformations manipulate DataSets by applying user-defined functions with application logic. The entire set of transformations then spans the DAG, which is optimized before execution. The optimizer selects the best execution plan for the DAG with a cost-based estimation, e. g., network, I/O and CPU cost. This includes reusing data partitioning and sort orders for successive transformations such as joins as well as chained execution for successive (map) functions for better performance[4]. Depending on the physical data distribution, data needs to be shuffled across cluster nodes to execute transformations such as join or reduce. Still, the network traffic for data shuffling can be reduced by various techniques. For example, instead of a reducer, a combiner transformation can be used to pre-aggregate data. Therefore, when possible, elements are already combined locally in the context of each worker.

From the set of libraries providing additional functionality, we use the graph processing engine Gelly in our approaches. In particular, we employ Gelly graphs containing a DataSet of vertices and edges

```
class Graph<K, VV, EV> graph {
    DataSet<Vertex<K, VV>> vertices;
    DataSet<Edge<K, EV>> edges;
}.
```

The complex data types `Vertex` and `Edge` are inherited from the Flink `Tuple` classes `Tuple2<K,VV>` (type K as vertex ID, VV as vertex value), `Tuple3<K,K,EV>`

[3]Apache Flink release notes version 1.7.0 (https://flink.apache.org/news/2018/11/30/release-1.7.0.html) and 1.8.0 (https://flink.apache.org/news/2019/04/09/release-1.8.0.html

[4]Optimizer Internals Apache Flink https://cwiki.apache.org/confluence/display/FLINK/Optimizer+Internals

(source vertex ID, target vertex ID (each type K) and EV as edge value), respectively. Operators like *join*, *filter* or *group-by* rely on key expressions or tuple positions (starting from 0), e. g.,

```
vertices.join(edges)
 .where(0).equalTo(1)
 .with((vertex, edge) -> new Tuple1<>(edge.getSimilarity))
 .filter(tuple -> tuple.f0 >= 0.9);
```

This expression joins all edges with the vertices where the vertex ID (tuple position 0 in `vertices`) equals the target ID of the edge (tuple position 1 in `edges`). It then returns the similarity value if the accompanied filter function is successfully evaluated. Using Gelly's graphs, the structures from the data model (Section 2.2) can be easily transferred into vertices and edges. Entities $\mathcal{E}$ correspond to Gelly vertices and links $\mathcal{L}$ with their properties are handled as Gelly edges. Similarly, clusters $\mathcal{C}$ and representatives $\mathcal{R}$ are mapped to Gelly vertices while potential similarities between clusters resp. representatives are depicted by Gelly edges.

Additional programming abstractions provide efficient iterations that can transfer state information from one iteration to the next. In the special case of DeltaIteration, the set of elements for the next iteration is kept separate from the set of already finalized elements. This procedure means that subsequent iteration steps in rapidly converging procedures require decreasing numbers of calculations. Complete recalculations in each iteration step can thus be avoided.

By using Gelly, we also benefit from abstract graph processing operators like graph neighborhood aggregations or operators that apply the above-described iterations to graphs. Efficient iterations are an important factor for distributed graph processing. Several programming abstractions have been developed to solve algorithmic problems, such as vertex-centric, scatter-gather, gather-sum-apply [99]. These programming abstractions are also supported by the graph processing library Gelly which is included in Apache Flink. Compared to the equivalent GraphX in Apache Spark (vertex-centric iteration only), Gelly offers more flexibility in choosing the appropriate model [98].

The implementation of the concepts for holistic and incremental clustering (Chapters 5 and 6) for physical data integration on Apache Flink therefore employs various (graph processing) operators from Flink and Gelly. In particular, we use the delta iteration in different variations, e. g., vertex-centric iteration (Pregel [119]), gather-sum-apply computation (PowerGraph [67]) or custom implementations.

8 Distributed Clustering Strategies

Preamble

The major part of this chapter is based on [130] and describes the distributed implementation of the SplitMerge clustering approach. It covers all steps of the complex data integration process leading to the final cluster representation. Accordingly, Apache Flink and Gelly specific algorithms such as vertex-centric iteration are used and illustrated with examples. However, results for scalability are presented for both SplitMerge [131] clustering and incremental clustering [133]. The distributed holistic clustering [130] was presented at the ODBASE conference in 2017 and was published in the respective conference proceedings of the OTM 2017.

8.1 Motivation

Linking entities from various sources and domains is one of the crucial steps to support data integration in the Web of Data. A manual generation of links is very time-consuming and nearly infeasible for the large number of existing entities and data sources. As a consequence, there has been much research effort to develop link discovery frameworks [132] for automatic link generation. Platforms like `DataHub`[1] or `<sameAs>`[2] or repositories such as LinkLion [134] collect and provide large sets of links between numerous different knowledge sources. They can be reused to avoid an expensive re-

[1]DataHub platform `https://datahub.io`
[2]sameAs platform `http://sameas.org`

determination of the links. It is particularly complex to ensure high link quality, i. e., the generation of correct and complete link sets. Existing link repositories cover only a small number of inter-source mappings and automatically generated links can be erroneous in many cases [52]. Despite the huge number of sources to be linked, most link discovery tools focus on a pairwise (binary) linking of sources. However, link discovery approaches need to scale for n-ary linking tasks as well as for an increasing number of entities and sources that are added to the Web of Data over time [155].

To address these shortcomings, we recently proposed the SplitMerge approach to cluster Linked Data entities from multiple data sources into a holistic representation with unified properties [131]. The method combines entities that refer to the same real-world object in one compact cluster instead of maintaining a high number of binary links for k sources. The approach is based on existing `owl:sameAs` links and can deal with entities of different semantic types as they occur in many sources (e. g., for geographical datasets, countries, cities, lakes). Input links are checked for consistency and new links (e. g., for previously unconnected sources) are identified. We also proposed an approach to handle the addition of new entities or data sources to existing knowledge bases [133], to support the integration of dynamically changing data sources.

Considering the huge size and number of sources to be linked, scalability becomes a major issue. Linking and clustering approaches usually comprise complex operations such as similarity computations to identify entities or clusters according to a given relation. These complex work steps can often be parallelized using distributed Big Data execution frameworks such as Apache Spark or Flink [27] to reduce execution time significantly. With regard to the ever-increasing amount of data to be linked and integrated in typical Big Data processing workflows, scalable solutions for link discovery and holistic entity clustering are essential.

A further and long-standing problem is the poor availability of reference datasets that can be used as a gold standard for quality evaluations. Since link discovery usually focus on pairwise matching, the few existing benchmarks cover links between two data sources. However, there is an increasing need to evaluate the effectiveness of multi-source clustering, as holistic data integration scenarios like the construction of knowledge graphs gain increasing research interest [46, 146]. The creation of new reference datasets for quality evaluation of n-ary linking and clustering approaches can be useful for the community and support the development of improved holistic clustering approaches.

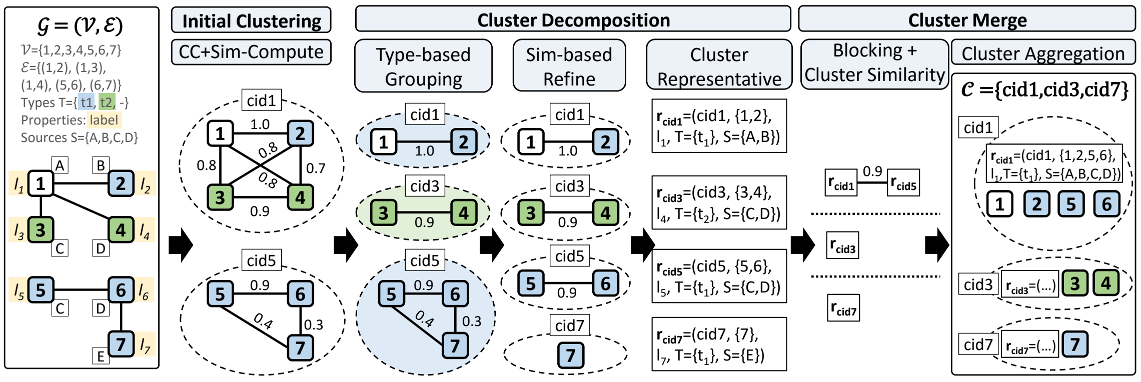
Herein we study distributed clustering approaches for SplitMerge and incremental clustering. In contrast to the previous work, we support blocking strategies to reduce unnecessary comparisons and present a comprehensive evaluation of quality and efficiency on real-world data for different domains. An extended version with more details on Flink implementation and creation of a reference dataset can be found in [129]. This chapter provides the following contributions:

- We present distributed clustering approaches for Linked Data to enable an effective and efficient clustering of large entity sets from many data sources.
- We provide a novel reference dataset for multi-source clustering from the geographic domain. We evaluate the effectiveness of our approach with the new gold standard and a further artificial dataset from the music domain.
- We further evaluate the efficiency and scalability of the distributed implementation of the SplitMerge and incremental clustering approaches for very large datasets with millions of entities from various domains.

The remainder of this chapter is organized as follows. After a description of the problem statement in Section 8.2, the implementation of the distributed clustering approaches is presented in Sections 8.3 and 8.4. The clustering gold standard is provided in Section 8.5, together with evaluation results for both distributed clustering approaches. Finally, we discuss related work in Section 8.6 and conclude.

8.2 Problem Statement

In accordance with the data model from Section 2.2, the goal of the holistic clustering approaches is to cluster same real-world entities from different data sources. For an efficient determination of clusters for large datasets, the batch processing part of Apache Flink [27] is employed. The execution of the workflow is parallelized on a shared-nothing cluster, while computations are based on in-memory data structures. By using the graph processing engine Gelly, entities $\mathcal{E}$ and clusters $\mathcal{C}$ can be mapped to Gelly vertices, and accordingly, links are mapped to Gelly edges. For the sake of clarity, the common Gelly terms vertices and edges are used in this chapter rather than entities and links. Our implementation employs the Flink DataSet API with data transformations like filter, join, union, group-by or aggregations (relational databases) and map, flat-map and reduce (MapReduce paradigm). As described in Section 7.2.2, we additionally utilize abstract Flink and Gelly operators, e. g., to perform efficient iterative computations. Intermediate results are represented as vertices, edges and graphs, which can be written to disk, e. g., in JSON format. Within complex transformations, unneeded properties are removed using the Flink `TupleX` representation instead of `Vertex` or `Edge` to reduce the amount of network traffic and memory consumption.



G = (V, E)
V={1,2,3,4,5,6,7}
E={(1,2), (1,3), (1,4), (5,6), (6,7)}
Types T={t1, t2, -}
Properties: label
Sources S={A,B,C,D}
Initial Clustering
CC+Sim-Compute
Cluster Decomposition
Type-based Grouping
Sim-based Refine
Cluster Representative
Cluster Merge
Blocking + Cluster Similarity
Cluster Aggregation
C ={cid1,cid3,cid7}
r_cid1=(cid1, {1,2}, l1, T={t1}, S={A,B})
r_cid3=(cid3, {3,4}, l4, T={t2}, S={C,D})
r_cid5=(cid5, {5,6}, l5, T={t1}, S={C,D})
r_cid7=(cid7, {7}, l7, T={t1}, S={E})
r_cid1=(cid1, {1,2,5,6}, l1,T={t1}, S={A,B,C,D})

Figure 8.1: Running example for distributed SplitMerge clustering workflow.

8.3 Distributed SplitMerge Clustering

In this section, we discuss the transformation and adaptation of the SplitMerge workflow towards a distributed processing workflow. From a high-level perspective, entities and links are loaded into a Gelly graph to generate clusters using a set of transformation operators. We illustrate the workflow steps using the running example in Figure 8.1. There are six input edges $\mathcal{E}$ and seven input vertices $\mathcal{V}$ further described by a label (l_1, l_2, ...), the originating data source from S and colored dependent on their semantic type ($t1$, $t2$ or no type).

8.3.1 Preprocessing

In preprocessing, we apply several user-defined functions on the input graph, e. g., to harmonize semantic type information, remove inconsistent edges and vertices and normalize the label property value. First, we compute similarities for the given input edges based on vertex property values. For each vertex, we carry out a consistency validation using grouping on adjacent vertices and associated edges, and further remove neighbors with equal data sources (for details see Chapter 5). We omit the preprocessing step in the example (Figure 8.1) and directly start with the preprocessed input graph $\mathcal{G}$.

8.3.2 Initial Clustering

To determine initial clusters, we compute the connected components (CC) within $\mathcal{G}$ and assign a cluster ID to each vertex. In the example, vertices $1-4$ obtain cluster IDs *cid1* and vertices $5-7$ *cid5*. Intra-cluster edges are then generated within each cluster accompanied by a similarity computation based on properties such as a linguistic similarity on labels or normalized geographical distance.

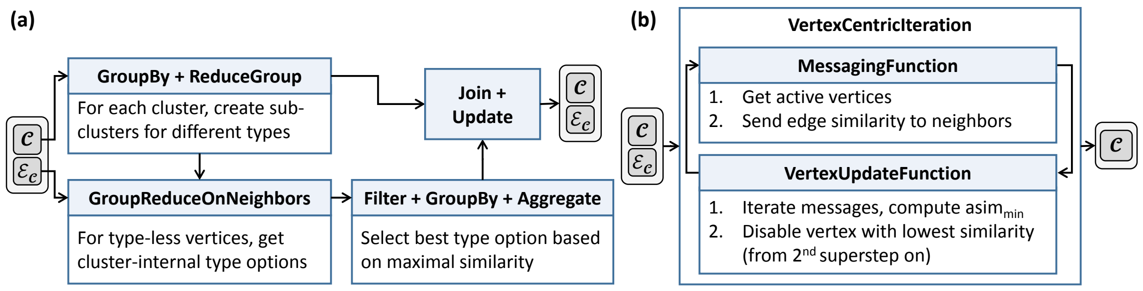

(a)
GroupBy + ReduceGroup
For each cluster, create sub-clusters for different types
GroupReduceOnNeighbors
For type-less vertices, get cluster-internal type options
Join + Update
Filter + GroupBy + Aggregate
Select best type option based on maximal similarity
(b)
VertexCentricIteration
MessagingFunction
1. Get active vertices
2. Send edge similarity to neighbors
VertexUpdateFunction
1. Iterate messages, compute asim_min
2. Disable vertex with lowest similarity (from 2nd superstep on)

Figure 8.2: Sub-workflows with operators for type-based grouping (a) and similarity-based refinement (b).

8.3.3 Cluster Decomposition

Type-based grouping is the first part of the decomposition to split clusters into subcomponents dependent on the compatibility of semantic types. Figure 8.2a shows the sequence of applied transformations and short descriptions. Within clusters, a ReduceGroup function assigns new cluster IDs based on semantic types, e. g., vertex 3 and 4 are separated from vertex 2. Vertices without a type (like vertex 1) require special handling. We apply GroupReduceOnNeighbors (a Gelly CoGroup function to handle neighboring vertices and edges) to produce tuples for vertices with missing semantic type. Vertex 1 creates a Tuple `(id,sim,type,cid)` for each outgoing edge ($(1,2), (1,3), (1,4)$), namely `(1, 1.0, t_1,cid1)` for edge $(1,2)$ and `(1, 0.8, t_2, cid2)` for edge $(1,3)$ and $(1,4)$. Grouping on the vertex ID executes an aggregation function for each group to return the tuple with the highest similarity per vertex, which is `(1, 1.0, t_1, cid1)` for vertex 1. Processed vertices update their cluster ID accordingly (e. g., vertex $1 \rightarrow cid1$). The result of the type-based grouping is a set of clusters with intra-cluster edges.

Similarity-based refinement is the second part to decompose clusters by removing non-similar entities from clusters. We use a Gelly vertex-centric iteration to exchange messages between vertices and update the vertices accordingly (see Figure 8.2 b for details). At the end of each iteration (also called superstep), the process is synchronized between all computer nodes. In the first superstep, all vertices are active and send messages to all their neighbors. Messages are tuples containing the originating id, the edge similarity and an average edge similarity $asim$ of all incoming messages (0 in the first superstep). Starting with the second superstep, we illustrate the sent messages for vertex 7 in cluster $cid5$ in our example: vertex 5 sends `(5, 0.4, 0.65)` to 7, vertex 6 sends `(6, 0.3, 0.6)` to 7 and vertex 7 sends messages to 5 and 6, resulting in $asim = (0.4 + 0.3)/2 = 0.35$ for vertex 7. Now in each cluster, the vertex with the lowest $asim$ is deactivated (and excluded from the cluster), given that this $asim$ is be-

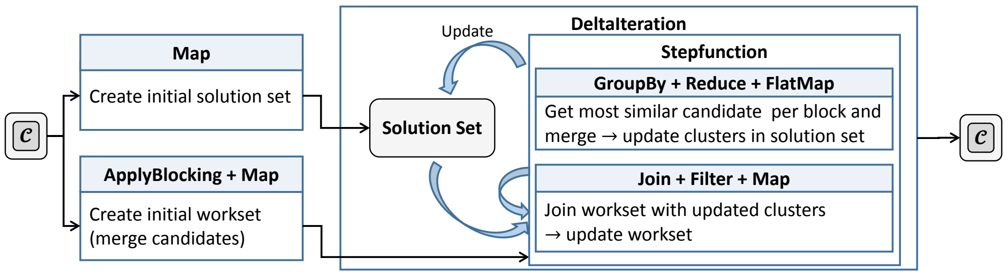

Map
Create initial solution set
ApplyBlocking + Map
Create initial workset (merge candidates)
Update
Solution Set
DeltaIteration
Stepfunction
GroupBy + Reduce + FlatMap
Get most similar candidate per block and merge → update clusters in solution set
Join + Filter + Map
Join workset with updated clusters → update workset

Figure 8.3: Sequence of transformations for the cluster merge using Flink DeltaIteration.

low a certain similarity threshold. In Figure 8.1, vertex 7 deactivates and is isolated into cluster $cid7$. Vertices send only messages if they are updated and deactivated vertices never send messages again; therefore, iteration termination is guaranteed.

Finally, we create a unified *Cluster Representative* for each cluster based on contained entities. Aggregation of property values is used for covered data sources and semantic types as well as the selection of best label or geographic coordinates, see Figure 8.1.

8.3.4 Cluster Merge

For the final merge phase in our distributed holistic clustering workflow, we use the Flink DeltaIterate operator together with user-defined functions. The main operators are sketched in Figure 8.3. During merge, we iteratively aggregate highly similar (likely small) clusters into larger ones. With the creation of representatives for each cluster, we already reduced the number of entities for the merge step. Since we potentially compare every cluster with every other cluster, the quadratic complexity can become a problem for very large cluster sets. Thus we employ blocking strategies and avoid unnecessary comparisons. Currently, we implement standard blocking on specified property values, such as using the first letters of the label as a blocking key. We avoid further unnecessary comparisons since we do not compare representatives with incompatible semantic types and check for already covered data sources since we assume duplicate-free data sources. Our example in Figure 8.1 shows three blocks resulting from the blocking step: r_{cid1} and r_{cid5} need to be compared, such that a triplet $(r_{cid1}, 0.9, r_{cid5})$ is created as a merge candidate. For r_{cid3}, no merge candidate is created because there is no other representative with type t_2, whereas r_{cid7} is in a separate block due to its dissimilar label compared to other representatives.

The delta iteration starts with an initial solution set containing the previously determined clusters and an initial workset (merge candidates), as seen in Figure 8.3. Within each iteration of DeltaIterate, a custom step function updates the current workset to generate changes for the current solution set. The next workset and the updates to the solution set are then passed to the next iteration step. In detail, the workset is grouped by the blocking key and for each block, the triplet with the highest similarity exceeding the minimum similarity is selected using a custom Reduce function. For our running example, $(r_{cid1}, 0.9, r_{cid5})$ is the best merge candidate and therefore merged within a custom FlatMap function. The new cluster r_{cid1} contains combined values for properties like sources $S = \{A, B, C, D\}$, the list of contained vertices ($\{1, 2, 5, 6\}$) and unified properties like the label l_1. This directly affects the cluster representatives in the solution set, and the already merged cluster r_{cid5} is deactivated in the solution set. Now the merge candidates within the workset are adapted based on changed clusters within the iteration step (see Figure 8.3 solution set). The appropriate new cluster r_{cid1} replaces the deactivated cluster representatives and merged triplets ($(r_{cid1}, 0.9, r_{cid5})$) as well as generated duplicate triplets are removed from the workset. Again, triplets are discarded if the data sources for the participating clusters overlap or exceed the maximum possible number of covered sources. The delta iteration ends either when the workset is empty (default, and true for our running example after the first iteration) or a maximum number of iterations took place. Note, that for larger datasets, parts of the dataset may converge faster to a solution, when clusters can not be merged anymore. These parts are not recomputed in following iterations, such that only smaller parts of the data are processed.

8.4 Distributed Incremental Clustering

The concept of incremental clustering and the distributed workflow is described in Chapter 6. This section deals with implementation details related to the distributed execution of the incremental clustering substeps on Apache Flink.

First, the entities and possibly existing clusters are distributed to the individual workers in the cluster by means of blocking. For this purpose, we apply the blocking strategy Block Split [106], which is optimized for shared-nothing clusters. This allows load balancing to be used to distribute skewed blocks to several machines automatically. Within the parallel generation of cluster candidates based on connected components, the source-consistency and a minimal similarity between candidates are ensured. From this point on, two different incremental methods realize a base approach and a source-specific approach.

The base approach tries to achieve the optimal assignment of entities for each connected component to the existing clusters. For this purpose, a Gelly GroupReduceOn-

Neighbors function is used to enrich links with useful information from the neighborhood. In particular, this includes information about the source affiliation of the entities to determine an optimal cluster assignment together with the similarities. Sorting the links by descending similarity in the next Reduce Function allows to select the best links that meet the conditions, e. g., regarding source-consistency. Selected links lead to a cluster assignment and at the same time to the removal of the corresponding entity from the set of cluster candidates to avoid duplicate assignments. A final Reduce function for the base approach creates new cluster representatives based on connected components.

The source-specific approach aims to achieve a 1:1 assignment of the generated candidates, for which Max-Both is used. Based on the similarities between the candidates, Max-Both calculates two sets of individual top links for both the new entities and the existing clusters. The parallel execution is handled by basic Flink operators to group and select the maximum similarity per entity. The intersection of both top link sets is determined by a join operator and post-processed by a user-defined function. Finally, similar to the base approach, cluster representatives are adapted to complete the assignment.

8.5 Evaluation

We evaluate the distributed clustering approaches w.r.t. effectiveness and efficiency for datasets from three domains. First, we describe details of the used datasets (Section 8.5.1) and present a novel reference dataset for multi-source clustering, including its creation process (Section 8.5.2). Then, the effectiveness of the distributed clustering approaches is evaluated in Section 8.5.3. Based on large datasets from various domains, we also analyze the efficiency and scalability of the distributed clustering approaches.

8.5.1 Datasets

To evaluate the distributed clustering approaches, we utilize seven datasets of different sizes from the geographic, music and person domain. Table 8.1 shows the available property values for entities, the number of covered entities and sources, as well as the number of correct links and clusters in the reference mappings. We use datasets DS-G1 and DS-M1 to evaluate the quality of entity clusters generated by the distributed Split-Merge clustering while DS-G2, DS-M2 and DS-M3 are used to analyze the efficiency and scalability of the approach (see Section 8.5.3). For distributed incremental clustering, we extend the evaluation of Chapter 6 and focus on large-scale datasets. Therefore, DS-M2 and DS-P1 are employed to describe qualitative results, while both person datasets are used to show scalability and execution times (see Section 8.5.3). In the following, we

Table 8.1: Overview of evaluation datasets. Number of resulting clusters and deduced correct links are given for reference datasets.

	domain	entity properties	#entities	#src	#correct links	#clusters
DS-G1	geography	label, semantic type,	3,054	4	4,391	820
DS-G2		longitude, latitude	1,537,243	5	-	-
DS-M1	music	artist, title, album,	19,375	5	16,250	10,000
DS-M2		year, length,	1,937,500	5	1,624,503	1,000,000
DS-M3		language, number	19,375,000	5	16,242,849	10,000,000
DS-P1	person	name, surname,	5,000,000	5	3,500,840	3,331,384
DS-P2		suburb, postcode	10,000,000	10	6,625,848	14,995,973

briefly introduce the datasets for the given domains before we describe the novel reference dataset for multi-source clustering in Section 8.5.2. All datasets can be retrieved from the Database Group Leipzig website[3].

Geographic Domain

We use two datasets (DS-G1, DS-G2) from the geographic domain, covering entities from the data sources DBpedia, GeoNames, NY Times, Freebase for DS-G1 and additionally LinkedGeoData for DS-G2. Entities for both datasets have been enriched with properties like entity label, semantic type and geographic coordinates by using provided SPARQL endpoints or REST APIs. DS-G1 is based on a subset of existing links provided by the OAEI 2011 Instance Matching Benchmark[4] that is also a subset of DS-G2. The 3,054 entities in DS-G1 create a novel reverence dataset for multi-source clustering (see Figure 8.4 a, dataset creation described in Section 8.5.2). Dataset DS-G2 covers about 1.5 million entities from five sources (see Figure 8.4 b) and originates from the link repository LinkLion [134]. We reuse about 1 Mio existing `owl:sameAs` links from LinkLion as input for the clustering. However, there is no reference dataset available to evaluate the quality of created clusters for dataset DS-G2. We use DS-G2 to evaluate the scalability of our approach for very large entity sets.

[3]Benchmark datasets https://dbs.uni-leipzig.de/de/research/projects/object_matching/benchmark_datasets_for_entity_resolution

[4]http://oaei.ontologymatching.org/2011/instance/

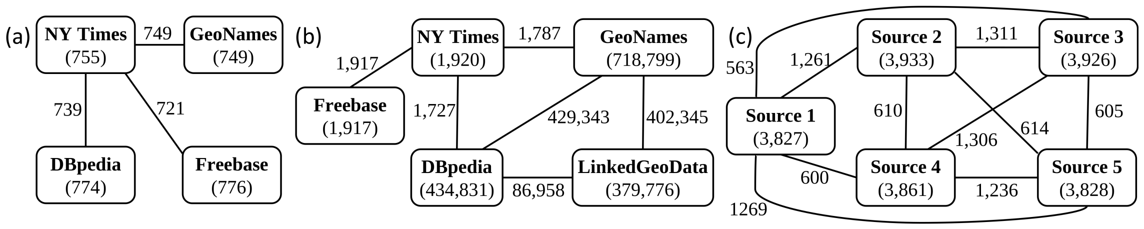


Figure 8.4: Exemplary dataset structures for DS-G1 (a), DS-G2 (b) and DS-M1 (c) with number of entities and links.

Music Domain

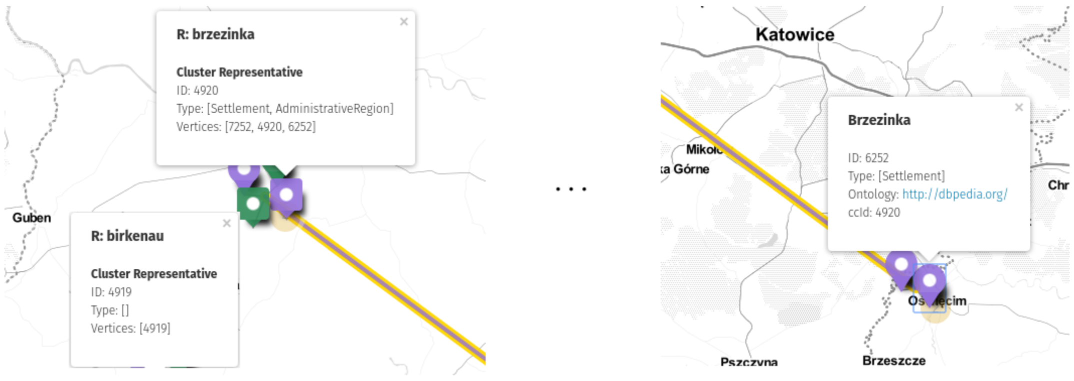
The publicly available Musicbrainz dataset covers artificially adapted entities to represent entities from five different data sources [82][5]. Every entry in the input dataset represents an audio recording and has properties like title, artist, album, year, language and length. The property values have been partially modified and omitted to generate a certain degree of unclean data and duplicate entities that need to be identified. This includes format changes for properties like year (e. g., ′06, 06 or 2006) and song length (2m 4sec, 02:04, 124000 or 2.0667).

The artificially generated datasets cover between 19,375 and 19,375,000 tuples (see Table 8.1, DS-M1 to DS-M3). Each of the datasets contains a fixed proportion for each cluster size: cluster size 1 (50 %), size 2 (25 %), size 3 (12.5 %), size 4 and 5 (6.25 %, resp.). This means that, for instance, 12.5 % of all entities can be assigned to clusters of size 3. Furthermore, cluster size 3 is the equivalent of three same-as entities from different sources due to the source-consistency. Besides a set of artificially created duplicates, each dataset covers cluster IDs from which links between entities, that refer to the same object, can be easily derived. Resulting clusters cover between about 16,000 and 16,000,000 links and up to 10 million clusters.

Person Domain

The two datasets from the person domain (see Table 8.1, DS-P1 and DS-P2) are based on publicly available voter data from North Carolina. This voter data was synthetically duplicated and the contained attribute values name, surname, suburb and postcode were partially corrupted. Thus five (DS-P1) resp. ten (DS-P2) data sources with up to 10,000,000 entities were generated, which can be used for holistic clustering. Although DS-P2 does not have the largest number of entities, it has great complexity due to the

[5]Musicbrainz test data https://vsis-www.informatik.uni-hamburg.de/oldServer/teaching//projects/QloUD/DaPo/testdata/


R: brzezinka

Cluster Representative
ID: 4920
Type: [Settlement, AdministrativeRegion]
Vertices: [7252, 4920, 6252]

R: birkenau

Cluster Representative
ID: 4919
Type: []
Vertices: [4919]

Brzezinka

ID: 6252
Type: [Settlement]
Ontology: http://dbpedia.org/
ccId: 4920

Figure 8.5: Visualization of an incorrect cluster for two different settlements named "Brezinka" and "Birkenau" connected by an incorrect same-as link. Cluster representatives are illustrated by rectangles (on the left), while vertices are shown as circular pins (on the right).

largest number of data sources and most clusters in the perfect result (nearly 15 million clusters).

8.5.2 Reference Dataset for Multi-source Clustering

We created a new manually curated reference dataset for multi-source clustering to support the evaluation of the quality for generated clusters in holistic clustering approaches. Available benchmarks usually only contain links between two data sources. We provide a gold standard based on real-world data from the geographic domain and make it available for other researchers. The reference dataset covers the input dataset and the perfect cluster result as JSON files and can be downloaded on the project site[6].

The reference dataset is a selection of entities with the semantic type "settlement" from the location subset of the OAEI 2011 Instance Matching Benchmark. We made a manual selection decision for vertices using available properties and edges. We further checked the correctness of semantic types. For instance, the vertex for "Canary Islands" was removed since its correct type in the geographic dataset should be "island" instead of "settlement". Removing a vertex resulted in the deletion of all its associated edges. For manual curation, we visualized the data using an open-source geographic map tool[7] (see Figure 8.5). To determine the perfect clusters and check for the correctness of the input data, we have two views on the data. First, the original links are represented as (thick) yellow links. The second view shows the resulting clusters created by SplitMerge clustering. Colored circular pins represent actual vertices with their properties, while rectangular pins illustrate the newly determined cluster representatives with label, type and vertices within the cluster. The pin color distinguishes different clusters.

[6]https://dbs.uni-leipzig.de/research/projects/linkdiscovery
[7]uMap https://umap.openstreetmap.fr/en/, import data as GeoJSON

In particular, correct clusters were defined based on the results of the SplitMerge clustering. In case of doubt, Wikipedia was used as an additional knowledge source about certain geographical entities. For each cluster, we decided whether it is correct or needs to be deleted, split into several clusters or merged with another cluster. During the manual curation, all clusters of maximum size (w.r.t. the number of data sources) were correct. Modifications to obtain the perfect result included the addition of further vertices to a cluster or removing vertices from clusters. For example, in Figure 8.5, the original Birkenau/Brzezinka (german/polish name) cluster needs to be split into two clusters due to a wrong link between two distinct geographic places. In the original data, the cluster covered all four entities, while SplitMerge determined two clusters. The green pin labeled *Birkenau (Poland)* in NYTimes, as well as the purple pin next to it (labeled *Brzezinka* in GeoNames), should actually form one cluster. The two vertices of the southeast cluster are another settlement in Poland, which has the same name, *Brzezinka*. The example represents a difficult case for automatic clustering and linking due to very differing vertex labels. The resulting reference dataset covers 820 cluster representatives containing information on covered cluster vertices. Based on the cluster IDs, one can easily derive all intra-cluster links to obtain the set of all correct links.

8.5.3 Experimental Results

We now present evaluation results for the quality of the determined clusters and the scalability for the distributed implementations of SplitMerge as well as the incremental clustering for three domains.

Setup and Configurations

The experiments are carried out on a cluster with 16 workers, each of them equipped with an Intel Xeon E5-2430 6x 2.5 GHz, 48 GB RAM, 2x 4 GB SATA disks and 1 GBit Ethernet connection. The machines operate on OpenSUSE 13.2 using Hadoop 2.6.0 and Flink 1.1.2. All experiments are carried out three times to determine the average execution time.

To obtain input link datasets for SplitMerge clustering on the geographic dataset DS-G1, we applied link discovery methods on the input entities using three different configurations (config 1, config 2, config 3). All configurations compute similarities based on JaroWinkler on the entity label; configurations 2 and 3 additionally compute a normalized geographic distance similarity below a maximum distance of 1358 km. Config 1 applies a minimal similarity threshold of 0.9 for labels while configs 2 and 3 apply threshold 0.85 and 0.9 for the average label and geographic similarity, respectively. Links for

Table 8.3: SplitMerge evaluation of cluster quality for geography dataset DS-G1 based on precision (P), recall (R) and F-measure (F1).

	config 1			config 2			config 3		
	P	R	F1	P	R	F1	P	R	F1
Input links	0.933	0.806	**0.865**	0.964	0.938	0.951	0.981	0.799	0.881
Best (Star1, Star2)	0.863	0.844	0.853	0.963	0.941	**0.952**	0.951	0.838	0.891
SplitMerge	0.903	0.824	0.862	0.913	0.919	0.916	0.968	0.836	**0.897**

DS-G2 were extracted from the LinkLion [134] repository, and have been computed by different link discovery tools from the community.

For the music dataset *DS-M1*, we created input links using a soft TF/IDF implementation weighted on title (0.6), artist (0.3) and album (0.1) with a threshold of 0.35. *DS-M2* and *DS-M3* are used to show scalability; therefore, we create edges based on the cluster ID from the perfect result by linking the first entity of each cluster with all its neighbors.

Distributed SplitMerge Clustering

The achieved *cluster quality* is analyzed for the geographic dataset DS-G1 based on precision, recall and F-measure. We first use existing input links from the considered subset in the original OAEI dataset (see Figure 8.4 a). This manually curated benchmark achieves a precision of 100 %. However, many links between certain data sources are missing leading to a reduced recall of only 50 % and an F-measure of 66.7 %. With SplitMerge, we achieve very good results w.r.t. recall (97.1 %) while preserving a good precision (99.8 %), resulting in the F-measure of 98.5 %. This shows that our approach produces high-quality clusters based on existing input links, thereby finding many new links.

However, as input mappings are not perfect in real-world situations, we used automatically generated input links for three linking configurations (config 1-3), as described above. To evaluate the cluster quality, we further provide a comparison to the recently published results from [163]. The work implemented several existing clustering algorithms. Here, we only select the respectively best results achieved with two versions of Star clustering [6] (Star1, Star2). It is important to note that in contrast to SplitMerge, Star clustering creates overlapping clusters. Thus clusters may contain duplicates. Besides, Star clustering does not create a compact cluster representation. Table 8.3 shows results for the cluster quality for the computed input links, the best result of (Star1, Star2) and our approach. SplitMerge clustering (F-measure 86.2 %) nearly retains the input link quality (86.5 %) for config 1, while best(Star1, Star2) achieves slightly worse results. For config 2, the Star2 implementation achieves a slightly better F-measure (95.2 %) com-

Table 8.4: SplitMerge evaluation of cluster quality for music dataset DS-M1 w.r.t. precision, recall and F-measure.

	Precision	Recall	F-Measure
Input links	0.835	0.783	0.808
SplitMerge	0.890	0.861	**0.876**

pared to the input mapping (95.1 %). For config 3, SplitMerge clustering improves the quality of the input mapping by 1.6 % w.r.t. F-measure (89.7 %).

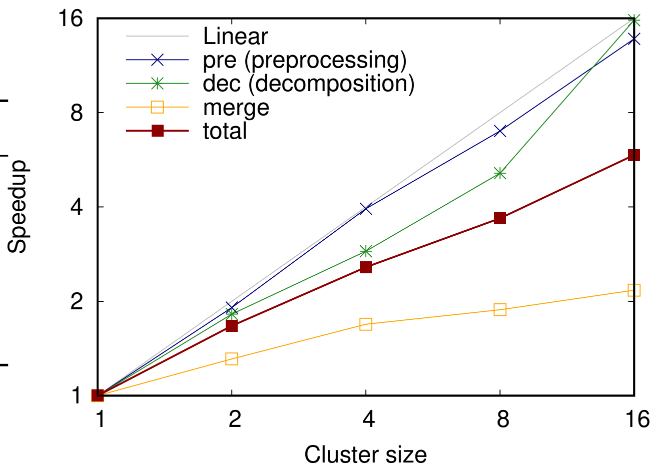
For the music domain, we evaluate the cluster quality for DS-M1 using a set of computed input links (see setup in Section 8.5.3). Overall, the quality of the input links is lower than for DS-G1. Due to strongly corrupted entities and more properties, DS-M1 is more difficult to handle. Applying the SplitMerge clustering, we identify a quality improvement for both precision and recall, resulting in a significant increase of F-measure by approx. 7 % to 87.6 % (see in Table 8.4), showing that our clustering approach can handle such unclean data.

Overall, the SplitMerge approach achieves competitive results, although the DS-G1 dataset facilitates achieving relatively good input mappings, making it difficult for any clustering approach to find additional or incorrect links.

Finally, *efficiency and scalability* of distributed SplitMerge clustering are evaluated. For this purpose, the execution times and the speedup for the very large geographic (DS-G2) and music datasets (DS-M2, DS-M3) are determined. In Flink, several options allow the fine-tuning of parameters for a distributed cluster environment. In most cases, a reasonable way to improve execution times for very large datasets is the increase of deployed workers. Another important parameter is the parallelism to specify the maximum number of parallel instances of operators or data sinks/sources that are available to process data within a Flink workflow. We here used the parallelism equal to the number of workers, for which we achieved the best execution times.

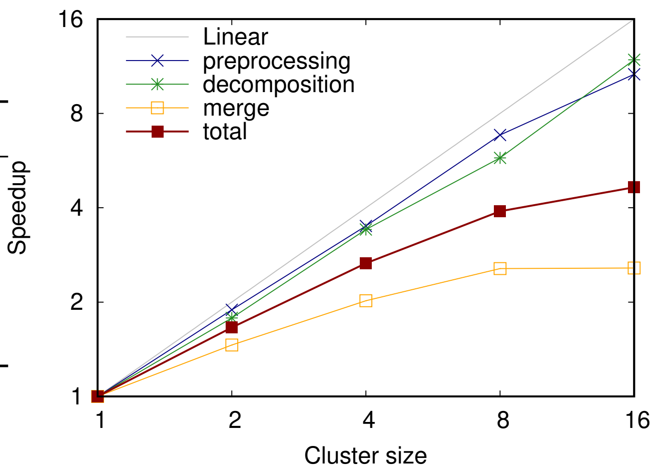
Figure 8.6 shows the achieved execution times for DS-G2 and DS-M2, respectively, for different phases of the clustering workflow as well as the overall workflow execution time. For each phase, an increased number of workers leads to improved execution times. For both domains, the best improvement can be achieved for the preprocessing (pre) and decomposition (dec) phases. The merge phase is far more complex. While preprocessing and decomposition operate within connected components and clusters, the merge phase attempts to combine similar clusters based on the assignment in the blocking step and therefore can suffer from data skew problems for some blocks. These effects also become clear in Figure 8.6, showing the speedup results compared to the

#workers	pre	dec	merge	total
1	312	668	351	1,331
2	164	367	268	799
4	79	231	207	518
8	45	130	186	361
16	23	42	162	227



(a) Execution times (left) in seconds and speedup (right) for geographic dataset DS-G2.

#workers	pre	dec	merge	total
1	423	419	608	1,450
2	224	236	417	876
4	121	123	301	545
8	62	73	238	372
16	40	35	237	312



(b) Execution times (left) in seconds and speedup (right) for music dataset DS-M2.

Figure 8.6: SplitMerge execution times and speedup for single workflow phases and total workflow.

linear optimum. Preprocessing and decomposition achieve nearly linear speedup, while the merge phase shows decreased speedup values. In total, we achieve a good speedup of 5.86 for the large geographic dataset DS-G2 (Figure 8.6a) and 4.65 for the large music dataset DS-M2 (Figure 8.6b). For the largest dataset DS-M3 with ≈ 20 million entities, we could determine results for two configurations: 8 workers could finish the complex task in 43,589 seconds, and 16 workers finished after 24,722 seconds (reduced by factor ≈1.8).

Altogether, the distributed SplitMerge clustering achieves good execution times and moderate scalability results for very large entity sets. The approach is scalable for different data sources and employs a multi-source clustering instead of a basic binary linking of two sources. The distributed implementation further allows to scale for a growing number of entities and data sources and is very useful for complex data integration scenarios in Big Data processing workflows.

Distributed Incremental Clustering

This part of the evaluation complements the results for incremental clustering in Chapter 6. We analyze the source-specific incremental clustering for the larger music and person datasets (DS-M2, DS-P2) for different blocking approaches and minimum similarity thresholds t_{min}. The obtained precision, recall and F-measure results are shown in Figure 8.7, where the values for t_{min} (x-axes) range between 0.6 and 0.9 and the prefix lengths for the blocking keys lie between 1 and 5 for DS-M2 and between 2 and 8 for DS-P2. As expected, precision increases and recall decreases with growing t_{min} while F-Measure is relatively stable across different threshold values with the best result for $t_{\text{min}} = 0.7$. Despite the large data volume (0.4 to 1 million entities per source) and the integration of 5-10 sources, F-measure reaches relatively good values of up to 87.41% for DS-M2 and up to 80.47% for DS-P2.

The choice of blocking key substantially impacts cluster quality and runtime for both datasets. For DS-M2, precision is not affected by the blocking key so that it remains good even for smaller prefixes (larger blocks) probably favored by the high precision of Max-Both. As a result, the best F-measure is achieved for the smallest prefix length $B = 1$ (largest blocks) supporting the highest recall. The downside of this selection, however, is the large execution time, which is hugely dependent on the chosen blocking key. As shown in Table 8.5, the runtime for $B = 1$ is more than 5 times as high than for $B = 2$ and more than a factor 40 slower than for $B = 4$. For the larger dataset DS-P2 with 10 sources, precision suffers from larger block sizes (small prefix lengths), while recall is worst for longer prefix values. So we obtain the best precision but lowest recall for prefix length 8, while prefix length 6 allows for the best compromise value and thus the best F-Measure. For prefix length 6, we analyze the effect of the block size distribution. We therefore partition the blocks in three block size ranges: blocks with 1-100, 101-1000 and more than 1000 entities per block. Only $0.03\,\%$ of all blocks contain > 1000 entities (biggest block contains 6617 entities), while these blocks contain $4.6\,\%$ of all entities, they need most of the time for the actual comparison. $1.7\,\%$ of all blocks are within the range 101-1000 but contain $35.3\,\%$ of the entities. Finally, $98.3\,\%$ of the blocks have 1-100 entities ($60.2\,\%$ of all entities belong to these blocks). For bigger prefix length, the distribution is likely to have more and smaller blocks reducing computational effort but also reducing the possibility to compare potentially similar entities with each other. Again, the runtime differences (Table 8.5) are substantial where blocking with

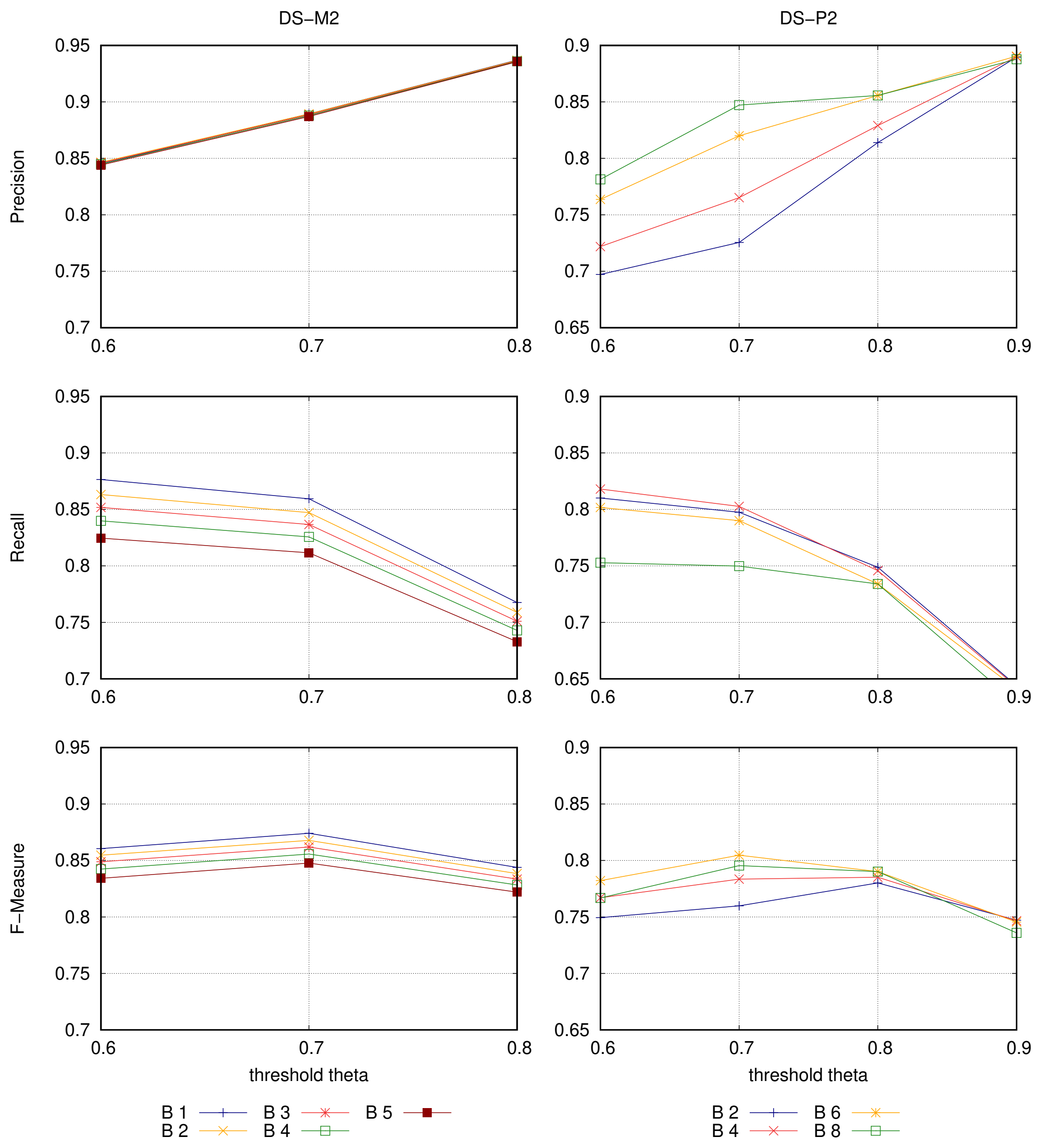


Figure 8.7: Precision, recall and F-measure for incremental clustering on music dataset DS-M2 (left) and person dataset DS-P2 (right).

prefix length 6 achieves a fast clustering of only 706 s (12 min) for $t_{\min} = 0.7$, which is two orders of magnitude faster than with prefix length 2.

This is especially remarkable since the runtime typically increases strongly with the number of sources; each new source increases the number of clusters and thus the match overhead heavily. We illustrate this in Table 8.6 for dataset DS-M2 showing how the total runtime is distributed over the additions of the different sources. We observe that the time to add a source increases continuously so that adding the 5th source is more than

Table 8.5: Incremental clustering runtime (s) for different blocking key (bk) lengths and t_{min} values (16 workers).

	DS-M2				DS-P2			
bk length	4	3	2	1	8	6	4	2
t_{min} 0.6	1071	2044	8292	46346	447	690	3621	67681
t_{min} 0.7	1093	2077	8062	46678	431	706	3338	72301
t_{min} 0.8	1139	2210	8875	49154	442	754	3203	96855

Table 8.6: Runtimes (s) for single steps with source-specific incremental clustering for DS-M2 with different blocking key (bk) lengths and $t_{\text{min}} = 0.7$.

	runtime (s)				
bk length	5	4	3	2	1
source 1 + source 2	129	170	315	1183	6456
add source 3	173	235	440	1692	9862
add source 4	218	311	600	2319	13481
add source 5	272	377	722	2868	16879
total	792	1093	2077	8062	46678

twice as slow than adding the second source. This is because 25% of the entities in each source have no duplicates for DS-M2[8], thereby resulting in the creation of additional clusters that have to be considered in the subsequent cluster decisions.

We finally analyze the *speedup* behavior of source-specific incremental clustering w.r.t. clusters of different sizes. For this experiment, we focus on the two largest person datasets DS-P1 (5 sources) and DS-P2 (10 sources). In both cases, we consider two blocking configurations with prefix lengths 4 and 6. The resulting runtime and speedup values for using 1 to 16 workers are shown in Figure 8.8. The results show that we can always improve runtime by using more workers where the speedup is best for the most expensive configurations with bigger blocks (prefix length 4). For DS-P2 and we achieve near-linear speedups of 5.7 for 8 workers and 9.6 for 16 workers in this case. For prefix length 6, the runtime to incrementally cluster 5 and 10 million entities is less than 4 and 12 minutes, respectively, for 16 workers and thus remarkably fast.

[8] 50% of the clusters in DS-M2 (500K) are singletons while the other half of the clusters has between 2 and 5 duplicate entities.

	runtime (s)			
cluster size	P1 b4	P1 b6	P2 b4	P2 b6
1	9,751	860	32,093	3,648
2	5,468	732	17,583	2,275
4	2,997	430	9,346	1,494
8	1,785	320	5,677	1,023
16	1,100	217	3,338	706

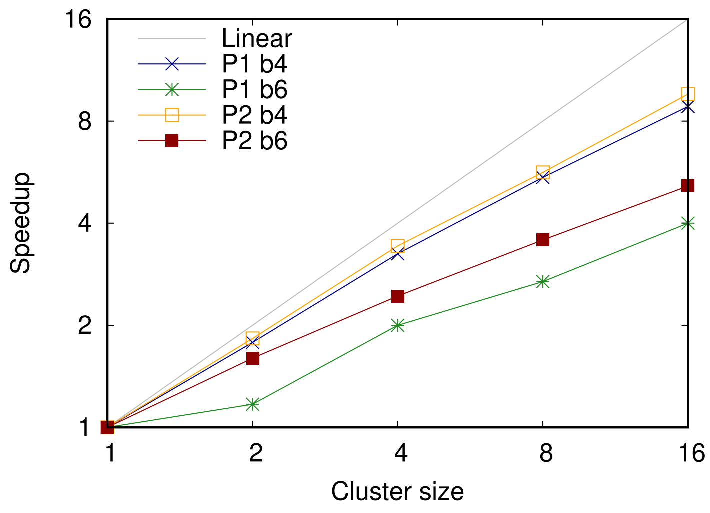


Figure 8.8: Incremental clustering execution times (top) in seconds and speedup (bottom) for datasets DS-P1 (P1) and DS-P2 (P2) for blocking key lengths 4 and 6.

8.6 Related Work

Typical link discovery approaches apply binary linking methods for matching two data sources but lack efficient and effective methods for integrating entities from k different data sources to provide a holistic view for Linked Data [132]. Some approaches enable distributed link discovery or for matching two data sources. For instance, Silk MapReduce [89] and LIMES [83] realized link discovery based on MapReduce before distributed data processing frameworks like Spark or Flink became state of the art. Similarly, the entity resolution framework Dedoop [105] allows executing complex matching workflows on MapReduce. These tools suffer from limitations of MapReduce, like repeated data materialization within and between single jobs and the lack of iterations. They further focus on pairwise matching and do not support holistic clustering for multiple data sources or the reuse of existing link sets.

While link discovery is driven by pairwise linking of data sources, support for multiple data sources can be found in related research areas. In [72], ontology concepts from multiple data sources are clustered based on topic forests for extracted keywords from concepts and their descriptions to determine matching concepts within groups of similar topics. In [122], a maximum-weighted graph matching and structural similarity computations are applied to concepts of multiple ontologies to find high-quality alignments. However, these holistic ontology matching approaches do not focus on clustering of concepts or entities and have limitations w.r.t. scalability for link discovery and many large entity data sources.

Few link discovery approaches apply clustering or linking for Linked Data on multiple sources. Thalhammer et al. [177] present a pipeline for web data fusion using multiple data sources applying hierarchical clustering, cluster refinement and selection of representatives to achieve a similarity-based clustering with unified entities. The unsupervised link discovery approach Colibri [145] considers error detection and correction for link discovery in multiple knowledge bases relying on the transitivity of interlinked entities, while clustering of entities is not the main focus. These approaches do not realize distributed clustering or linking and have not been evaluated regarding scalability. There has been some research for distributed clustering methods such as community detection within social networks, e. g., using MapReduce exclusively [125] or MapReduce and Spark [73]. However, these approaches do not focus on complex link discovery and data integration workflows and partially suffer from MapReduce limitations. The work in [163] considers the implementation of existing clustering algorithms on top of Apache Flink for entity resolution and de-duplication of several data sources. The approach does not handle incorrect links and lacks the use of semantic type information. It further does not create representatives for a compact cluster representation as useful for further applications.

In our previous work [131], we proposed a holistic clustering approach for multiple Linked Data sources with few initial evaluation results. Here we present an extended workflow implementation, in particular, the realization of holistic clustering in a distributed environment. Besides, we show comprehensive evaluation results w.r.t. cluster quality based on a new gold standard as well as scalability for datasets covering very large entity sets from multiple different data sources and domains. The created compact cluster representation is particularly useful for reuse and incremental cluster extension.

8.7 Conclusion

In this chapter, we presented distributed holistic clustering workflows for Linked Data using the distributed data processing framework Apache Flink. The first approach is based on the reuse of existing links and can handle entities from various data sources. Besides this approach for static entity clustering, continuously changing data sources are supported in a second approach for incremental clustering. We showed the realization of both holistic clustering workflows with dataset transformations and user-defined functions. Albeit based on Apache Flink, the methods could also be implemented on other frameworks for distributed data processing such as Apache Spark. We further provide a novel gold standard for multi-source clustering from the geographic domain to support the development and evaluation of novel holistic clustering methods. We showed comprehensive evaluation results for datasets from three different domains, with up to 20 million entities. Our results showed that the proposed approaches achieve a very high cluster quality. In particular, we were able to find many new correct links and could remove incorrect links.

In terms of runtime efficiency, both approaches reached very good execution times for most of the considered dataset sizes. As shown for incremental clustering, the blocking strategy and the associated parameters strongly influence the runtime and therefore need to be adjusted accordingly. Both the distributed SplitMerge and the incremental clustering showed good scalability on the Apache Flink cluster with up to 16 machines.

Overall, the presented results show that distributed entity clustering is an effective solution to integrate a growing amount of data sources in the Web of Data. So far, there is no comprehensive comparison of related clustering methods. Consequently, the following chapter discusses a comparative evaluation of distributed clustering approaches.

9 Comparative Evaluation of Clustering Methods

Preamble

The subsequent chapter is based on [162]. We compare multiple holistic clustering approaches with an implementation on Apache Flink with each other. The coverage of very general as well as specialized clustering approaches provides a comprehensive overview. A standardized workflow is adopted to ensure the comparability of the evaluation with different datasets. The journal article was published in CSIMQ in 2017.

9.1 Motivation

With more frequent use of distributed data processing systems in general, the application for data integration tasks also increases. Link discovery (also data matching or entity resolution) is a sub-task for data integration in which approaches link entities from different heterogeneous data sources with each other. The link discovery process must be realized in compliance with relevant requirements (see Section 3.2 for details). Besides effective algorithms, the requirements include efficient execution, e. g., by utilization of distributed processing systems such as Hadoop services.

Therefore, modern distributed processing systems such as Apache Spark or Apache Flink provide an abstraction layer for the transparent use of shared-nothing computer clusters. The systems distribute the program code and (if necessary) the required data in order to use the available resources as fully as possible. With the application of these distributed processing systems, the user benefits directly from improvements such as in-memory processing. Still, varying specializations of the systems ensure that performance differences can be detected for distinct classes of workload [64, 98, 116, 183].

Growing data size also makes scalable approaches useful for the integration of Linked Data. The generalized workflow from Section 3.3 includes all components for the link discovery process and can be used for more than two data sources. To increase the efficiency of the procedures, it is necessary to apply measures to avoid the naive comparison of all entities with each other. For this reduction of the search space, blocking or filtering strategies are used to assign the entities (e. g., title starting with the same letters) into blocks. By distributing disjunctive blocks of entities to the existing computer infrastructure, a local problem solution in each block can lead to a significant reduction in the amount of data that has to be exchanged between the computer nodes. However, load balancing problems can occur if the frequency distribution of the selected attributes for reducing the search space is skewed too much. Various optimizations to avoid load balancing problems during search space reduction have been investigated, especially for MapReduce [106, 107].

The subsequent instance matching challenges approaches to deal with multiple data sources in order to produce high-quality results at the lowest possible runtime. One possibility to handle equivalent entities from different sources is the integration of unified clusters, as proposed in Chapter 5. In addition to a compact representation, clusters can also be used to enable continuous data integration with new data sources or entities, as shown in Chapter 6. To cope with the larger data sources, these clustering approaches have been implemented and evaluated on Apache Flink (see Chapter 8). Other systems for multi-source entity clustering on distributed processing systems use similar techniques to achieve high-quality results and to maintain consistency [164].

While various clustering approaches employ distributed processing systems, there are no comprehensive and uniform comparisons of quality and runtime on large datasets. As a scientific question, it is therefore interesting to see how efficiently and effectively different clustering methods perform when using the same datasets. Based on this question, this chapter contains the following contributions:

- A description of the investigated methods for multi-source entity clustering, which are available for the distributed data processing system Apache Flink.
- We perform a comprehensive evaluation of qualitative results and runtime on different datasets to show the scalability of the investigated methods.

9.2 Problem Statement

We start with a set of data sources k, which are to be connected employing a data integration workflow. With the use of the distributed processing system Apache Flink, it is possible to increase overall scalability and to use special operators for iterative or graph-based computing. Since previous clustering approaches do not always provide a complete processing pipeline, we utilize the multi-source clustering system FAMER [162] to enable the execution of linking and various clustering algorithms. The general workflow is shown in Figure 9.1 and consists of two main steps.

The first part includes techniques such as blocking and pairwise comparisons of entities to create a similarity graph. According to the data model in Section 2.2, the similarity graph corresponds to the set of entities $\mathcal{E}$ as graph vertices and the set of determined links $\mathcal{L}$ between the different sources as graph edges. An efficient process for a similarity calculation can then be migrated towards a dedicated similarity graph generation. The computed results are made available to arbitrary clustering methods and allow a comparable initial situation. This also allows the specification of the same configuration parameters and threshold values in order to create a consistent similarity graph.

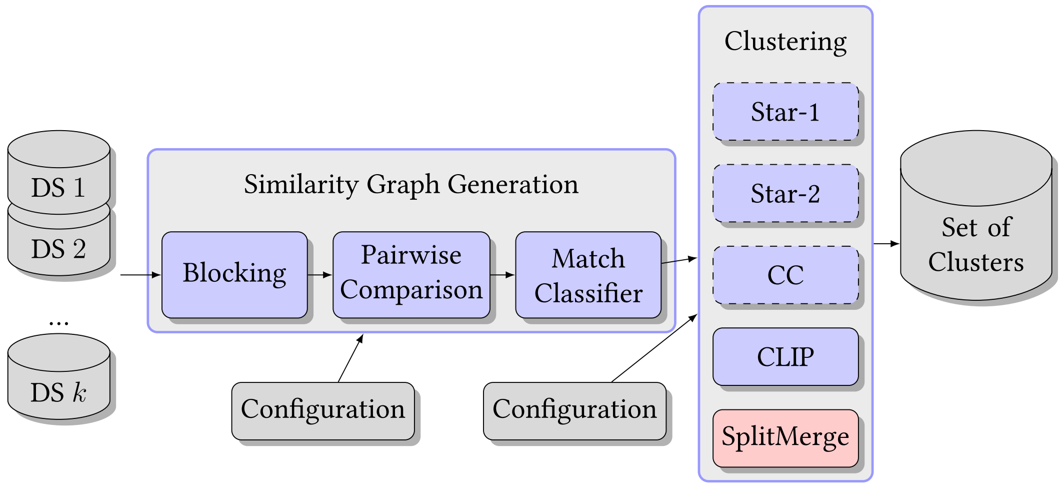


Figure 9.1: Workflow overview for multi-source clustering in the FAMER system. Our SplitMerge approach has been added to the set of possible clustering methods.

Based on this similarity graph, the entities are then further processed using a clustering approach. Each resulting cluster reflects a set of equivalent real-world objects. Compared to the result set for link discovery (see Section 3.3), a cluster with m entities represents $(m \cdot (m-1))/2$ match pairs. In order to focus on the clustering step in the workflow, we assume comparable entities in the data sources. This is expressed above all in the equivalent semantic type (e. g., only locations, persons or songs) but also with comparable attributes. As in previous chapters, data sources do not contain any duplicates. Accordingly, a source consistency is again assumed for resulting clusters, i. e., for each source at most one entity and thus not more than k entities in the cluster. These steps aim to avoid data quality problems by applying data cleaning methods [158] to create a consistent data representation.

In the following two sections, the similarity graph generation and clustering are discussed in detail to show the process of multi-source clustering. For this purpose, exemplary data entries about persons are used, which can be seen in table Table 9.1. Multiple property values are used to describe entities of four different data sources. Erroneous data values make the correct assignment to a cluster more difficult. The partitioning of the table (horizontal lines) already shows matching entities that correspond to the same real-world objects. Therefore, Section 9.3 describes how to create a similarity graph using the steps blocking, pairwise comparison and match classifier. We present a selection of four appropriate clustering methods in Section 9.4. In addition to a simple method, which creates connected components based on the available links, three methods using more complex metrics for the discovery of clusters are described.

Table 9.1: Sample person entities to show problems for clustering approaches. The partitioning using horizontal separation indicates different real-world objects.

ID	Name	Surname	Suburb	Post code	SourceID
a_0	Bertha	Watkins	Wilmington	28282	SrcA
b_0	Bertha	Watkins	Wilmingtn	2822	SrcB
c_0	Brtha	Watkins	Wilmington	28222	SrcC
d_0	Bertha	Watkens	Winington	28223	SrcD
a_1	Bernie	Watkins	Wilmsor	28572	SrcA
b_1	Bernie	Watkns	Winstom	2787z	SrcB
c_1	Bernii	Wakens	Windsor	28571	SrcC
a_2	ge0rge	Walker	Winston salem	271o6	SrcA
b_2	Gerge	Waker	Winston salem	27106	SrcB
c_2	George	Alker	Winstons	27106	SrcC
d_2	Geoahge	Alker	Winston	271oo	SrcD
a_3	Gerald	Waker	Winston Salem	27707	SrcA
b_3	Gera1d	Walker	Winston Salem	27707	SrcB
d_4	Larry	Walker	salem	28090	SrcD

Table 9.2: Blocking keys.

ID	Key
a_0	wa
a_1	wa
a_2	wa
a_3	wa
b_0	wa
b_1	wa
b_2	wa
b_3	wa
c_0	wa
c_1	wa
d_0	wa
d_4	wa
c_2	al
d_2	al

9.3 Similarity Graph Generation

To create the similarity graph, a blocking step is first performed to reduce the number of comparisons required between entities from different sources. Blocking can be done by partitioning the entities according to distinctive features of the data, e. g., only compare persons starting with the same initial letter. The selection of the blocking technique can have a decisive influence on the runtime [31, 152].

We will show an example for generating blocking keys with standard blocking based on the entities from Table 9.1. The first two letters of the surname as blocking key causes an unequal partitioning of almost all entities into the block “wa”, as shown in Table 9.2. Still, the number of comparisons is decreased because entities c_2 and d_2 have been assigned to an extra block.

On distributed processing systems, the partitioning of distinct blocks ensures that parts of the computation can be executed locally. In the context of the blocking step, the unequal distribution of block sizes can result in individual cluster workers consuming excessive time. For this reason, the handling of load balancing problems is an important issue on systems such as Apache Flink. In our case, we therefore use the block split method [106] to avoid load balancing problems. In detail, too large blocks such as “wa” are split among the number of available workers in the cluster. In any search space reduction procedure, data quality problems or inadequate method selection can result in a reduced recall by not finding relevant links. Example entities with poor data quality are c_2 and d_2; they are assigned to an extra block because of the misspelled surname. The effects of incorrect classification can be mitigated by repeatedly blocking based on different blocking attributes or by post-processing the results.

The blocking step is followed by the pairwise comparison of all entities within each block. A similarity function specifies which attribute values are compared for two entities. This similarity function is provided to the process in the configuration step and can be determined manually or trained by learning-based approaches. Typically, different string similarity metrics and domain-specific metrics (e. g., to support geographic coordinates or dates) are used. Based on these similarities, a classifier then determines the matching links to be included in the similarity graph. For this decision, match rules from the configuration are utilized, e. g., a minimum average threshold or a weighted average in favor of certain properties.

The resulting similarity graph $\mathcal{SG} = (\mathcal{E}, \mathcal{L})$ contains all entities $\mathcal{E}$ (vertices) and all matching links $\mathcal{L}$ (edges) and can be saved for later reuse, e. g., as a basis for clustering algorithms. The next section presents all relevant information about the clustering step, including details on the investigated algorithms.

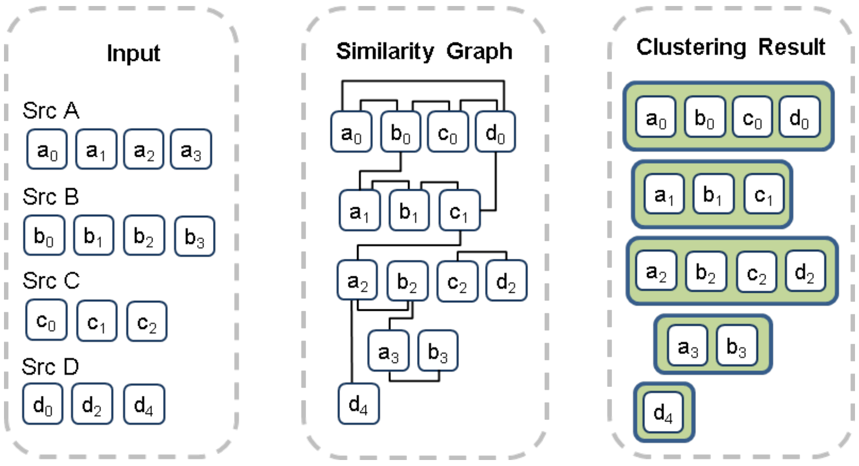


Figure 9.2: Clustering phases showing generated similarity graph and perfect clustering result for example entities.

9.4 Clustering approaches

The clustering step identifies matching entities from previously determined similarity values to provide a compact, bundled representation. Therefore, we employ the created similarity graph with entities $\mathcal{E}$ and links $\mathcal{L}$ as the input. With the similarity values stored in the links, the application of more or less complex procedures for grouping entities determines clusters. With the baseline algorithm (weakly) connected components, all entities that can be reached via arbitrary links are merged into a cluster. The Star approach determines clusters from entities with the highest number of links. Both approaches do not consider the constraints for creating clusters (see Section 9.2) and can therefore contain source-inconsistencies. In contrast, the two approaches CLIP and SplitMerge only generate source-consistent clusters, i. e., a cluster contains at most one entity per data source. CLIP uses metadata from the similarity graph such as link strength and link degree to classify clusters correctly. SplitMerge refines the initial clustering result in several sub-steps (such as split and merge) and can therefore correct wrong blocking assignments as for entities c_2 and d_2. In the remaining section, the four different concepts are presented accordingly in detail.

The approaches typically form clusters based on high similarities between entities and thus build new clusters when only low similarities exist. Thus, in the set of links from the similarity graph, intra-cluster similarities are maximized while inter-cluster similarities are minimized, thereby forming the set of resulting clusters. With an efficient clustering approach, these clusters contain more correct links (all links within the cluster) and no false links (between different clusters) compared to the initial similarity graph.

We implemented all clustering approaches on Apache Flink. The usage of the graph-processing engine Gelly provides many advantages, e. g., the graph data model, specific graph-processing operators or programming abstractions for iterative graph processing. Especially the different iterative graph processing abstractions are a strength of Apache Flink and are utilized accordingly. Besides the Gelly implementation for weakly connected components (gather-sum-apply [67] or scatter-gather [176]) and vertex-centric iteration [119], the delta iteration of Apache Flink are also used.

9.4.1 Connected components

Weakly connected components are used as baseline algorithm to determine clusters. This means that all entities connected by links from the similarity graph span a connected component. In the exemplary similarity graph, two components of very different sizes are created. Entities c_2 and d_2 form the first component and all other entities span up the second component. Due to potentially large components, which are accepted as a result without further checking, many additional links will likely be found. Compared to the similarity graph, it can be expected that a higher number of relevant links will be found, which can have a positive effect on recall. Similarly, the number of false-positive links can lead to decreased precision.

In Apache Flink, the computation of weakly connected components can be carried out using the programming abstraction scatter-gather. In the scatter phase, the iterative method propagates the unique identifiers of the entities to all neighboring entities. In the gather phase, each entity then selects the minimum of the received identifiers and uses them as a starting point for the next iteration. The algorithm terminates when an iteration no longer updates entity identifiers.

9.4.2 Star clustering

The Star clustering algorithm [6] originates from the domain of information retrieval and tries to form clusters based on the strength and proximity of neighborhood relationships in a graph. These relationships can be determined by similarity calculations between the property values of entities to create a similarity graph. With the limitation to similarity values above a threshold, only significant edges remain. Two variants of the algorithm are presented. The clustering algorithm Star-1 determines iteratively (central) entities with the highest degree and forms a cluster with the adjacent entities. In our example graph in Figure 9.2, entities b_0, d_0, c_1 and a_2 are selected in the first iteration having three edges. Clusters are then created by adding all adjacent entities for each of these entities. This process may lead to overlapping clusters, e. g., c_0 is added to both b_0 and d_0. Alternatively, for Star-2, the next central entity is determined by the highest average similarity across all edges. Accordingly, the neighboring entities create the cluster together with the central entity.

The Apache Flink implementation is realized with the scatter-gather abstraction and is described in [164]. In a three-step procedure, first, the edge degree is calculated, then the central entities are selected and finally, the clusters are spanned around the central elements.

9.4.3 SplitMerge clustering

The SplitMerge approach proposed in Chapter 5 is more general than the other clustering schemes as it can deal with entities of different semantic types as well as dirty input sources and links, e. g., with duplicates in sources. For this study, we will refine the existing SplitMerge algorithm in this section to achieve optimal results according to the task at hand.

In contrast to Chapter 5, we omit the preprocessing since only entities of a single semantic type and duplicate-free sources are considered. Further improvements include the application of the already calculated similarity graph. In addition, initial clustering is optimized and the merge phase is made more efficient through the use of blocking. Parameters such as similarity thresholds for split/merge and a blocking function for the merge phase are specified in the configuration.

To show these optimizations for a single semantic type, we apply the refined SplitMerge to the similarity graph, as shown in Figure 9.3. The pseudo-code from Algorithm 5 shows the three phases of the SplitMerge approach: (1) determining initial clusters by applying connected components and making the components source-consistent, (2) splitting clusters to ensure a high intra-cluster similarity and (3) merging similar clusters. More details on the Flink implementation of SplitMerge are described in Chapter 8.

SplitMerge determines connected components (line 2 of Algorithm 5) on the input similarity graph to create initial components $\mathcal{C}_{\text{init}}$. The resulting components may often violate the required source consistency since entities from the same source may be indirectly linked and thus become members of the same connected component. In our example in Figure 9.3, there are only two connected components where the smaller one (with entities c_2 and d_2) is source-consistent, while the larger one contains up to four entities per source. To achieve source-consistent clusters, we decompose the inconsistent components by removing violating links. The links between (a_0, b_0) and (b_0, a_1) result in a source inconsistency for source A and we solve this by removing one of the two

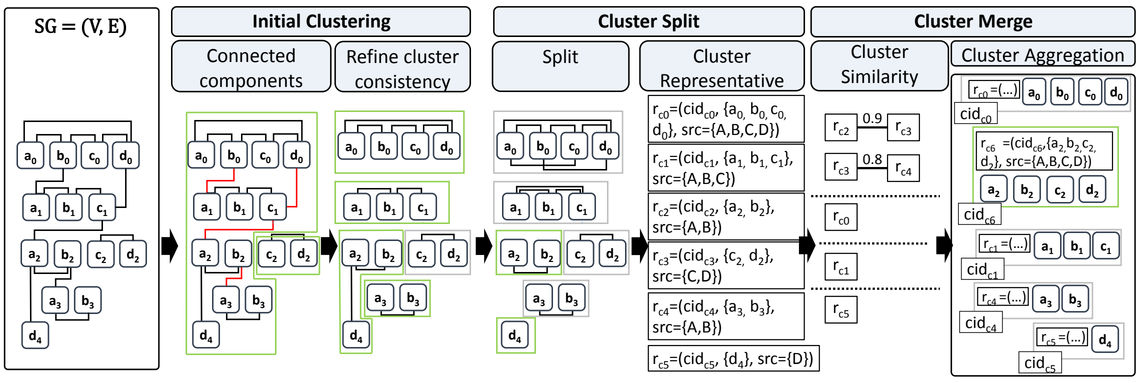

SG = (V, E)
Initial Clustering
Connected components
Refine cluster consistency
Cluster Split
Split
Cluster Representative
Cluster Merge
Cluster Similarity
Cluster Aggregation
r_c0=(cid_c0, {a_0, b_0, c_0, d_0}, src={A,B,C,D})
r_c1=(cid_c1, {a_1, b_1, c_1}, src={A,B,C})
r_c2=(cid_c2, {a_2, b_2}, src={A,B})
r_c3=(cid_c3, {c_2, d_2}, src={C,D})
r_c4=(cid_c4, {a_3, b_3}, src={A,B})
r_c5=(cid_c5, {d_4}, src={D})
0.9
0.8
r_c6 =(cid_c6,{a_2,b_2,c_2, d_2}, src={A,B,C,D})
cid_c0
cid_c6
cid_c1
cid_c4
cid_c5

Figure 9.3: Running example processed with SplitMerge clustering.

links (the one with lower similarity). Another example with three links resulting in a source inconsistency is (b_1, c_1, a_2, b_2); again, we eliminate at least one link, e. g., (c_1, a_2), to solve the problem.

To identify the links to be removed, we record for every entity e the set of already associated data sources in an element $\texttt{assocSrc}(e)$, which initially contains the source of e (line 4). We iterate over all links of a component in descending order of their similarity. For each considered link (e_s, e_t), we check whether it results in a source inconsistency, which is the case if there is a non-empty overlap between $\texttt{assocSrc}(e_s)$ and $\texttt{assocSrc}(e_t)$. If there is such a conflict, the link will be eliminated (line 8). Otherwise, we update both sets of associated sources to the union of $\texttt{assocSrc}(e_s)$ and $\texttt{assocSrc}(e_t)$ (line 10). In Figure 9.3, the conflicting links are highlighted (red color) to indicate the removal. For example, if we first process link (a_0, b_0), we will add sources A and B to $\texttt{assocSrc}(a_0)$ and $\texttt{assocSrc}(b_0)$. The link (b_0, a_1) will then lead to a conflict for b_0, which is already associated with source A. Thus, the link (b_0, a_1) is eliminated. After the processing of all links, we determine the connected components with the remaining links to compute the source-consistent subcomponents (line 11). In our running example, we obtain the four smaller clusters shown (with green borders) in the third graph from the left in Figure 9.3.

Split

For the split phase, we process the clusters from the first phase in parallel. For each cluster, we first calculate link similarities for each pair of entities based on the similarity function f_{sim} provided in the input. This is needed to identify entities with an insufficient similarity to other cluster members. To detect possible splits (line 15) we determine for each entity the average similarity of its links to other cluster members and separate an entity if the average similarity is below the split threshold t_s. After the elimination of such entities, we iteratively repeat this split processing based on recomputed entity similarities until the similarity threshold t_s is met for all links. This leads to the elimination of d_4 from cluster a_2, b_2, d_4 (fourth graph from the left in Figure 9.3). For each resulting cluster, we next determine a cluster representative (line 16) from the properties of the cluster members, e. g., based on the values of preferred sources or a majority consensus of values. As indicated in Figure 9.3, each cluster representative has a unique ID and keeps track of the covered cluster entities and their sources as provenance information. The representatives are used for a simplified computation of cluster similarities as needed for the final merge phase.

Algorithm 5: Refined SplitMerge Clustering

```
Input: $\mathcal{SG} = (\mathcal{V}, \mathcal{E})$, simFunction $f_{\text{sim}}$, blocking function $f_{\text{blocking}}$, thresholds $t_s, t_m$
Output: Cluster set $\mathcal{CS}$
1  $\mathcal{CS} \leftarrow \emptyset$
                                                  /* initial clustering */
2  $\mathcal{C}_{\text{init}} \leftarrow$computeConnectedComponents($\mathcal{V}, \mathcal{E}$)
3  for $\mathcal{C}_i(\mathcal{V}_i, \mathcal{E}_i) \in \mathcal{C}_{\text{init}}$ in Parallel do
4      $\mathcal{V}_{\text{i}} \leftarrow$ initAssocSrc($\mathcal{V}_i$)
5      $\mathcal{E}_{\text{sorted}} \leftarrow$ sortLinkSims($\mathcal{E}_i$)
6      foreach $(e_s, e_t) \in \mathcal{E}_{sorted}$ do
7          if (assocSrc($e_s$) $\cap$ assocSrc($e_t$) $\neq \emptyset$) then
8              $\mathcal{E}_{\text{i}} \leftarrow$ removeLink(($e_s, e_t$))
9          else
10             $\mathcal{V}_{\text{i}} \leftarrow$ updateAssocSrc(assocSrc($e_s$) $\cup$ assocSrc($e_t$))
11     $\mathcal{C}'_{\text{i}} \leftarrow$ computeConnectedComponents($\mathcal{V}_i, \mathcal{E}_i$)
12     $\mathcal{CS} \leftarrow \mathcal{CS} \cup \mathcal{C}'_{\text{i}}$
13 for $\mathcal{C}_i \in \mathcal{CS}$ in Parallel do
14     $\mathcal{C}_{\text{i}} \leftarrow$ computeLinkSim($\mathcal{C}_i, f_{\text{sim}}$)
15     $\mathcal{C}_{\text{split}} \leftarrow$ clusterSplit($\mathcal{C}_i, t_s$)          /* cluster split */
16     $\mathcal{C}_{\text{split}} \leftarrow$ createRepresentatives($\mathcal{C}_{\text{split}}$)
17     $\mathcal{CS} \leftarrow \mathcal{CS} \cup \mathcal{C}_{\text{split}}$
                                    /* create cluster mapping $\mathcal{CM}$ */
18 $\mathcal{CM} \leftarrow$ computeClusterSim($\mathcal{CS}, f_{\text{sim}}, t_m, bf$)
19 while $\mathcal{CM} \neq \emptyset$ do
20     $(c_1, c_2) \leftarrow$ getBestMatch($\mathcal{CM}$)
21     $c_m \leftarrow merge(c_1, c_2)$                        /* cluster merge */
22     $\mathcal{CS} \leftarrow \mathcal{CS} \setminus \{c_1, c_2\} \cup \{c_m\}$
23     $\mathcal{CM} \leftarrow$ adaptMapping($\mathcal{CM}, \mathcal{CS}, c_m, c_1, c_2, f_{\text{sim}}, t_m$)
24 return $\mathcal{CS}$
```

Merge

The goal of the merge phase is to identify highly similar pairs of clusters that likely represent the same real-world entity and should thus be combined. The merge phase can also help to assign entities separated during the split phase to a more similar cluster. The first step is to determine a so-called cluster mapping $\mathcal{CM}$ (line 18 of Algorithm 5) consisting of all cluster pairs with a similarity above the merge threshold t_m (merge candidates). The similarity between clusters is computed by applying function f_{sim} on the cluster representatives. Since the computation of these similarities is an expensive process for many clusters, we reduce the number of comparisons by applying a blocking function f_{blocking} specified as an input parameter (in the current implementation, we apply standard blocking on selected properties of the cluster representatives). Furthermore, we only compare clusters with entities from different sources since otherwise, merging them would violate source consistency. In Figure 9.3, we have three clusters in the first block and only one in the remaining three blocks. For the first block, we obtain two merge candidates with a sufficiently high cluster similarity.

Cluster merging is an iterative process (lines 19 to 23) that continues as long as there are merge candidates in the determined cluster mapping $\mathcal{CM}$. In each iteration, we select the pair of clusters (c_1, c_2) with the highest similarity from $\mathcal{CM}$ (line 20) and merge it into a new cluster c_m (line 21). This merging also includes the computation of a new representative for c_m. The "old" clusters c_1 and c_2 are removed from the cluster set and the new cluster c_m is added. We further need to adapt $\mathcal{CM}$ by removing all cluster pairs involving either c_1 or c_2 (line 22). Furthermore, we have to extend $\mathcal{CM}$ by similar cluster pairs (c_i, c_m) for the new cluster c_m with a cluster similarity of at least t_m and entities from different sources (line 23). For our running example, we first process the merge candidate with similarity 0.9 and obtain the merged cluster $\{a_2, b_2, c_2, d_2\}$. The second merge candidate will be removed and it is checked whether the new cluster results in new merge candidates. Since the new cluster contains already entities from every source, merging any other cluster would result in a source inconsistency so that no new merge candidates result in the example. The outcome of SplitMerge contains five clusters that correspond to the perfect result in Figure 9.2.

9.4.4 CLIP

The clustering algorithm CLIP (Clustering based on LInk Priority) [164] makes use of link characteristics given with the similarity graph. Each entity determines the maximum similarity of all its links – links with this characteristic are called maximum links. The individual link strength is determined with the help of the maximum links and distinguishes between strong, normal and weak. *Strong* links are a maximum link for entities on both sides. Because there may be multiple maximum links for an entity, each of these can be strong if it is also a maximum link for the opposite entity. A *normal* link implies that only one entity has determined it as a maximum. For *weak* links, neither of the two entities has flagged it as maximum. Furthermore, we use the term *link degree* as the minimum number of links of either of the associated vertices. In the similarity graph of Figure 9.2, we show an example of the link degree for (a_2, d_4). Given that the link degree of a_2 is 3 and that of d_4 is 1, the resulting degree is 1. In addition to the characteristic of the source-consistency of clusters, a cluster is a *complete cluster* if it is source-consistent and contains one entity from each available data source. In the (perfect) clustered result in Figure 9.2, this applies to the four entities a_0–d_0 and a_2–d_2, each forming a separate complete cluster.

CLIP is a two-phase approach that uses the concepts cluster strength, link degree and complete cluster for cluster determination. The first phase only determines complete clusters. To achieve this, the link strength is calculated for all links. Based on the strong links, connected components are then determined on the graph. Complete clusters are returned as the result of the first phase and not processed any further. An example of a complete cluster obtained at the end of the first phase are entities a_0–d_0. The remaining entities and links form the basis for the second phase and are shown in Figure 9.4. By adhering to the priorities set by the link strength, source-consistent clusters are constructed iteratively. This is achieved by redetermining connected components using the remaining strong and all normal links. There are two options for the resulting components. In the case of source-consistency, the clusters are added to the set of result clusters and not processed any further. This applies for c_2 and d_2 creating a cluster and another one for a_3 and b_3. In the latter case of source-inconsistency, components are post-processed as follows. The links within a source-inconsistent component are weighted according to the link characteristics (link strength, link degree, similarity). This weighted list is then processed sequentially to build clusters that are source-consistent. In the running example, this applies to the remaining entities shown in the center of Figure 9.4. Due to the individual weighting, the strongest links are (a_2, b_2) and (b_1, c_1), forming initial clusters. Next, the evaluation of the link (c_1, a_2) leads to a source-inconsistency, so the cluster is not expanded. Finally, a_1 extends the cluster that contains b_1 and c_1. The d_4 entity is associated with the cluster containing a_2 and b_2, as illustrated in the final output in Figure 9.4. Compared to the perfect result of Figure 9.2, the cluster with index 2 is not recognized correctly, because the similarity graph already does not contain links to c_2 and d_2.

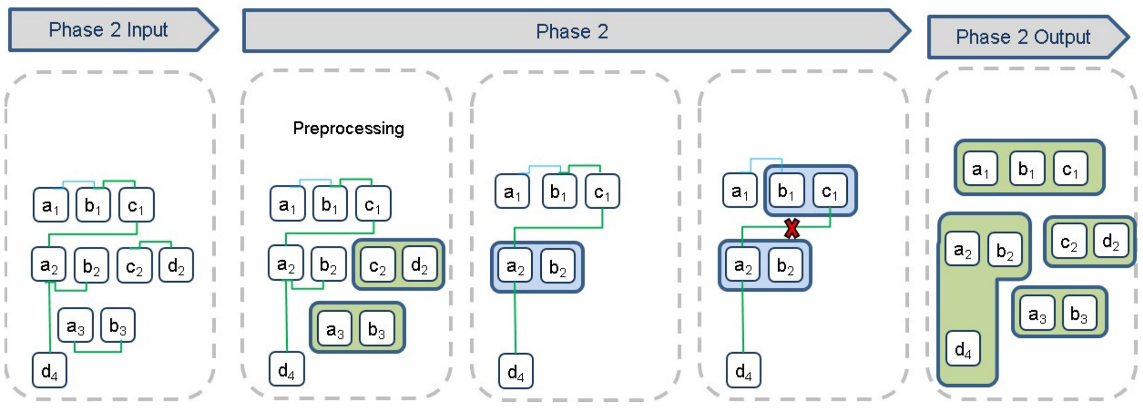


Figure 9.4: Running example processed with CLIP clustering.

9.5 Related Work

Many studies have already investigated link discovery and the related problem of entity resolution (see Chapter 3). Therefore, we focus on related work that explores efficient clustering approaches for link discovery and entity resolution, respectively. Most link discovery tools determine links, but do not process them into a fused representation. Few approaches allow the physical integration of entities in combination with quality assessment, for example as part of a linking framework [25, 123]. While clustering techniques are used to handle Linked Data effectively, there are more extensive comparisons of clustering approaches for entity resolution. As an example, Hassanzadeh et al. [77] provide a comparative evaluation of multiple clustering methods such as star, correlation or Markov clustering for deduplication within a single data source.

To improve the efficiency of link discovery workflows, distributed data processing systems can be used. Previously proposed approaches mainly used Hadoop MapReduce [83, 105, 140]. Both Apache Spark and Apache Flink improve on MapReduce due to better utilization of in-memory processing and better support for iterative algorithms as needed for clustering [172]. Various workflows for entity resolution already rely on Apache Spark [62, 124] or Apache Flink [130, 164].

In [164], the best-performing clustering approaches from [77] are implemented on Apache Flink. To create the same starting point for all clustering approaches, a similarity graph is created in a preliminary step. This graph contains binary match links between all entities of different data sources. Analogously, a similarity graph has already been used in [71, 153].

The evaluation compares simple and established clustering approaches with the two specialized multi-source link discovery approaches CLIP and SplitMerge. Contrary to

Table 9.3: Properties of the utilized datasets.

name	domain	attributes	#entities	#sources	#perfect match pairs	#clusters
DS-G	geography	label, longitude, latitude	3,054	4	4,391	820
DS-M	music	title, length, artist, album, year, language	20,000	5	16,250	10,000
DS-P1	person	name, surname, suburb, postcode	5,000,000	5	3,331,384	3,500,840
DS-P2			10,000,000	10	14,995,973	6,625,848

previous studies [77], the implementations of the clustering approaches on Apache Flink show both runtime and scalability for different domains.

9.6 Evaluation

The goal of this section is a comparative evaluation to show the effectiveness and efficiency of the previously described clustering approaches. We realize all the implementations within the distributed processing framework Apache Flink. The evaluation is based on multiple datasets covering three domains. These datasets are described in the following Section 9.6.1, together with the configuration, e. g., for creating the similarity graph. Section 9.6.2 then compares the resulting match and clustering quality for the different clustering approaches. The evaluation concludes in Section 9.6.3 with an assessment of scalability and runtime behavior.

9.6.1 Datasets and configuration setup

First, we will describe the three datasets from different domains. Table 9.3 shows the most important properties of the already duplicate-free sources. These properties include the number of entities, the thematic classification by domain and attributes as well as information about the expected clustering result.

In the first dataset DS-G, real-world entities from the geographical domain are described. The dataset comes from the instance matching track of OAEI 2011[1] and has already been used by various tools for evaluation. The dataset is limited to 3054 settlement entities from four data sources since the perfect clustering result was determined

[1]OAEI 2011 instance matching http://oaei.ontologymatching.org/2011/instance/

Table 9.4: Default blocking and match configuration for different datasets.

dataset	blocking key	similarity functions	match rule
DS-G	prefixLength1(label)	sim1: JaroWinkler (name) sim2: geographical distance	sim1 $\geq \theta$ & sim2 $\leq$ 1358 km
DS-M	prefixLength1(album)	sim1: trigram (title)	sim1 $\geq \theta$
DS-P1& DS-P2	prefixLength3(surname)	sim1: JaroWinkler (name) sim2: JaroWinkler (surname) sim3: JaroWinkler (suburb) sim4: JaroWinkler (postcode)	sim1≥ 0.9 & sim2 ≥ 0.9 & sim3 $\geq \theta$ & sim4 $\geq \theta$

manually. The records for the music and person domains were created using data generators. Many initial entities are corrupted and duplicated by controlled change operations. Through repeated execution of the data corruption, e. g., with varying attributes or increasing levels of corruption, new artificial data sources can be provided for evaluation. The dataset DS-M is based on real songs from the MusicBrainz database, which are manipulated with the data generator DAPO [82]. In comparison to DS-G, a higher number of attributes and up to five synthetically created data sources present new difficulties for the algorithms under investigation. By far the largest datasets DS-P1 and DS-P2 are based on person data from a voter registry in North Carolina. Again, the data generator (GeCo [32]) utilizes artificial corruptions and attribute omissions to generate duplicates of existing entities. These duplicates are used to simulate additional data sources, which leads to 5 (10) data sources for DS-P1 (DS-P2). With one million entities per data source, the entire dataset can access 5 resp. 10 million entities.

For each of these datasets, we generate a similarity graph as the baseline for clustering. Based on experience from previous work [130, 163] and adapted to the characteristics of the datasets, we use a default configuration for blocking and match parameters. Details on the configuration parameters are shown in Table 9.4. For each of the datasets, standard blocking is applied with the appropriate blocking key. Each similarity function describes the significant properties for determining the similarity for candidates in the individual dataset. The compliance to a match rule results in the creation of a link. Match rules usually contain a combination of logical expressions, e. g., to comply with thresholds for the individual attribute similarities. As we show in the following evaluation regarding effectiveness, the created similarity graph already has a relatively high quality. The clustering approaches are thus faced with the additional challenge of surpassing this baseline.

9.6.2 Comparison Match Quality

For the evaluation of the clustering results, we utilize precision, recall and F-measure to compare the calculated results against perfect results of the datasets. On the three datasets DS-G, DS-M and DS-P2 we compare the quality of the results in Figure 9.5.

The creation of the similarity graph (input graph) is based on the default configuration from the last section with the threshold θ. For each of the datasets, we apply different minimum similarity thresholds θ. This is intended to achieve the highest possible quality for the similarity graph, which should be further improved by clustering. With a higher minimum similarity threshold, the graph contains fewer links, which usually leads to increased precision and decreased recall. A reduction of the threshold may be necessary, for example, if entities in the initial dataset are particularly prone to errors, such as in DS-M. Therefore, the best F-measure values for DS-M can be achieved within the threshold range of $0.35 \leq \theta \leq 0.45$. The resulting F-measure values with a maximum of 0.75 are still below the values of DS-G and DS-P2. With thresholds between 0.75 and 0.9, DS-G and DS-P2 achieve F-measure values for the similarity graph of more than 0.9 and 0.8, respectively. In order to recognize the improvement of the clustering approaches in comparison to the input, Figure 9.5 also shows the quality of the corresponding input (similarity) graph.

A comparison of the clustering approaches reveals that there are substantial differences in the relative match quality. As expected, CC generally achieves the worst values for F-measure, primarily due to the very low precision values. These are caused by the fact that CC links all entities being accessible via arbitrary paths, which results in many false positives. The two Star approaches, which calculate clusters based on neighborhood relationships, show no significant improvement compared to the quality of the input graph. The cluster center determination usually leads to a slight improvement of the recall, but at the expense of precision. The reduced precision is especially noticeable for Star-1 and low θ thresholds. Low θ values provide more similarity links in the input graph, which then form (too) large clusters based on the vertex degree leading to increased numbers of false positives. Compared to the input graph, we recognize a reduction of the F-measure except for DS-G when using Star-2. In general, Star-2 provides better results as it includes the degree of similarity to neighbors in the formation of cluster centers.

In contrast, the two newly introduced algorithms CLIP and SplitMerge achieve excellent clustering quality. In addition, we assess a variation of the SplitMerge approach where the final merge phase is not performed for faster execution (and therefore called Split). These three clustering approaches outperform the previous algorithms (and the

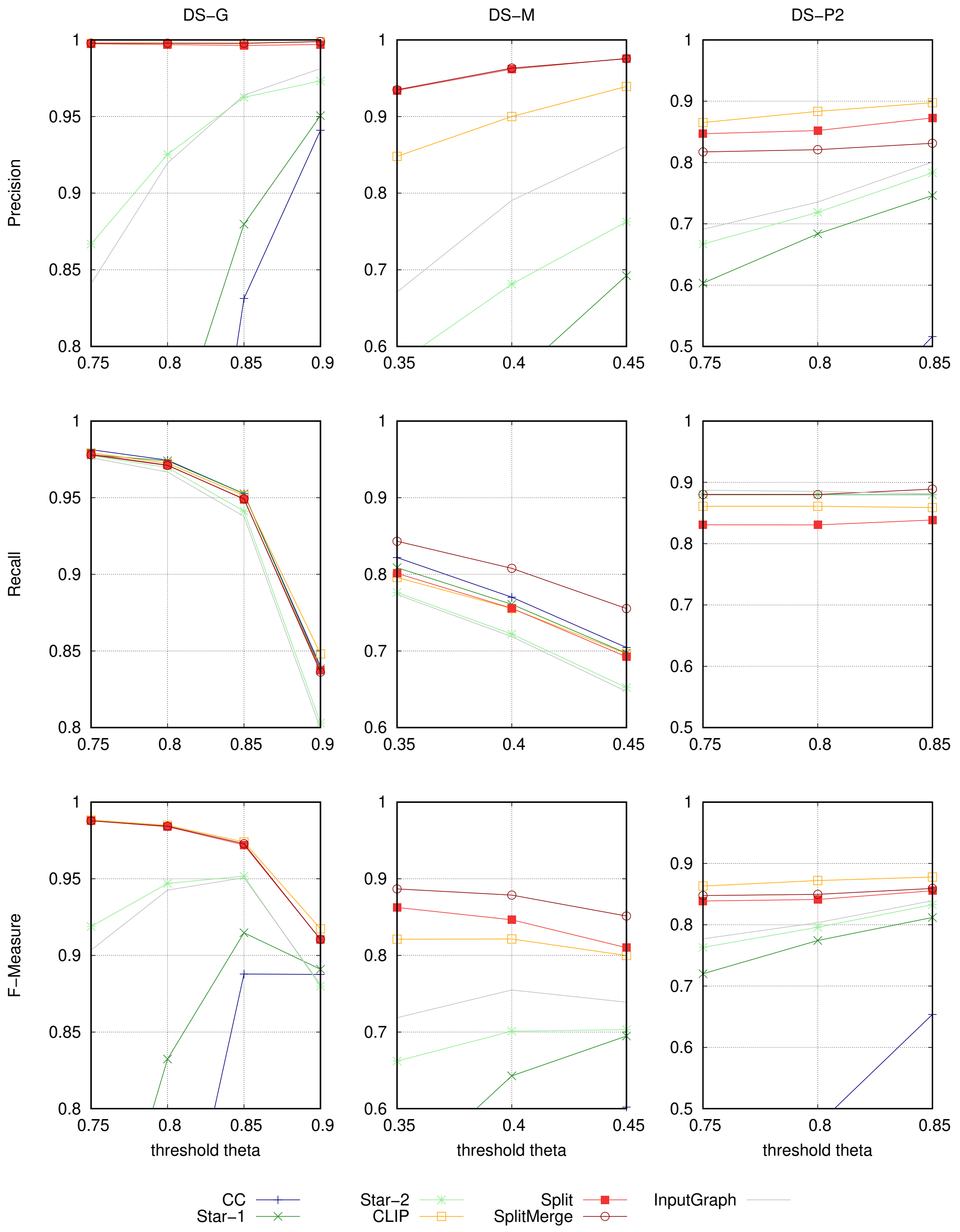


Figure 9.5: Match quality of selected clustering approaches. In terms of the F-measure, the more advanced approaches Split, SplitMerge and CLIP show a superior performance for all datasets.

input graph) in terms of precision and F-measure for all three datasets. SplitMerge requires additional thresholds for the split and merge routine. We tested different values for these thresholds and found that the split threshold should be chosen lower than the similarity threshold θ. As a result, clusters are only split if they contain links with low similarity. By contrast, the merge threshold should be higher than θ so that only very similar clusters are merged. The shown results refer to a fixed setting per dataset, e. g., a split threshold of 0.4 and a merge threshold of 0.8 for DS-G.

Despite generally strong results for CLIP, Split and SplitMerge on the three datasets, strengths and weaknesses for the approaches can be identified. While results for DS-G are fairly even for the three algorithms, CLIP has a slight lead in F-measure because of better recall values for all configurations. For DS-M and DS-P2, we show relevant differences between the three approaches. In particular, SplitMerge has the best recall for both datasets but is only ahead for DS-M in precision. We can identify the reason for the high recall of SplitMerge by comparing it to Split. Since both methods execute the split, the increased recall for SplitMerge is due to the subsequently executed merge. The cluster merge unites entities that cannot be identified by any other algorithm. Therefore, for DS-M, both Split and SplitMerge outperform CLIP and the other approaches in terms of F-measure. For the dataset DS-P2, the recall effect of SplitMerge is not so obvious anymore, because the precision is reduced in the merge phase. However, the resulting F-measure values of SplitMerge are still higher than those of Split. For DS-P2, CLIP has the best precision, good recall values and therefore the best F-measure of all methods.

9.6.3 Runtimes and speedup

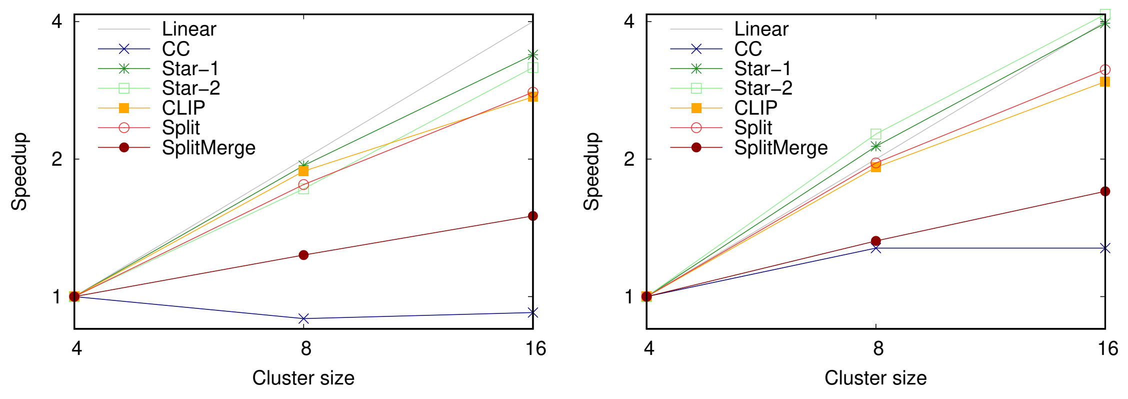
For the assessment of runtime and speedup, it is helpful to provide information on the cluster used. The experiments of the clustering approaches are based on a shared-nothing cluster with 16 workers, each of them running 6 CPUs at $2.5\,\mathrm{GHz}$ with $48\,\mathrm{GiB}$ RAM and $1\,\mathrm{Gbit}$ Ethernet connection. The machines operate on OpenSUSE 13.2, Apache Hadoop 2.6.0 and Apache Flink 1.1.2. The Flink setup makes use of 6 parallel threads and $40\,\mathrm{GiB}$ for each worker. Since some clustering approaches are not executable on 1 or 2 workers due to memory requirements, we evaluate the speedup for 4, 8 and 16 workers with full parallelism. For the experiments, we limit ourselves to the largest datasets DS-P1 and DS-P2 and use the threshold $\theta = 0.8$ to determine the similarity graph. The similarity graph is generated once and used for all clustering approaches, therefore, the runtime is not included. Experiments for each configuration are repeated three times to compute an average runtime.

Table 9.5: Runtimes for all clustering schemes (seconds). A varying number of workers in the cluster shows the scalability of the methods.

	DS-P1			DS-P2		
# workers	4	8	16	4	8	16
CC	51	57	55	101	79	79
Star-1	288	149	85	783	367	197
Star-2	214	124	67	720	317	173
Split	255	145	86	873	445	278
SplitMerge	1754	1423	1168	4792	3618	2819
CLIP	190	101	69	674	351	228

In Table 9.5, the runtime for DS-P1 and DS-P2 is shown for all clustering approaches. As expected, DS-P2 leads to a higher runtime of all approaches by doubling the number of entities. The simplest method calculates connected components (CC) and, as anticipated, has the shortest runtime. Star-1, Star-2, Split and CLIP achieve good runtime on a similar level. The use of additional workers leads to significantly shorter runtimes for all approaches except CC. The SplitMerge approach generally shows the worst runtimes. If the Split approach is executed separately; however, the runtime and also the quality (see Figure 9.5) is very good. Thus the weak point of SplitMerge is in the merge phase. This is partly due to the new calculation of similarities between previously unconnected clusters as the basis for merging. The fact that the merge phase iteratively links clusters creates additional complexity during execution. An improved implementation of CLIP (compared to the initial publication [164]) has significantly improved the runtime of the algorithm and is therefore faster than Split and SplitMerge.

As a final consideration, the runtime will be used to make conclusions regarding the speedup. Therefore, Figure 9.6 shows how the speedup responds for each clustering approach, when the worker capacity is doubled. For both datasets DS-P1 and DS-P2, all clustering approaches except SplitMerge (and the baseline CC) show an almost linear speedup. Especially for CLIP and Split, this shows good scalability despite high-quality results. For the larger dataset DS-P2 the speedup is even better, and for the Star approaches even super-linear. Since the experiments only start with a cluster size of 4 workers, this can also be an unfavorable representation. The elaborate merge phase of SplitMerge also causes bad speedup performance. The speedup is constant as the number of workers increases to 8 or 16, but cannot keep up with Split, CLIP or Star. For the baseline CC, a negative speedup can be observed for DS-P1, since the simple approach cannot use the complete performance of the cluster.



(a) Speedup for dataset DS-P1. **(b)** Speedup for dataset DS-P2.

Figure 9.6: Speedup for varying cluster sizes (number of workers 4, 8 and 16).

9.7 Conclusion

In this chapter, we compared the methodology and results of several multi-source clustering approaches. To enable this comparison on the distributed processing system Apache Flink, equal prerequisites were provided through a standardized workflow in the scalable clustering framework FAMER. An initial similarity graph construction was followed by the individual clustering strategy of each approach to determine a clustered set of entities. We then evaluated the investigated methods concerning the achieved quality and the runtime behavior on very large datasets.

The evaluation of the clustering results clearly shows that the specialized approaches Split, SplitMerge and CLIP achieve a significantly better quality than previously known clustering approaches. In particular, they ensure source-consistent clusters with at most one entity per source as required for duplicate-free sources. Especially Split and CLIP achieve high match quality, good execution times and scalability making them good default approaches for multi-source entity clustering. The runtimes of SplitMerge are too time-consuming for large datasets. However, the improvement in quality for the music domain shows that post-processing of clustering results can be useful. Across all datasets from different domains such as geography, music or person, results were found to be highly dependent on the initial quality of the data and the quality of the calculated similarity graph.

Part IV

Conclusion and Outlook

10 Conclusion and Outlook

10.1 Conclusion

This dissertation elaborated and assessed various approaches for scalable integration of Linked Data. An introductory discussion showed the need for additional methods according to new challenges and identified requirements that should lead to more efficient and effective data integration. An extensive thematic overview then examined the Semantic Web in general and link discovery in particular and concluded with destinctions compared to this thesis. The second part described new strategies for reuse of existing linking results and clustering of entities for multiple Linked Data sources. Following the initial requirements, the third part of the dissertation showed the scalability of the proposed clustering approaches within the distributed processing platform Apache Flink.

10.1.1 Reuse of Link Discovery Results and Clustering Strategies

Previous research had placed little emphasis on the reuse of existing link discovery results. Thus, the design and deployment of the LinkLion platform was an essential contribution to enable the reuse of existing links. The publicly accessible web service allows the storage and retrieval of link discovery results as well as the addition of provenance information. By using standardized vocabularies, provenance information could be provided in a transparent way for further reference. A public SPARQL endpoint was set up to allow the stored link discovery results to be further used in the Web of Data. Overall, the linking repository is a foundation for innovative and holistic methods for data integration, e. g., to recognize new relations via sequences of links using composition methods or to link multiple Linked Data sources with each other.

The data from LinkLion was then reused as a basis for multiple clustering-based approaches to create an integrated knowledge base. A first approach examined the possibilities of linking entities from multiple Linked Data sources. This holistic clustering approach created a consolidated data representation based on existing links and allowed the optimization of clusters with split and merge operators. By including source information and semantic type information in the split phase, correct clusters are determined. All entities in the resulting clusters represent equivalent objects from different data sources. By bundling these entities into a unified cluster representative, fewer comparisons are required in the subsequent merge phase. The evaluation showed that reusing existing links can integrate many previously unconnected entities while avoiding repeated calculations. Furthermore, incorrect links have been detected by analyzing the semantic types and provenance information of the data sources. Overall, the effectiveness and efficiency of the data integration process were improved by the creation of a holistic knowledge base.

An additional study investigates how already existing knowledge graphs can be extended in case of continuously changing data sources. Two generic incremental clustering approaches allow an optimal assignment of new entities while supporting blocking strategies. Besides an incremental base method, a more efficient and effective source-specific approach was presented, which allows an optimal 1:1 assignment of entities to clusters. The assumption that cluster quality decreases through the deployment of incremental clustering approaches was largely confirmed. However, the source-specific approach often delivered only slightly worse precision and recall than static clustering and could even achieve higher quality for a single domain. Interestingly, the evaluation also showed that the order of the data sources in incremental clustering has a decisive influence on the results and that high-quality sources should be used earlier, if possible.

10.1.2 Parallelization of Clustering Strategies

A crucial part of the work focused on distributed dataflow systems to parallelize the proposed static and incremental clustering strategies for improved efficiency. Therefore, it was examined how systems like Apache Flink can process increasingly complex and growing Linked Data sources to enable scalable data integration. For the efficient implementation of SplitMerge, different abstract operators of the Apache Flink platform have been used. In the first step, a grouping of neighboring entities (vertices) based on semantic type characteristics was realized. A second step then removed dissimilar elements from the initial clusters with a vertex-centric iteration. Finally, an iterative merge algorithm ensured that the clusters are merged if they are sufficiently similar. Since parts of the cluster set have no merge partners, some of the result clusters converge faster than others. The DeltaIterate operator thus reduces the number of elements in subsequent iteration steps by transferring only modified elements to the next iteration.

To avoid unnecessary calculations and for efficient execution on distributed systems, support for arbitrary blocking strategies was introduced. By partitioning the data into blocks based on semantic properties (blocking keys), the similarity determination was carried out locally for each block. This distribution on an Apache Flink cluster promises a significantly accelerated execution, e.g., by reduced data exchange over the network. In this blocking process, various possibilities for optimization were investigated to influence the resulting quality and the runtime. Unequal distribution of attribute values can lead to load balancing problems. Resulting skewed blocks are redistributed to several nodes in the cluster to achieve an overall reduced runtime. Regardless of the blocking strategy, the selection of the blocking key could be optimized in most of the cases. For the presented clustering approaches, variable lengths of the blocking keys were used to partition data according to standard blocking. This enabled the blocking key lengths to be adjusted according to the respective domain to reduce the runtime while maintaining a high quality of results.

The evaluation examined both scalability and quality of results for very large datasets with up to 20 million entities. Overall, both the static SplitMerge and the incremental clustering achieved good execution times for datasets from different domains. In addition, the influence of an increasing number of workers in the Apache Flink cluster was evaluated. For the static SplitMerge approach, almost linear speedup values for large parts of the workflow could be determined. However, the complex merge phase reduced this speedup and should only be used if the quality of the results requires further improvement or the runtime is of secondary importance. Omitting the complex merge phase resulted in an excellent runtime behavior with very high resulting cluster quality in many cases. The comparison of this split (only) clustering method with other multi-source clustering approaches on distributed processing systems showed that the performance is competitive.

For the support of dynamic data sources, the scalability of the proposed incremental clustering methods was also analyzed on the Apache Flink platform. In addition to an incremental base approach, a more efficient and effective source-based approach was introduced. It finds a 1:1 assignment between clusters and new entities using the Max-Both strategy. In comparison to static clustering methods, it was observed that a similar runtime is achieved when the static method is allowed to process all increments in one step. However, a realistic approach would require static methods to start a complete recalculation with each increment. This leads to a clear runtime advantage for the proposed incremental approaches. Ultimately, the investigations for the speedup showed an almost linear performance increase of 9.6x on the cluster infrastructure with 16 workers.

10.2 Outlook

This dissertation on the optimization of data integration processes for Linked Data represents an essential contribution to the advancement of the Semantic Web. Increased application of Semantic Web technologies and continuously growing volume of data on the Web requires reliable concepts to integrate data towards interconnected knowledge graphs. This insight has already led to the development of knowledge graphs at large corporations such as Google, Microsoft or Facebook. A very recent article [148] describes these corporate experiences and derives future challenges such as managing changing data sources and scalability for knowledge graph operations. Together with comprehensive findings of this thesis regarding reuse of existing links and clustering strategies on scalable infrastructure, various important directions for future research can be identified.

Reusability and Provenance The presented capabilities to store and maintain existing links together with relevant meta-information allows the reuse in new Semantic Web applications. In addition to describing the metadata using different ontologies, these also allow easy extensibility for other researchers. For instance, the possibility of publishing link specifications between data sources on the LinkLion platform was added by [170]. Thus, relevant and most distinctive properties and thresholds for different domains can automatically be extracted as a starting point for future algorithms. The extensibility of the link repository is used to encourage further research to increase reuse and provenance when integrating Linked Data sources.

The existing LinkLion capabilities for storing provenance information can be extended with additional metadata to increase the value of the platform. In particular, it is necessary to describe the quality of the provided data. Up to now, the addition of links to the repository lacks a validation process. Thus, erroneous data or links with insufficient metadata can be uploaded. This finding has also been supported by an inconsistency detection approach [180] applied to LinkLion. The method was able to detect that links from frameworks like Silk or LIMES have a significantly lower error rate than other algorithms. In the future, the consistency of the data should be checked when it is added. Incorrect data could either be rejected or included in the portal with a semantic description of the error. This extensive data could then be used to investigate how the continuous evolution of links influences the quality when new linking frameworks or updated data sources are utilized. Recent publications [121, 127] also examine link reuse and quality checks, but limit themselves to specific data sources or do not yet deal with errors.

Data integration approaches based on LinkLion already reuse links, but they do not attempt to use any of the included metadata. Therefore, linking and clustering algorithms should be improved to use extensive provenance information for subsequent match decisions. Simple metrics (e. g., based on a single provenance feature) could be used to exclude links of individual frameworks that have produced poor quality results. However, calculating complex quality metrics for a large number of links can be very time-consuming. It is thus essential to examine whether quality measures resulting from the provenance information can be combined to introduce a rating for links. For example, this link rating could positively include the support of different algorithms or frameworks and negatively include a poor consistency check. The resulting rating can then easily be used for any data integration task.

Static and Incremental Clustering for Evolving Knowledge Graphs The presented clustering approaches are a significant contribution towards a physical data integration process for multiple Linked Data sources to form consolidated knowledge bases. A recent study [164] confirms the findings stating that the detection of inconsistencies leads to significantly improved qualitative results. Overall, the investigations have also shown promising opportunities for further research directions to improve effectiveness and efficiency.

Existing linking and clustering approaches typically rely on similarity computations on different entity attributes to identify matching elements. Additional information such as ontologies or information from highly interconnected groups of entities is often neglected. Therefore, it is interesting to study how the context of the surrounding entities (e. g., parental ontology concepts or neighborhood relationships) influences the resulting quality. In particular, it can be investigated if a higher degree of connectivity of the ontology concepts affects the results with an increasing number of data sources.

The cluster representatives represent a central component for the physical data integration. Thus, the selection of optimal property values from equivalent entities to form a cluster representative is a crucial task. So far, this thesis as well as a more recently published related work [61] use simple rule-based procedures such as majority consensus or priority-based selection. Therefore, a detailed investigation of clustering methods with a more sophisticated selection of property values for representatives can further improve the resulting quality.

Besides improving effectiveness, it is also of central importance to optimize the efficiency of the data integration process. The execution of a complex processing pipeline to cluster huge datasets could be demonstrated using the distributed processing system Apache Flink. The additional use of blocking strategies allowed a decisive contribution to the runtime reduction. However, the research has shown that blocking strategies do not react flexibly to the increasing amount of data. Already implemented optimizations such as load balancing support the distribution of the workload, but cannot avoid the initial imbalance. To further increase performance, adaptive blocking methods can be investigated to select the appropriate strategy for the respective dataset. One idea is to study how to automatically execute one or more relevant blocking strategies according to the respective thematic domain and the size of the dataset, leading to an optimal distribution of the data. Already established machine learning methods such as binary classification or genetic programming [18, 51] can be used to find the best configurations while maintaining high-quality results. In this context, the design and implementation of load balancing mechanisms for adaptive blocking strategies on systems like Apache Flink can be a challenge. Therefore, future work should investigate how adaptive blocking methods can be used to partition very large datasets from different domains for data integration tasks on distributed processing systems like Apache Flink or Apache Spark.

Given the increasing number of dynamic Linked Data sources, it is also interesting to investigate the proposed optimizations for incremental knowledge graph integration. To meet the requirements of the integrated knowledge bases, previously missing operations for adaptation must also be supported. The presented approaches already allow the addition of new entities. It is thus of interest to investigate change and delete operations for incremental holistic data integration processes. Besides the coverage of missing operations, the scalability of incremental methods has to be studied for varying data sizes. As the runtime reduction of the proposed incremental clustering methods are lower than expected, future work could investigate how similar methods [35, 71, 190] perform in comparison. Another interesting topic is the combination of continuously changed data sources and the logging of changes for traceability. The availability of information on the temporal sequence of changes for entities then allows the use of systems for temporal graph analysis [161]. Recognized inconsistencies could then lead to repeated checking of the elements added at the same time or restore a backup from an earlier point of time.

Integration into Existing Systems The comparison of different multi-source clustering approaches made it possible to recognize that specialized clustering algorithms using semantic information achieve better results. These results were also achieved by integrating SplitMerge and other clustering approaches into the FAMER framework [162]. In this context, the question arises, how the general comparability and applicability of research approaches within an already existing data integration process is possible.

Future research into scalable data integration strategies should also focus on comparability and ease of use for researchers. Comparability of results can, for instance, already be ensured by benchmarking on platforms such as HOBBIT [96]. Various tasks regarding generation, linking and analysis for very large Linked Data sources are provided to cover components of the Linked Data life cycle. However, coverage by a benchmark does not guarantee that high-performance methods can be used by other researchers in the Semantic Web. Therefore, future data integration approaches should ensure that integration into existing systems is possible. Recent studies [8, 33] confirm this finding with the proposal to create stable platforms for data management and analysis. For the management of Big Data, extensive platforms with easily adaptable modular processes are to be developed. The Semantic Web platform BDE [8] already offers various modules for data processing with established platforms like Apache Spark or Flink. Additionally, modules that enable NLP [175], link discovery [185] or large-scale RDF processing [113] can also be used. Future work could thus integrate supplementary frameworks such as FAMER [162] or Gradoop [97] to cover further data integration or analytics tasks.

This listing of possible future work shows that many interesting problems in the surveyed research areas are still to be adressed. Our contributions for scalable data integration for Linked Data can serve as a basis for further investigations. The findings and complications of other researchers also confirm the conclusions made.

Bibliography

[1] Manel Achichi et al. “Results of the Ontology Alignment Evaluation Initiative 2017”. In: *Proceedings of the 12th International Workshop on Ontology Matching co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017.* Ed. by Pavel Shvaiko et al. Vol. 2032. CEUR Workshop Proceedings. CEUR-WS.org, 2017, pp. 61–113. URL: http://ceur-ws.org/Vol-2032/oaei17_paper0.pdf.

[2] Alsayed Algergawy et al. “Results of the Ontology Alignment Evaluation Initiative 2018”. In: *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, October 8, 2018.* Ed. by Pavel Shvaiko et al. Vol. 2288. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pp. 76–116. URL: http://ceur-ws.org/Vol-2288/oaei18_paper0.pdf.

[3] Arvind Arasu, Venkatesh Ganti, and Raghav Kaushik. “Efficient Exact Set-Similarity Joins”. In: *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, ed. by Umeshwar Dayal et al. ACM, 2006, pp. 918–929. ISBN: 1-59593-385-9. URL: http://dl.acm.org/citation.cfm?id=1164206.

[4] Michael Armbrust et al. “Spark SQL: Relational Data Processing in Spark”. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, ed. by Timos K. Sellis et al. ACM, 2015, pp. 1383–1394. ISBN: 978-1-4503-2758-9. DOI: 10.1145/2723372.2742797.

[5] Natanael Arndt, Patrick Naumann, Norman Radtke, Michael Martin, and Edgard Marx. “Decentralized Collaborative Knowledge Management Using Git”. In: *J. Web Semant.* 54 (2019), pp. 29–47. DOI: 10.1016/j.websem.2018.08.002.

[6] Javed A. Aslam, Ekaterina Pelekhov, and Daniela Rus. “The Star Clustering Algorithm for Static and Dynamic Information Organization”. In: *J. Graph Algorithms Appl.* 8 (2004), pp. 95–129. URL: http://jgaa.info/accepted/2004/Aslam+2004.8.1.pdf.

[7] Sören Auer et al. “Managing the Life-Cycle of Linked Data with the LOD2 Stack”. In: *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part II*, ed. by Philippe Cudré-Mauroux et al. Vol. 7650. Lecture Notes in Computer Science. Springer, 2012, pp. 1–16. ISBN: 978-3-642-35172-3. DOI: 10.1007/978-3-642-35173-0_1.

[8] Sören Auer et al. “The BigDataEurope Platform - Supporting the Variety Dimension of Big Data”. In: *Web Engineering - 17th International Conference, ICWE 2017, Rome, Italy, June 5-8, 2017, Proceedings*, ed. by Jordi Cabot et al. Vol. 10360. Lecture Notes in Computer Science. Springer, 2017, pp. 41–59. ISBN: 978-3-319-60130-4. DOI: 10.1007/978-3-319-60131-1_3.

[9] Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. “Scaling up all pairs similarity search”. In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, ed. by Carey L. Williamson et al. ACM, 2007, pp. 131–140. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242591.

[10] Khalid Belhajjame et al. *PROV Model Primer*. W3C Note. W3C, Apr. 2013. URL: http://www.w3.org/TR/prov-primer/.

[11] Kedar Bellare, Carlo Curino, Ashwin Machanavajihala, Peter Mika, Mandar Rahurkar, and Aamod Sane. “WOO: A Scalable and Multi-tenant Platform for Continuous Knowledge Base Synthesis”. In: *PVLDB* 6.11 (2013), pp. 1114–1125. DOI: 10.14778/2536222.2536236.

[12] T. Berners-Lee, R. Fielding, and L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. The Internet Society, Jan. 2005. URL: http://rfc.net/rfc3986.html.

[13] Tim Berners-Lee. *Linked Data design issues*. W3C design issue document. June 2009. URL: http://www.w3.org/DesignIssues/LinkedData.html.

[14] Tim Berners-Lee. *Metadata Architecture design issues*. W3C design issue document. https://www.w3.org/DesignIssues/Metadata. Jan. 1997. URL: https://www.w3.org/DesignIssues/Metadata.

[15] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web”. In: *Scientific american* 284.5 (2001), pp. 34–43.

[16] Nikos Bikakis and Timos K. Sellis. "Exploration and Visualization in the Web of Big Linked Data: A Survey of the State of the Art". In: *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops 2016, Bordeaux, France, March 15, 2016.* Ed. by Themis Palpanas et al. Vol. 1558. CEUR Workshop Proceedings. CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1558/paper28.pdf.

[17] Nikos Bikakis, Chrisa Tsinaraki, Nektarios Gioldasis, Ioannis Stavrakantonakis, and Stavros Christodoulakis. "The XML and Semantic Web Worlds: Technologies, Interoperability and Integration: A Survey of the State of the Art". In: *Semantic Hyper/Multimedia Adaptation - Schemes and Applications*, ed. by Ioannis Anagnostopoulos et al. Vol. 418. Studies in Computational Intelligence. Springer, 2013, pp. 319–360. ISBN: 978-3-642-28976-7. DOI: 10.1007/978-3-642-28977-4_12.

[18] Mikhail Bilenko, Beena Kamath, and Raymond J. Mooney. "Adaptive Blocking: Learning to Scale Up Record Linkage". In: *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China.* IEEE Computer Society, 2006, pp. 87–96. ISBN: 0-7695-2701-9. DOI: 10.1109/ICDM.2006.13.

[19] Christian Bizer, Tom Heath, and Tim Berners-Lee. "Linked Data - The Story So Far". In: *Int. Journal on Semantic Web and Information Systems (IJSWIS)* 5.3 (Mar. 2009), pp. 1–22. ISSN: 1552-6283. DOI: 10.4018/jswis.2009081901.

[20] Olivier Bodenreider. "The Unified Medical Language System (UMLS): integrating biomedical terminology". In: *Nucleic Acids Research* 32.Database-Issue (2004), pp. 267–270. DOI: 10.1093/nar/gkh061.

[21] Christoph Böhm, Gerard de Melo, Felix Naumann, and Gerhard Weikum. "LINDA: distributed web-of-data-scale entity matching". In: *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, ed. by Xue-wen Chen et al. ACM, 2012, pp. 2104–2108. ISBN: 978-1-4503-1156-4. DOI: 10.1145/2396761.2398582.

[22] Loes M. M. Braun, Floris Wiesman, H. Jaap van den Herik, Arie Hasman, and Erik Korsten. "Towards patient-related information needs". In: *I. J. Medical Informatics* 76.2-3 (2007), pp. 246–251. DOI: 10.1016/j.ijmedinf.2006.03.004.

[23] Dan Brickley and R. V. Guha. *RDF Schema 1.1.* W3C Recommendation. W3C, Feb. 2014. URL: https://www.w3.org/TR/rdf-schema/.

[24] Sergey Brin and Lawrence Page. “The Anatomy of a Large-Scale Hypertextual Web Search Engine”. In: *Computer Networks* 30.1-7 (1998), pp. 107–117. DOI: 10.1016/S0169-7552(98)00110-X.

[25] Volha Bryl et al. “Interlinking and Knowledge Fusion”. In: *Linked Open Data - Creating Knowledge Out of Interlinked Data - Results of the LOD2 Project*, ed. by Sören Auer et al. Vol. 8661. Lecture Notes in Computer Science. Springer, 2014, pp. 70–89. ISBN: 978-3-319-09845-6. DOI: 10.1007/978-3-319-09846-3_4.

[26] Lorenz Bühmann, Jens Lehmann, Patrick Westphal, and Simon Bin. “DL-Learner Structured Machine Learning on Semantic Web Data”. In: *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, ed. by Pierre-Antoine Champin et al. ACM, 2018, pp. 467–471. DOI: 10.1145/3184558.3186235.

[27] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. “Apache Flink™: Stream and Batch Processing in a Single Engine”. In: *IEEE Data Eng. Bull.* 38.4 (2015), pp. 28–38. URL: http://sites.computer.org/debull/A15dec/p28.pdf.

[28] Michelle Cheatham and Pascal Hitzler. “String Similarity Metrics for Ontology Alignment”. In: *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, ed. by Harith Alani et al. Vol. 8219. Lecture Notes in Computer Science. Springer, 2013, pp. 294–309. ISBN: 978-3-642-41337-7. DOI: 10.1007/978-3-642-41338-4_19.

[29] Ying Chen, JD Elenee Argentinis, and Griff Weber. “IBM Watson: How Cognitive Computing Can Be Applied to Big Data Challenges in Life Sciences Research”. In: *Clinical Therapeutics* 38.4 (2016), pp. 688–701. ISSN: 0149-2918. DOI: https://doi.org/10.1016/j.clinthera.2015.12.001.

[30] Peter Christen. “A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication”. In: *IEEE Trans. Knowl. Data Eng.* 24.9 (2012), pp. 1537–1555. DOI: 10.1109/TKDE.2011.127.

[31] Peter Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer, 2012. ISBN: 978-3-642-31163-5. DOI: 10.1007/978-3-642-31164-2.

[32] Peter Christen and Dinusha Vatsalan. "Flexible and extensible generation and corruption of personal data". In: *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, ed. by Qi He et al. ACM, 2013, pp. 1165–1168. ISBN: 978-1-4503-2263-8. DOI: 10.1145/2505515.2507815.

[33] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. "End-to-End Entity Resolution for Big Data: A Survey". In: *CoRR* abs/1905.06397 (2019). arXiv: 1905.06397.

[34] Mihai Codescu, Daniel Couto Vale, Oliver Kutz, and Till Mossakowski. "Ontology-based Route Planning for OpenStreetMap". In: *Proceedings of the Terra Cognita Workshop on Foundations, Technologies and Applications of the Geospatial Web, Boston, USA, November 12, 2012*, ed. by Dave Kolas et al. Vol. 901. CEUR Workshop Proceedings. CEUR-WS.org, 2012, pp. 62–73. URL: http://ceur-ws.org/Vol-901/paper6.pdf.

[35] Gianni Costa, Giuseppe Manco, and Riccardo Ortale. "An incremental clustering scheme for data de-duplication". In: *Data Min. Knowl. Discov.* 20.1 (2010), pp. 152–187. DOI: 10.1007/s10618-009-0155-0.

[36] Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. "AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies". In: *PVLDB* 2.2 (2009), pp. 1586–1589. DOI: 10.14778/1687553.1687598.

[37] Richard Cyganiak, David Wood, and Markus Lanthaler. *Resource Description Framework (RDF) 1.1 Concepts and Abstract Syntax*. W3C Recommendation. W3C, Feb. 2014. URL: https://www.w3.org/TR/rdf11-concepts/.

[38] Data Artisans. *data Artisans Streaming Ledger - Serializable ACID Transactions on Streaming Data*. Tech. rep. Data Artisans, 2018.

[39] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". In: *6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004*, ed. by Eric A. Brewer et al. USENIX Association, 2004, pp. 137–150. URL: http://www.usenix.org/events/osdi04/tech/dean.html.

[40] Jeremy Debattista, Christoph Lange, Sören Auer, and Dominic Cortis. "Evaluating the quality of the LOD cloud: An empirical investigation". In: *Semantic Web* 9.6 (2018), pp. 859–901. DOI: 10.3233/SW-180306.

[41] Jack B. Dennis. "First version of a data flow procedure language". In: *Programming Symposium, Proceedings Colloque sur la Programmation, Paris, France, April 9-11, 1974*, ed. by Bernard Robinet. Vol. 19. Lecture Notes in Computer Science. Springer, 1974, pp. 362–376. ISBN: 3-540-06859-7. DOI: 10.1007/3-540-06859-7_145.

[42] Hong Hai Do and Erhard Rahm. "COMA - A System for Flexible Combination of Schema Matching Approaches". In: *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002*. Morgan Kaufmann, 2002, pp. 610–621. ISBN: 978-1-55860-869-6. DOI: 10.1016/B978-155860869-6/50060-3.

[43] AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012. ISBN: 978-0-12-416044-6. URL: http://research.cs.wisc.edu/dibook/.

[44] Xin Luna Dong and Theodoros Rekatsinas. "Data Integration and Machine Learning: A Natural Synergy". In: *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, ed. by Gautam Das et al. ACM, 2018, pp. 1645–1650. DOI: 10.1145/3183713.3197387.

[45] Xin Luna Dong and Divesh Srivastava. *Big Data Integration*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2015. DOI: 10.2200/S00578ED1V01Y201404DTM040.

[46] Xin Dong et al. "Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion". In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, ed. by Sofus A. Macskassy et al. ACM, 2014, pp. 601–610. ISBN: 978-1-4503-2956-9. DOI: 10.1145/2623330.2623623.

[47] Uwe Draisbach and Felix Naumann. "A generalization of blocking and windowing algorithms for duplicate detection". In: *2011 International Conference on Data and Knowledge Engineering, ICDKE 2011, Milano, Italy, September 6, 2011*, ed. by Ji Zhang et al. IEEE, 2011, pp. 18–24. ISBN: 978-1-4577-0865-7. DOI: 10.1109/ICDKE.2011.6053920.

[48] M. Duerst and M. Suignard. *Internationalized Resource Identifiers (IRIs)*. RFC 3987. The Internet Society, Jan. 2005. URL: https://tools.ietf.org/html/rfc3987.

[49] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. “Duplicate Record Detection: A Survey”. In: *IEEE Trans. Knowl. Data Eng.* 19.1 (2007), pp. 1–16. DOI: 10.1109/TKDE.2007.250581.

[50] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, 2007. DOI: 10.1007/978-3-540-49612-0.

[51] Luiz Osvaldo Evangelista, Eli Cortez, Altigran Soares da Silva, and Wagner Meira Jr. “Adaptive and Flexible Blocking for Record Linkage Tasks”. In: *JIDM* 1.2 (2010), pp. 167–182. URL: http://seer.lcc.ufmg.br/index.php/jidm/article/view/45.

[52] Daniel Faria, Ernesto Jiménez-Ruiz, Catia Pesquita, Emanuel Santos, and Francisco M. Couto. “Towards Annotating Potential Incoherences in BioPortal Mappings”. In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*, ed. by Peter Mika et al. Vol. 8797. Lecture Notes in Computer Science. Springer, 2014, pp. 17–32. ISBN: 978-3-319-11914-4. DOI: 10.1007/978-3-319-11915-1_2.

[53] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F. Cruz, and Francisco M. Couto. “The AgreementMakerLight Ontology Matching System”. In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences - Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings*, ed. by Robert Meersman et al. Vol. 8185. Lecture Notes in Computer Science. Springer, 2013, pp. 527–541. ISBN: 978-3-642-41029-1. DOI: 10.1007/978-3-642-41030-7_38.

[54] Daniel Faria et al. “OAEI 2016 results of AML”. In: *Proceedings of the 11th International Workshop on Ontology Matching co-located with the 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 18, 2016.* Ed. by Pavel Shvaiko et al. Vol. 1766. CEUR Workshop Proceedings. CEUR-WS.org, 2016, pp. 138–145. URL: http://ceur-ws.org/Vol-1766/oaei16_paper2.pdf.

[55] Daniel Faria et al. “Results of AML in OAEI 2017”. In: *Proceedings of the 12th International Workshop on Ontology Matching co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017.* Ed. by Pavel Shvaiko et al. Vol. 2032. CEUR Workshop Proceedings. CEUR-WS.org, 2017, pp. 122–128. URL: http://ceur-ws.org/Vol-2032/oaei17_paper2.pdf.

[56] David C. Faye, Olivier Curé, and Guillaume Blin. "A survey of RDF storage approaches". In: *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées* 15 (2012), pp. 11–35. URL: https://hal.inria.fr/hal-01299496.

[57] Ivan P. Fellegi and Alan B. Sunter. "A Theory for Record Linkage". English. In: *Journal of the American Statistical Association* 64.328 (1969), pp. 1183–1210. ISSN: 01621459.

[58] Alfio Ferrara, Andriy Nikolov, and François Scharffe. "Data Linking for the Semantic Web". In: *Int. J. Semantic Web Inf. Syst.* 7.3 (2011), pp. 46–76. DOI: 10.4018/jswis.2011070103.

[59] David A. Ferrucci et al. "Building Watson: An Overview of the DeepQA Project". In: *AI Magazine* 31.3 (2010), pp. 59–79. URL: http://www.aaai.org/ojs/index.php/aimagazine/article/view/2303.

[60] Martin Franke, Ziad Sehili, Marcel Gladbach, and Erhard Rahm. "Post-processing Methods for High Quality Privacy-Preserving Record Linkage". In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings*, ed. by Joaquín García-Alfaro et al. Vol. 11025. Lecture Notes in Computer Science. Springer, 2018, pp. 263–278. ISBN: 978-3-030-00304-3. DOI: 10.1007/978-3-030-00305-0_19.

[61] Johannes Frey, Marvin Hofer, Daniel Obraczka, Jens Lehmann, and Sebastian Hellmann. "DBpedia FlexiFusion The Best of Wikipedia > Wikidata > Your Data". In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, ed. by Chiara Ghidini et al. Vol. 11779. Lecture Notes in Computer Science. Springer, 2019, pp. 96–112. ISBN: 978-3-030-30795-0. DOI: 10.1007/978-3-030-30796-7_7.

[62] Luca Gagliardelli, Song Zhu, Giovanni Simonini, and Sonia Bergamaschi. "Bigdedup: a Big Data integration toolkit for duplicate detection in industrial scenarios". In: *25th International Conference on Transdisciplinary Engineering (TE2018).* Vol. 7. 2018, pp. 1015–1023.

[63] D. Gale and L. S. Shapley. "College Admissions and the Stability of Marriage". In: *The American Mathematical Monthly* 120.5 (2013), pp. 386–391. DOI: 10.4169/amer.math.monthly.120.05.386.

[64] Diego García-Gil, Sergio Ramírez-Gallego, Salvador García, and Francisco Herrera. “A comparison on scalability for batch big data processing on Apache Spark and Apache Flink”. In: *Big Data Analytics* 2.1 (Mar. 2017). DOI: 10.1186/s41044-016-0020-2.

[65] Kleanthi Georgala, Mohamed Ahmed Sherif, and Axel-Cyrille Ngonga Ngomo. “An Efficient Approach for the Generation of Allen Relations”. In: *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, ed. by Gal A. Kaminka et al. Vol. 285. Frontiers in Artificial Intelligence and Applications. IOS Press, 2016, pp. 948–956. ISBN: 978-1-61499-671-2. DOI: 10.3233/978-1-61499-672-9-948.

[66] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. “Similarity Search in High Dimensions via Hashing”. In: *VLDB’99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, ed. by Malcolm P. Atkinson et al. Morgan Kaufmann, 1999, pp. 518–529. ISBN: 1-55860-615-7. URL: http://www.vldb.org/conf/1999/P49.pdf.

[67] Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. “PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs”. In: *10th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2012, Hollywood, CA, USA, October 8-10, 2012*, ed. by Chandu Thekkath et al. USENIX Association, 2012, pp. 17–30. ISBN: 978-1-931971-96-6. URL: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/gonzalez.

[68] William D. Gropp, Ewing L. Lusk, and Anthony Skjellum. *Using MPI: portable parallel programming with the message-passing interface, 2nd Edition*. Scientific and engineering computation series. MIT Press, 1999. ISBN: 026257134X. URL: http://www.worldcat.org/oclc/41548279.

[69] Paul Groth and Luc Moreau. *PROV-Overview*. W3C Note. W3C, Apr. 2013. URL: https://www.w3.org/TR/prov-overview/.

[70] Tom Gruber. “Ontology”. In: *Encyclopedia of Database Systems, Second Edition*, ed. by Ling Liu et al. Springer, 2018, pp. 2574–2576. ISBN: 978-1-4614-8266-6. DOI: 10.1007/978-1-4614-8265-9_1318.

[71] Anja Gruenheid, Xin Luna Dong, and Divesh Srivastava. “Incremental Record Linkage”. In: *PVLDB* 7.9 (2014), pp. 697–708. DOI: 10.14778/2732939.2732943.

[72] Toni Grütze, Christoph Böhm, and Felix Naumann. "Holistic and Scalable Ontology Alignment for Linked Open Data". In: *WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012*, ed. by Christian Bizer et al. Vol. 937. CEUR Workshop Proceedings. CEUR-WS.org, 2012. URL: http://ceur-ws.org/Vol-937/ldow2012-paper-08.pdf.

[73] Kun Guo, Wenzhong Guo, Yuzhong Chen, Qirong Qiu, and Qishan Zhang. "Community discovery by propagating local and global information based on the MapReduce model". In: *Inf. Sci.* 323 (2015), pp. 73–93. DOI: 10.1016/j.ins.2015.06.032.

[74] Peter Haase, Tobias Mathäß, and Michael Ziller. "An evaluation of approaches to federated query processing over linked data". In: *Proceedings the 6th International Conference on Semantic Systems, I-SEMANTICS 2010, Graz, Austria, September 1-3, 2010*, ed. by Adrian Paschke et al. ACM International Conference Proceeding Series. ACM, 2010. ISBN: 978-1-4503-0014-8. DOI: 10.1145/1839707.1839713.

[75] Olaf Hartig and Bryan Thompson. "Foundations of an Alternative Approach to Reification in RDF". In: *CoRR* abs/1406.3399 (2014). arXiv: 1406.3399.

[76] Michael Hartung, Anika Groß, and Erhard Rahm. "Composition Methods for Link Discovery". In: *Datenbanksysteme für Business, Technologie und Web (BTW), 15. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 11.-15.3.2013 in Magdeburg, Germany. Proceedings*, ed. by Volker Markl et al. Vol. 214. LNI. GI, 2013, pp. 261–277. ISBN: 978-3-88579-608-4. URL: https://dl.gi.de/20.500.12116/17325.

[77] Oktie Hassanzadeh, Fei Chiang, Renée J. Miller, and Hyun Chul Lee. "Framework for Evaluating Clustering Algorithms in Duplicate Detection". In: *PVLDB* 2.1 (2009), pp. 1282–1293. DOI: 10.14778/1687627.1687771.

[78] Oktie Hassanzadeh et al. "Discovering Linkage Points over Web Data". In: *PVLDB* 6.6 (2013), pp. 444–456. DOI: 10.14778/2536336.2536345.

[79] Jim Hendler and Theresa A. Pardo. *A Primer on Machine Readability for Online Documents and Data*. 2012. URL: https://perma.cc/B9N6-X3GL.

[80] Daniel Hernández, Aidan Hogan, and Markus Krötzsch. "Reifying RDF: What Works Well With Wikidata?" In: *Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, October 11, 2015*. Ed. by Thorsten Liebig et al. Vol. 1457. CEUR Workshop Proceedings. CEUR-WS.org,

2015, pp. 32–47. URL: http://ceur-ws.org/Vol-1457/SSWS2015_paper3.pdf.

[81] Mauricio A. Hernández and Salvatore J. Stolfo. “The Merge/Purge Problem for Large Databases”. In: *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA, May 22-25, 1995.* Ed. by Michael J. Carey et al. ACM Press, 1995, pp. 127–138. DOI: 10.1145/223784.223807.

[82] K. Hildebrandt, F. Panse, N. Wilcke, and N. Ritter. “Large-Scale Data Pollution with Apache Spark”. In: *IEEE Transactions on Big Data* (2018), pp. 1–1. ISSN: 2332-7790. DOI: 10.1109/TBDATA.2016.2637378.

[83] Stanley Hillner and Axel-Cyrille Ngonga Ngomo. “Parallelizing LIMES for large-scale link discovery”. In: *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011*, ed. by Chiara Ghidini et al. ACM International Conference Proceeding Series. ACM, 2011, pp. 9–16. ISBN: 978-1-4503-0621-8. DOI: 10.1145/2063518.2063520.

[84] I. G. Husein, S. Akbar, B. Sitohang, and F. N. Azizah. “Review of ontology matching with background knowledge”. In: *2016 International Conference on Data and Software Engineering (ICoDSE).* IEEE. Oct. 2016, pp. 1–6. DOI: 10.1109/ICODSE.2016.7936159.

[85] Renato Iannella. *Semantic Web Architectures.* Tech. rep. Semantic Identity, Aug. 2014.

[86] Robert Isele and Christian Bizer. “Active learning of expressive linkage rules using genetic programming”. In: *J. Web Sem.* 23 (2013), pp. 2–15. DOI: 10.1016/j.websem.2013.06.001.

[87] Robert Isele and Christian Bizer. “Learning Expressive Linkage Rules using Genetic Programming”. In: *PVLDB* 5.11 (2012), pp. 1638–1649. DOI: 10.14778/2350229.2350276.

[88] Robert Isele, Anja Jentzsch, and Christian Bizer. “Efficient Multidimensional Blocking for Link Discovery without losing Recall”. In: *Proceedings of the 14th International Workshop on the Web and Databases 2011, WebDB 2011, Athens, Greece, June 12, 2011*, ed. by Amélie Marian et al. 2011.

[89] Robert Isele, Anja Jentzsch, and Christian Bizer. “Silk Server - Adding missing Links while consuming Linked Data”. In: *Proceedings of the First International Workshop on Consuming Linked Data, Shanghai, China, November 8, 2010*, ed. by Olaf Hartig et al. Vol. 665. CEUR Workshop Proceedings. CEUR-WS.org, 2010. URL: http://ceur-ws.org/Vol-665/IseleEtAl_COLD2010.pdf.

[90] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. “LogMap: Logic-Based and Scalable Ontology Matching”. In: *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, ed. by Lora Aroyo et al. Vol. 7031. Lecture Notes in Computer Science. Springer, 2011, pp. 273–288. ISBN: 978-3-642-25072-9. DOI: 10.1007/978-3-642-25073-6_18.

[91] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, and Valerie Cross. “LogMap family participation in the OAEI 2017”. In: *Proceedings of the 12th International Workshop on Ontology Matching co-located with the 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 21, 2017.* Ed. by Pavel Shvaiko et al. Vol. 2032. CEUR Workshop Proceedings. CEUR-WS.org, 2017, pp. 153–157. URL: http://ceur-ws.org/Vol-2032/oaei17_paper7.pdf.

[92] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, and Valerie V. Cross. “LogMap family participation in the OAEI 2016”. In: *Proceedings of the 11th International Workshop on Ontology Matching co-located with the 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 18, 2016.* Ed. by Pavel Shvaiko et al. Vol. 1766. CEUR Workshop Proceedings. CEUR-WS.org, 2016, pp. 185–189. URL: http://ceur-ws.org/Vol-1766/oaei16_paper9.pdf.

[93] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, and Ian Horrocks. “LogMap and LogMapLt results for OAEI 2013”. In: *Proceedings of the 8th International Workshop on Ontology Matching co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 21, 2013.* Ed. by Pavel Shvaiko et al. Vol. 1111. CEUR Workshop Proceedings. CEUR-WS.org, 2013, pp. 131–138. URL: http://ceur-ws.org/Vol-1111/oaei13_paper5.pdf.

[94] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Alessandro Solimando, and Valerie V. Cross. “LogMap family results for OAEI 2015”. In: *Proceedings of the 10th International Workshop on Ontology Matching collocated with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, October 12, 2015.* Ed. by Pavel Shvaiko et al. Vol. 1545. CEUR Workshop Proceedings.

CEUR-WS.org, 2015, pp. 171–175. URL: http://ceur-ws.org/Vol-1545/oaei15_paper10.pdf.

[95] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Yujiao Zhou, and Ian Horrocks. "Large-scale Interactive Ontology Matching: Algorithms and Implementation". In: *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, ed. by Luc De Raedt et al. Vol. 242. Frontiers in Artificial Intelligence and Applications. IOS Press, 2012, pp. 444–449. ISBN: 978-1-61499-097-0. DOI: 10.3233/978-1-61499-098-7-444.

[96] Ernesto Jiménez-Ruiz et al. "Introducing the HOBBIT platform into the ontology alignment evaluation campaign". In: *Proceedings of the 13th International Workshop on Ontology Matching co-located with the 17th International Semantic Web Conference, OM@ISWC 2018, Monterey, CA, USA, October 8, 2018.* Ed. by Pavel Shvaiko et al. Vol. 2288. CEUR Workshop Proceedings. CEUR-WS.org, 2018, pp. 49–60. URL: http://ceur-ws.org/Vol-2288/om2018_LTpaper5.pdf.

[97] Martin Junghanns, Max Kießling, Niklas Teichmann, Kevin Gómez, André Petermann, and Erhard Rahm. "Declarative and distributed graph analytics with GRADOOP". In: *PVLDB* 11.12 (2018), pp. 2006–2009. DOI: 10.14778/3229863.3236246.

[98] Marc Kaepke and Olaf Zukunft. "A Comparative Evaluation of Big Data Frameworks for Graph Processing". In: *4th International Conference on Big Data Innovations and Applications, Innovate-Data 2018, Barcelona, Spain, August 6-8, 2018.* IEEE, 2018, pp. 30–37. ISBN: 978-1-5386-7793-3. DOI: 10.1109/Innovate-Data.2018.00012.

[99] Vasiliki Kalavri, Vladimir Vlassov, and Seif Haridi. "High-Level Programming Abstractions for Distributed Graph Processing". In: *IEEE Trans. Knowl. Data Eng.* 30.2 (2018), pp. 305–324. DOI: 10.1109/TKDE.2017.2762294.

[100] Jan-Christoph Kalo, Silviu Homoceanu, Jewgeni Rose, and Wolf-Tilo Balke. "Avoiding Chinese Whispers: Controlling End-to-End Join Quality in Linked Open Data Stores". In: *Proceedings of the ACM Web Science Conference, WebSci 2015, Oxford, United Kingdom, June 28 - July 1, 2015*, ed. by David De Roure et al. ACM, 2015, 5:1–5:10. ISBN: 978-1-4503-3672-7. DOI: 10.1145/2786451.2786466.

[101] Ali Khalili and Sören Auer. "User interfaces for semantic authoring of textual content: A systematic literature review". In: *J. Web Semant.* 22 (2013), pp. 1–18. DOI: 10.1016/j.websem.2013.08.004.

[102] Michael Kifer and Harold Boley. *RIF Overview (Second Edition).* W3C Recommendation. W3C, Feb. 2013. URL: https://www.w3.org/TR/rif-overview/.

[103] Toralf Kirsten, Anika Groß, Michael Hartung, and Erhard Rahm. "GOMMA: a component-based infrastructure for managing and analyzing life science ontologies and their evolution". In: *J. Biomedical Semantics* 2 (2011), p. 6. DOI: 10.1186/2041-1480-2-6.

[104] Jannis Koch, Christian L. Staudt, Maximilian Vogel, and Henning Meyerhenke. "An empirical comparison of Big Graph frameworks in the context of network analysis". In: *Social Netw. Analys. Mining* 6.1 (2016), 84:1–84:20. DOI: 10.1007/s13278-016-0394-1.

[105] Lars Kolb, Andreas Thor, and Erhard Rahm. "Dedoop: Efficient Deduplication with Hadoop". In: *PVLDB* 5.12 (2012), pp. 1878–1881. DOI: 10.14778/2367502.2367527.

[106] Lars Kolb, Andreas Thor, and Erhard Rahm. "Load Balancing for MapReduce-based Entity Resolution". In: *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, ed. by Anastasios Kementsietsidis et al. IEEE Computer Society, 2012, pp. 618–629. ISBN: 978-0-7695-4747-3. DOI: 10.1109/ICDE.2012.22.

[107] Lars Kolb, Andreas Thor, and Erhard Rahm. "Multi-pass sorted neighborhood blocking with MapReduce". In: *Computer Science - R&D* 27.1 (2012), pp. 45–63. DOI: 10.1007/s00450-011-0177-x.

[108] Lars Kolb, Andreas Thor, and Erhard Rahm. "Parallel Sorted Neighborhood Blocking with MapReduce". In: *Datenbanksysteme für Business, Technologie und Web (BTW), 14. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 2.-4.3.2011 in Kaiserslautern, Germany*, ed. by Theo Härder et al. Vol. 180. LNI. GI, 2011, pp. 45–64. ISBN: 978-3-88579-274-1. URL: https://dl.gi.de/20.500.12116/19619.

[109] Hanna Köpcke and Erhard Rahm. "Frameworks for entity matching: A comparison". In: *Data Knowl. Eng.* 69.2 (2010), pp. 197–210. DOI: 10.1016/j.datak.2009.10.003.

[110] Hanna Köpcke, Andreas Thor, and Erhard Rahm. "Evaluation of entity resolution approaches on real-world match problems". In: *PVLDB* 3.1 (2010), pp. 484–493. DOI: 10.14778/1920841.1920904.

[111] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. "The SIDER database of drugs and side effects". In: *Nucleic Acids Research* 44.Database-Issue (2016), pp. 1075–1079. DOI: 10.1093/nar/gkv1075.

[112] Kyong-Ha Lee, Yoon-Joon Lee, Hyunsik Choi, Yon Dohn Chung, and Bongki Moon. "Parallel data processing with MapReduce: a survey". In: *SIGMOD Record* 40.4 (2011), pp. 11–20. DOI: 10.1145/2094114.2094118.

[113] Jens Lehmann et al. "Distributed Semantic Analytics Using the SANSA Stack". In: *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, ed. by Claudia d'Amato et al. Vol. 10588. Lecture Notes in Computer Science. Springer, 2017, pp. 147–155. ISBN: 978-3-319-68203-7. DOI: 10.1007/978-3-319-68204-4_15.

[114] Maurizio Lenzerini. "Data Integration: A Theoretical Perspective". In: *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, ed. by Lucian Popa et al. ACM, 2002, pp. 233–246. ISBN: 1-58113-507-6. DOI: 10.1145/543613.543644.

[115] Ulf Leser and Felix Naumann. *Informationsintegration - Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen.* dpunkt.verlag, 2007. URL: https://www.dpunkt.de/openbooks/informationsintegration.pdf.

[116] Jinfeng Li et al. "A comparison of general-purpose distributed systems for data processing". In: *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, ed. by James Joshi et al. IEEE Computer Society, 2016, pp. 378–383. ISBN: 978-1-4673-9005-7. DOI: 10.1109/BigData.2016.7840626.

[117] Vanessa López, Christina Unger, Philipp Cimiano, and Enrico Motta. "Evaluating question answering over linked data". In: *J. Web Semant.* 21 (2013), pp. 3–13. DOI: 10.1016/j.websem.2013.05.006.

[118] Zongmin Ma, Miriam A. M. Capretz, and Li Yan. "Storing massive Resource Description Framework (RDF) data: a survey". In: *Knowledge Eng. Review* 31.4 (2016), pp. 391–413. DOI: 10.1017/S0269888916000217.

[119] Grzegorz Malewicz et al. “Pregel: a system for large-scale graph processing”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, ed. by Ahmed K. Elmagarmid et al. ACM, 2010, pp. 135–146. ISBN: 978-1-4503-0032-2. DOI: 10.1145/1807167.1807184.

[120] Nicolas Marie and Fabien L. Gandon. “Survey of Linked Data Based Exploration Systems”. In: *Proceedings of the 3rd International Workshop on Intelligent Exploration of Semantic Data (IESD 2014) co-located with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Italy, October 20, 2014.* Ed. by Dhavalkumar Thakker et al. Vol. 1279. CEUR Workshop Proceedings. CEUR-WS.org, 2014. URL: http://ceur-ws.org/Vol-1279/iesd14_8.pdf.

[121] Alan Meehan, Dimitris Kontokostas, Markus Freudenberg, Rob Brennan, and Declan O’Sullivan. “Validating Interlinks Between Linked Data Datasets with the SUMMR Methodology”. In: *On the Move to Meaningful Internet Systems: OTM 2016 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2016, Rhodes, Greece, October 24-28, 2016, Proceedings*, ed. by Christophe Debruyne et al. Vol. 10033. Lecture Notes in Computer Science. 2016, pp. 654–672. ISBN: 978-3-319-48471-6. DOI: 10.1007/978-3-319-48472-3_39.

[122] Imen Megdiche, Olivier Teste, and Cássia Trojahn dos Santos. “An Extensible Linear Approach for Holistic Ontology Matching”. In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, ed. by Paul T. Groth et al. Vol. 9981. Lecture Notes in Computer Science. 2016, pp. 393–410. ISBN: 978-3-319-46522-7. DOI: 10.1007/978-3-319-46523-4_24.

[123] Pablo N. Mendes, Hannes Mühleisen, and Christian Bizer. “Sieve: linked data quality assessment and fusion”. In: *Proceedings of the 2012 Joint EDBT/ICDT Workshops, Berlin, Germany, March 30, 2012*, ed. by Divesh Srivastava et al. ACM, 2012, pp. 116–123. ISBN: 978-1-4503-1143-4. DOI: 10.1145/2320765.2320803.

[124] Demetrio Gomes Mestre, Carlos Eduardo Santos Pires, Dimas C. Nascimento, Andreza Raquel Monteiro de Queiroz, Veruska Borges Santos, and Tiago Brasileiro Araújo. “An efficient spark-based adaptive windowing for entity matching”. In: *Journal of Systems and Software* 128 (2017), pp. 1–10. DOI: 10.1016/j.jss.2017.03.003.

[125] Seunghyeon Moon, Jae-Gil Lee, Minseo Kang, Minsoo Choy, and Jin-Woo Lee. “Parallel community detection on large graphs with MapReduce and GraphChi”.

In: *Data Knowl. Eng.* 104 (2016), pp. 17–31. DOI: 10.1016/j.datak.2015.05.001.

[126] Luc Moreau and Paul T. Groth. *Provenance: An Introduction to PROV.* Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, 2013. ISBN: 9781627052214. DOI: 10.2200/S00528ED1V01Y201308WBE007.

[127] Michalis Mountantonakis and Yannis Tzitzikas. "High Performance Methods for Linked Open Data Connectivity Analytics". In: *Information* 9.6 (2018), p. 134. DOI: 10.3390/info9060134.

[128] James Munkres. "Algorithms for the assignment and transportation problems". In: *Journal of the society for industrial and applied mathematics* 5.1 (1957), pp. 32–38.

[129] Markus Nentwig, Anika Groß, Maximilian Möller, and Erhard Rahm. "Distributed Holistic Clustering on Linked Data". In: *CoRR* abs/1708.09299 (2017). arXiv: 1708.09299.

[130] Markus Nentwig, Anika Groß, Maximilian Möller, and Erhard Rahm. "Distributed Holistic Clustering on Linked Data". In: *On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part II*, ed. by Hervé Panetto et al. Vol. 10574. Lecture Notes in Computer Science. Springer, 2017, pp. 371–382. ISBN: 978-3-319-69458-0. DOI: 10.1007/978-3-319-69459-7_25.

[131] Markus Nentwig, Anika Groß, and Erhard Rahm. "Holistic Entity Clustering for Linked Data". In: *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain.* Ed. by Carlotta Domeniconi et al. IEEE Computer Society, 2016, pp. 194–201. DOI: 10.1109/ICDMW.2016.0035.

[132] 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. DOI: 10.3233/SW-150210.

[133] Markus Nentwig and Erhard Rahm. "Incremental Clustering on Linked Data". In: *2018 IEEE International Conference on Data Mining Workshops, ICDM Workshops, Singapore, Singapore, November 17-20, 2018*, ed. by Hanghang Tong et al. IEEE, 2018, pp. 531–538. ISBN: 978-1-5386-9288-2. DOI: 10.1109/ICDMW.2018.00084.

[134] Markus Nentwig, Tommaso Soru, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. "LinkLion: A Link Repository for the Web of Data". In: *The Semantic Web: ESWC 2014 Satellite Events - ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers*, ed. by Valentina Presutti et al. Vol. 8798. Lecture Notes in Computer Science. Springer, 2014, pp. 439–443. ISBN: 978-3-319-11954-0. DOI: 10.1007/978-3-319-11955-7_63.

[135] Sebastian Neumaier, Jürgen Umbrich, and Axel Polleres. "Automated Quality Assessment of Metadata across Open Data Portals". In: *J. Data and Information Quality* 8.1 (2016), 2:1–2:29. DOI: 10.1145/2964909.

[136] Axel-Cyrille Ngonga Ngomo. "HELIOS - Execution Optimization for Link Discovery". In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, ed. by Peter Mika et al. Vol. 8796. Lecture Notes in Computer Science. Springer, 2014, pp. 17–32. ISBN: 978-3-319-11963-2. DOI: 10.1007/978-3-319-11964-9_2.

[137] Axel-Cyrille Ngonga Ngomo. "Link Discovery with Guaranteed Reduction Ratio in Affine Spaces with Minkowski Measures". In: *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I*, ed. by Philippe Cudré-Mauroux et al. Vol. 7649. Lecture Notes in Computer Science. Springer, 2012, pp. 378–393. ISBN: 978-3-642-35175-4. DOI: 10.1007/978-3-642-35176-1_24.

[138] Axel-Cyrille Ngonga Ngomo. "On Link Discovery using a Hybrid Approach". In: *J. Data Semantics* 1.4 (2012), pp. 203–217. DOI: 10.1007/s13740-012-0012-y.

[139] 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, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, ed. by Toby Walsh. IJCAI/AAAI, 2011, pp. 2312–2317. ISBN: 978-1-57735-516-8. DOI: 10.5591/978-1-57735-516-8/IJCAI11-385.

[140] Axel-Cyrille Ngonga Ngomo, Lars Kolb, Norman Heino, Michael Hartung, Sören Auer, and Erhard Rahm. "When to Reach for the Cloud: Using Parallel Hardware for Link Discovery". In: *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, ed. by Philipp Cimiano et al. Vol. 7882. Lecture Notes in Computer Science.

Springer, 2013, pp. 275–289. ISBN: 978-3-642-38287-1. DOI: 10.1007/978-3-642-38288-8_19.

[141] Axel-Cyrille Ngonga Ngomo, Jens Lehmann, Sören Auer, and Konrad Höffner. "RAVEN - active learning of link specifications". In: *Proceedings of the 6th International Workshop on Ontology Matching, Bonn, Germany, October 24, 2011*, ed. by Pavel Shvaiko et al. Vol. 814. CEUR Workshop Proceedings. CEUR-WS.org, 2011. URL: http://ceur-ws.org/Vol-814/om2011_Tpaper3.pdf.

[142] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. "EAGLE: Efficient Active Learning of Link Specifications Using Genetic Programming". In: *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, ed. by Elena Simperl et al. Vol. 7295. Lecture Notes in Computer Science. Springer, 2012, pp. 149–163. ISBN: 978-3-642-30283-1. DOI: 10.1007/978-3-642-30284-8_17.

[143] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. "Unsupervised learning of link specifications: deterministic vs. non-deterministic". In: *Proceedings of the 8th International Workshop on Ontology Matching co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 21, 2013.* Ed. by Pavel Shvaiko et al. Vol. 1111. CEUR Workshop Proceedings. CEUR-WS.org, 2013, pp. 25–36. URL: http://ceur-ws.org/Vol-1111/om2013_Tpaper3.pdf.

[144] Axel-Cyrille Ngonga Ngomo, Klaus Lyko, and Victor Christen. "COALA – Correlation-Aware Active Learning of Link Specifications". In: *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, ed. by Philipp Cimiano et al. Vol. 7882. Lecture Notes in Computer Science. Springer, 2013, pp. 442–456. ISBN: 978-3-642-38287-1. DOI: 10.1007/978-3-642-38288-8_30.

[145] Axel-Cyrille Ngonga Ngomo, Mohamed Ahmed Sherif, and Klaus Lyko. "Unsupervised Link Discovery through Knowledge Base Repair". In: *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, ed. by Valentina Presutti et al. Vol. 8465. Lecture Notes in Computer Science. Springer, 2014, pp. 380–394. ISBN: 978-3-319-07442-9. DOI: 10.1007/978-3-319-07443-6_26.

[146] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. "A Review of Relational Machine Learning for Knowledge Graphs". In: *Proceedings of the IEEE* 104.1 (2016), pp. 11–33. DOI: 10.1109/JPROC.2015.2483592.

[147] Andriy Nikolov, Mathieu d'Aquin, and Enrico Motta. "Unsupervised Learning of Link Discovery Configuration". In: *The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings*, ed. by Elena Simperl et al. Vol. 7295. Lecture Notes in Computer Science. Springer, 2012, pp. 119–133. ISBN: 978-3-642-30283-1. DOI: 10.1007/978-3-642-30284-8_15.

[148] Natalya Fridman Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. "Industry-scale knowledge graphs: lessons and challenges". In: *Commun. ACM* 62.8 (2019), pp. 36–43. DOI: 10.1145/3331166.

[149] Natalya Fridman Noy et al. "BioPortal: ontologies and integrated data resources at the click of a mouse". In: *Nucleic Acids Research* 37.Web-Server-Issue (2009), pp. 170–173. DOI: 10.1093/nar/gkp440.

[150] Open Knowledge Foundation. *Open Definition 2.1*. 2017. URL: https://opendefinition.org/od/2.1/en/.

[151] George Papadakis, Georgia Koutrika, Themis Palpanas, and Wolfgang Nejdl. "Meta-Blocking: Taking Entity Resolutionto the Next Level". In: *IEEE Trans. Knowl. Data Eng.* 26.8 (2014), pp. 1946–1960. DOI: 10.1109/TKDE.2013.54.

[152] George Papadakis, Jonathan Svirsky, Avigdor Gal, and Themis Palpanas. "Comparative Analysis of Approximate Blocking Techniques for Entity Resolution". In: *PVLDB* 9.9 (2016), pp. 684–695. DOI: 10.14778/2947618.2947624.

[153] Maria Pershina, Mohamed Yakout, and Kaushik Chakrabarti. "Holistic Entity Matching Across Knowledge Graphs". In: *2015 IEEE International Conference on Big Data, Big Data 2015, Santa Clara, CA, USA, October 29 - November 1, 2015.* IEEE, 2015, pp. 1585–1590. ISBN: 978-1-4799-9926-2. DOI: 10.1109/BigData.2015.7363924.

[154] Bastian Quilitz and Ulf Leser. "Querying Distributed RDF Data Sources with SPARQL". In: *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, ed. by Sean Bechhofer et al. Vol. 5021. Lecture Notes in Computer Science. Springer, 2008, pp. 524–538. ISBN: 978-3-540-68233-2. DOI: 10.1007/978-3-540-68234-9_39.

[155] Erhard Rahm. "The Case for Holistic Data Integration". In: *Advances in Databases and Information Systems - 20th East European Conference, ADBIS 2016, Prague, Czech Republic, August 28-31, 2016, Proceedings*, ed. by Jaroslav Pokorný et al.

Vol. 9809. Lecture Notes in Computer Science. Springer, 2016, pp. 11–27. ISBN: 978-3-319-44038-5. DOI: 10.1007/978-3-319-44039-2_2.

[156] Erhard Rahm. "Towards Large-Scale Schema and Ontology Matching". In: *Schema Matching and Mapping*, ed. by Zohra Bellahsene et al. Data-Centric Systems and Applications. Springer, 2011, pp. 3–27. ISBN: 978-3-642-16517-7. DOI: 10.1007/978-3-642-16518-4_1.

[157] Erhard Rahm and Philip A. Bernstein. "A survey of approaches to automatic schema matching". In: *VLDB J.* 10.4 (2001), pp. 334–350. DOI: 10.1007/s007780100057.

[158] Erhard Rahm and Hong Hai Do. "Data Cleaning: Problems and Current Approaches". In: *IEEE Data Eng. Bull.* 23.4 (2000), pp. 3–13. URL: http://dc-pubs.dbs.uni-leipzig.de/files/Rahm2000DataCleaningProblemsand.pdf.

[159] Thanyalak Rattanasawad, Marut Buranarach, Kanda Runapongsa Saikaew, and Thepchai Supnithi. "A Comparative Study of Rule-Based Inference Engines for the Semantic Web". In: *IEICE Transactions* 101-D.1 (2018), pp. 82–89. DOI: 10.1587/transinf.2017SWP0004.

[160] David Reinsel, John Gantz, and John Rydning. *Data Age 2025 - The Digitization of the World From Edge to Core*. Tech. rep. IDC, Nov. 2018. URL: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf.

[161] Christopher Rost, Andreas Thor, and Erhard Rahm. "Temporal Graph Analysis using Gradoop". In: *Datenbanksysteme für Business, Technologie und Web (BTW 2019), 18. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS), 4.-8. März 2019, Rostock, Germany, Workshopband*, ed. by Holger Meyer et al. Vol. P-290. LNI. Gesellschaft für Informatik, Bonn, 2019, pp. 109–118. ISBN: 978-3-88579-684-8. DOI: 10.18420/btw2019-ws-11.

[162] Alieh Saeedi, Markus Nentwig, Eric Peukert, and Erhard Rahm. "Scalable Matching and Clustering of Entities with FAMER". In: *CSIMQ* 16 (2018), pp. 61–83. DOI: 10.7250/csimq.2018-16.04.

[163] Alieh Saeedi, Eric Peukert, and Erhard Rahm. "Comparative Evaluation of Distributed Clustering Schemes for Multi-source Entity Resolution". In: *Advances in Databases and Information Systems - 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings*, ed. by Marite Kirikova et al.

Vol. 10509. Lecture Notes in Computer Science. Springer, 2017, pp. 278–293. ISBN: 978-3-319-66916-8. DOI: 10.1007/978-3-319-66917-5_19.

[164] Alieh Saeedi, Eric Peukert, and Erhard Rahm. "Using Link Features for Entity Clustering in Knowledge Graphs". In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, ed. by Aldo Gangemi et al. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 576–592. ISBN: 978-3-319-93416-7. DOI: 10.1007/978-3-319-93417-4_37.

[165] Muhammad Saleem and Axel-Cyrille Ngonga Ngomo. "HiBISCuS: Hypergraph-Based Source Selection for SPARQL Endpoint Federation". In: *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, ed. by Valentina Presutti et al. Vol. 8465. Lecture Notes in Computer Science. Springer, 2014, pp. 176–191. ISBN: 978-3-319-07442-9. DOI: 10.1007/978-3-319-07443-6_13.

[166] Emanuel Santos, Daniel Faria, Catia Pesquita, and Francisco M. Couto. "Ontology Alignment Repair through Modularization and Confidence-Based Heuristics". In: *PLOS ONE* 10.12 (Dec. 2016), pp. 1–19. DOI: 10.1371/journal.pone.0144807.

[167] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim. "Adoption of the Linked Data Best Practices in Different Topical Domains". In: *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, ed. by Peter Mika et al. Vol. 8796. Lecture Notes in Computer Science. Springer, 2014, pp. 245–260. ISBN: 978-3-319-11963-2. DOI: 10.1007/978-3-319-11964-9_16.

[168] Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer. "Question answering on interlinked data". In: *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, ed. by Daniel Schwabe et al. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 1145–1156. ISBN: 978-1-4503-2035-1. DOI: 10.1145/2488388.2488488.

[169] 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, February 4-9, 2017, San Francisco, California, USA*. Ed. by Satinder P. Singh et al. AAAI Press,

2017, pp. 175–181. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14199.

[170] Mohamed Ahmed Sherif, Mofeed M. Hassan, Tommaso Soru, Axel-Cyrille Ngonga Ngomo, and Jens Lehmann. "Lion's Den: feeding the LinkLion". In: *Proceedings of the 11th International Workshop on Ontology Matching co-located with the 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 18, 2016.* Ed. by Pavel Shvaiko et al. Vol. 1766. CEUR Workshop Proceedings. CEUR-WS.org, 2016, pp. 235–236. URL: http://ceur-ws.org/Vol-1766/om2016_poster5.pdf.

[171] Pavel Shvaiko and Jérôme Euzenat. "Ontology Matching: State of the Art and Future Challenges". In: *IEEE Trans. Knowl. Data Eng.* 25.1 (2013), pp. 158–176. DOI: 10.1109/TKDE.2011.253.

[172] Dilpreet Singh and Chandan K. Reddy. "A survey on platforms for big data analytics". In: *J. Big Data* 2 (2015), p. 8. DOI: 10.1186/s40537-014-0008-6.

[173] John Miles Smith et al. "Multibase: integrating heterogeneous distributed database systems". In: *American Federation of Information Processing Societies: 1981 National Computer Conference, 4-7 May 1981, Chicago, Illinois, USA.* Vol. 50. AFIPS Conference Proceedings. AFIPS Press, 1981, pp. 487–499. DOI: 10.1145/1500412.1500483.

[174] Sunghwan Sohn, Jean-Pierre A. Kocher, Christopher G. Chute, and Guergana K. Savova. "Drug side effect extraction from clinical narratives of psychiatry and psychology patients". In: *JAMIA* 18.Supplement (2011), pp. 144–149. DOI: 10.1136/amiajnl-2011-000351.

[175] René Speck and Axel-Cyrille Ngonga Ngomo. "Named Entity Recognition using FOX". In: *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014*, ed. by Matthew Horridge et al. Vol. 1272. CEUR Workshop Proceedings. CEUR-WS.org, 2014, pp. 85–88. URL: http://ceur-ws.org/Vol-1272/paper_70.pdf.

[176] Philip Stutz, Abraham Bernstein, and William W. Cohen. "Signal/Collect: Graph Algorithms for the (Semantic) Web". In: *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I*, ed. by Peter F. Patel-Schneider et al. Vol. 6496. Lecture Notes in Computer Science. Springer, 2010, pp. 764–780. ISBN: 978-3-642-17745-3. DOI: 10.1007/978-3-642-17746-0_48.

[177] Andreas Thalhammer, Steffen Thoma, Andreas Harth, and Rudi Studer. "Entity-centric Data Fusion on the Web". In: *Proceedings of the 28th ACM Conference on Hypertext and Social Media, HT 2017, Prague, Czech Republic, July 4-7, 2017*, ed. by Peter Dolog et al. ACM, 2017, pp. 25–34. ISBN: 978-1-4503-4708-2. DOI: 10.1145/3078714.3078717.

[178] The W3C SPARQL Working Group. *SPARQL 1.1 Overview.* W3C Recommendation. W3C, Mar. 2013. URL: https://www.w3.org/TR/sparql11-overview/.

[179] Andreas Thor and Erhard Rahm. "MOMA - A Mapping-based Object Matching System". In: *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings.* www.cidrdb.org, 2007, pp. 247–258. URL: http://cidrdb.org/cidr2007/papers/cidr07p27.pdf.

[180] André Valdestilhas, Tommaso Soru, and Axel-Cyrille Ngonga Ngomo. "CEDAL: time-efficient detection of erroneous links in large-scale link repositories". In: *Proceedings of the International Conference on Web Intelligence, Leipzig, Germany, August 23-26, 2017*, ed. by Amit P. Sheth et al. ACM, 2017, pp. 106–113. ISBN: 978-1-4503-4951-2. DOI: 10.1145/3106426.3106497.

[181] Leslie G. Valiant. "A Bridging Model for Parallel Computation". In: *Commun. ACM* 33.8 (1990), pp. 103–111. DOI: 10.1145/79173.79181.

[182] Dinusha Vatsalan, Ziad Sehili, Peter Christen, and Erhard Rahm. "Privacy-Preserving Record Linkage for Big Data: Current Approaches and Research Challenges". In: *Handbook of Big Data Technologies*, ed. by Albert Y. Zomaya et al. Springer, 2017, pp. 851–895. ISBN: 978-3-319-49339-8. DOI: 10.1007/978-3-319-49340-4_25.

[183] Jorge Veiga, Roberto R. Expósito, Xoán C. Pardo, Guillermo L. Taboada, and Juan Touriño. "Performance evaluation of big data frameworks for large-scale data analytics". In: *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, ed. by James Joshi et al. IEEE Computer Society, 2016, pp. 424–431. ISBN: 978-1-4673-9005-7. DOI: 10.1109/BigData.2016.7840633.

[184] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. "Discovering and Maintaining Links on the Web of Data". In: *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, ed. by Abraham Bernstein et al. Vol. 5823. Lecture Notes

in Computer Science. Springer, 2009, pp. 650–665. ISBN: 978-3-642-04929-3. DOI: 10.1007/978-3-642-04930-9_41.

[185] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. "Silk - A Link Discovery Framework for the Web of Data". In: *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009.* Ed. by Christian Bizer et al. Vol. 538. CEUR Workshop Proceedings. CEUR-WS.org, 2009. URL: http://ceur-ws.org/Vol-538/ldow2009_paper13.pdf.

[186] W3C. *Linking Open Data Community Project.* Aug. 2007. URL: http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData.

[187] W3C. *Semantic Web.* URL: https://www.w3.org/standards/semanticweb/.

[188] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition).* W3C Recommendation. W3C, Dec. 2012. URL: https://www.w3.org/TR/owl2-overview/.

[189] W3C Semantic Web Activity. *Latest layercake diagram.* URL: https://www.w3.org/2001/sw/.

[190] Michael J. Welch, Aamod Sane, and Chris Drome. "Fast and accurate incremental entity resolution relative to an entity knowledge base". In: *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, ed. by Xue-wen Chen et al. ACM, 2012, pp. 2667–2670. ISBN: 978-1-4503-1156-4. DOI: 10.1145/2396761.2398719.

[191] Stephan Wölger, Katharina Siorpaes, Tobias Bürger, Elena Simperl, Stefan Thaler, and Christian Hofer. *A Survey On Data Interlinking Methods.* Tech. rep. STI Innsbruck, Mar. 2011.

[192] Chuan Xiao, Wei Wang, Xuemin Lin, and Jeffrey Xu Yu. "Efficient similarity joins for near duplicate detection". In: *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, ed. by Jinpeng Huai et al. ACM, 2008, pp. 131–140. ISBN: 978-1-60558-085-2. DOI: 10.1145/1367497.1367516.

[193] Eric P. Xing et al. "Petuum: A New Platform for Distributed Machine Learning on Big Data". In: *IEEE Trans. Big Data* 1.2 (2015), pp. 49–67. DOI: 10.1109/TBDATA.2015.2472014.

[194] Matei Zaharia et al. "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing". In: *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012*, ed. by Steven D. Gribble et al. USENIX Association, 2012, pp. 15–28. URL: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia.

[195] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. "Quality assessment for Linked Data: A Survey". In: *Semantic Web* 7.1 (2016), pp. 63–93. DOI: 10.3233/SW-150175.

[196] Kai Zeng, Jiacheng Yang, Haixun Wang, Bin Shao, and Zhongyuan Wang. "A Distributed Graph Engine for Web Scale RDF Data". In: *PVLDB* 6.4 (2013), pp. 265–276. DOI: 10.14778/2535570.2488333.

[197] Paul C. Zikopoulos, Dirk deRoos, Krishnan Parasuraman, Thomas Deutsch, David Corrigan, and James Giles. *Harness the Power of Big Data – The IBM Big Data Platform*. McGraw-Hill, 2013.