Data Analysis and Integration of Spatio-temporal Events in DBpedia and YAGO

Master Thesis

submitted by
Daniel Ringler
Matriculation Number 1454860

submitted to the
Data and Web Science Group
Prof. Dr. Heiko Paulheim
Prof. Dr. Christian Bizer

University of Mannheim

April 2017

Abstract

This thesis outlines identity resolution strategies for integrating spatio-temporal event data extracted from the DBpedia and YAGO Knowledge Graphs. Therefore, different state-of-the-art blocking methods and learned matching rules are compared using different subsets of the data. In addition, a Web application that integrates the data dynamically and visualizes the fused events on a map is implemented. The main finding of this thesis is that blocking only becomes efficient when applying block- or comparison-refinement methods. For the analyzed data, taking all attributes for block building and removing entities from 50% of the largest blocks works best when regarding Pairs Completeness and Reduction Ratio. A simple matching rule that compares the stripped URI of the entities using a Levenshtein similarity measure with a high threshold of 0.96 outperforms all learned matching rules concerning the F-measure.

Keywords: Data Integration, Blocking, Matching Rules, Semantic Web

Contents

List of Figures

List of Tables

Acronyms

ACl Attribute Clustering.

ARCS Aggregate Reciprocal Comparisons Scheme.

ASCII American Standard Code for Information Interchange.

BlBu Block Building.

BlCl Block Cleaning.

BlFi Block Filtering.

CBS Common Blocks Scheme.

CEP Cardinality Edge Pruning.

CNP Cardinality Node Pruning.

CoCl Comparison Cleaning.

DTD Document Type Definition.

ECBS Enhanced Common Blocks Scheme.

EJS Enhanced Jaccard Scheme.

ETL Extract, Transform, Load.

FN False Negatives.

FP False Positives.

HTML HyperText Markup Language.

IDF Inverse Document Frequency.

IETF Internet Engineering Task Force.

IRI Internationalized Resource Identifier.

ISO International Organization for Standardization.

JS Jaccard Scheme.

KG Knowledge Graph.

LOD Linked Open Data.

MeBl Meta Blocking.

OTime Overhead Time.

OWL Web Ontology Language.

PC Pairs Completeness.

PQ Pairs Quality.

RDF Resource Description Framework.

RDFS RDF Vocabulary Description Language.

ReCNP Reciprocal Cardinality Node Pruning.

ReWNP Reciprocal Weight Node Pruning.

RFC Request for Comments.

RIF Rule Interchange Format.

RR Reduction Ratio.

RTime Resolution Time.

SGML Standard Generalized Markup Language.

SPARQL SPARQL Protocol and RDF Query Language.

SQL Structured Query Language.

StBl Standard Blocking.

SVM Support Vector Machine.

SWRL Semantic Web Rule Language.

TF Term Frequency.

TF/IDF Term Frequency / Inverse Document Frequency.

TN True Negatives.

TP True Positives.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

W3C World Wide Web Consortium.

WGS84 World Geodetic System 1984.

XML Extensible Markup Language.

XSD XML Schema Definition Language.

YAGO Yet Another Great Ontology.

Chapter 1

Introduction

This thesis analyzes different blocking methods and matching rules for the integration of event data from the DBpedia and YAGO Knowledge Graphs. The blocking methods and matching rules are evaluated using different effectiveness measures. Chapter 1.1 outlines the problem statement, while Chapter 1.2 presents the structure of this thesis.

1.1 Problem Statement

> "Despite all the advances, one of the main challenges for the consolidation of the Web of Data is data integration." [17]

With the increasing amount and size of datasets in the Semantic Web, data integration becomes "more important than ever" for integrating different overlapping data sources [45, p. 11]. At a TED conference entitled "The Next Web", Tim Berners-Lee stated that:

> "[T]he really important thing about data is the more things you have to connect together, the more powerful it is." [3]

Due to the still high number of missing links between duplicates in the Semantic Web, the identification of these duplicates that describe the same real-world entity is essential for interconnecting different datasets [60]. In order to decrease the quadratic complexity of the brute-force approach that compares every entity with all other entities, blocking algorithms try to reduce the amount of candidate pairs with the risk of missing a small number of potential duplicates. Therefore, this thesis will analyze different blocking methods for datasets that are extracted from the Semantic Web describing a specific domain. The blocking framework

that is used by Papadakis *et al.* to compare state-of-the-art blocking methods on six real-world datasets provides the basis for this thesis [55]. The Pairs Completeness, Pairs Quality, and Reduction Ratio effectiveness measures are used to compare the performance of the blocking algorithms to the brute-force approach.

For all candidate pairs a matching rule then decides whether the two entities are considered to describe the same real-world entity. The Silk framework is used to learn matching rules automatically. All matching rules are evaluated using the precision, recall, and F-measure on a labeled gold standard. Furthermore, a sign test on the True Positives provides further insights regarding the effectiveness of the different matching rules.

In addition to the challenges of data integration regarding the amount and size of datasets, the semi-structured and noisy data that is extracted from the Semantic Web arises problems [18, p. 1138]. Therefore, different data heterogeneity types have to be resolved before integrating the data.

The examined domain comprises events having, among others, spatial and temporal attributes. These diverse attributes make the integration interesting and challenging. The different attributes are analyzed regarding their datatypes and instances. In addition, attribute densities are calculated to find the attributes that provide the most significant information.

The aim of this thesis is to find an efficient blocking algorithm and a robust matching rule for the extracted datasets. Thus, the final results of this thesis can be used as starting point for integrating data sources from the Semantic Web. The runtime of the identity resolution step should be drastically reduced with an appropriate blocking method in comparison to the brute-force approach.

The integrated results are visualized in a Web application that uses the information in the coordinate properties to show the events in an interactive map. The Web application uses either the extracted offline data from the Knowledge Graphs or queries the public SPARQL endpoints from DBpedia and YAGO to receive the data dynamically.

1.2 Structure of this Thesis

First, Chapter 2 presents the theoretical framework. Hence, the basic principles of Linked Data and the Semantic Web are outlined. In addition, the four main data integration steps are defined. The main focus of this thesis is the identity resolution step for finding duplicates in the datasets. Therefore, the theory of blocking algorithms and matching rules are described explicitly. All blocking methods and matching rule concepts that are employed for the experiments are specified in detail. Furthermore, their effectiveness evaluation is defined. Chapter 3 outlines the

selection of the data sources from the Semantic Web, as well as the extraction and target schema mapping process. Chapter 4 describes the experimental setup and presents the results and their discussion. In Chapter 5, the results are summarized and remarks for potential future work are suggested.

Chapter 2

Theoretical Framework

This chapter describes the theoretical foundations of data integration in the Semantic Web. First, the technologies and principles of the Semantic Web are outlined in Chapter 2.1. Then, Chapter 2.2 defines the four main steps of the data integration process. Challenges and opportunities of integrating Web data are specified in Chapter 2.3. As matching rules and blocking algorithms are the main focus of this thesis, Chapters 2.4 and 2.5 provide detailed descriptions of these identity resolution steps. Finally, the key background work that is required for this thesis is discussed in Chapter 2.6.

2.1 Linked Data and the Semantic Web

The vision of the **Semantic Web**, or Web of Data, is to provide machine-readability on the global Web [7, p. 17]. Therefore, data items from different sources should be connected with typed links for creating an "interconnected data space" [31, pp. 4-5]. Unlike using untyped hyperlinks to connect HTML documents in the Web [2], the concept of creating meaningful links to find related data is defined as **Linked Data** [7, p. 2].

This chapter describes the technical components of the Semantic Web Stack in Chapter 2.1.1, the principles and current state of Linked Data in Chapter 2.1.2, and the challenges in the Semantic Web in Chapter 2.1.3.

2.1.1 The Semantic Web Stack

Figure 2.1 shows the Semantic Web Stack as defined by Tim Berners-Lee [4, p. 14]. All layers of the Semantic Web Stack will be described in the following in a bottom-up way giving definitions of the used terms.

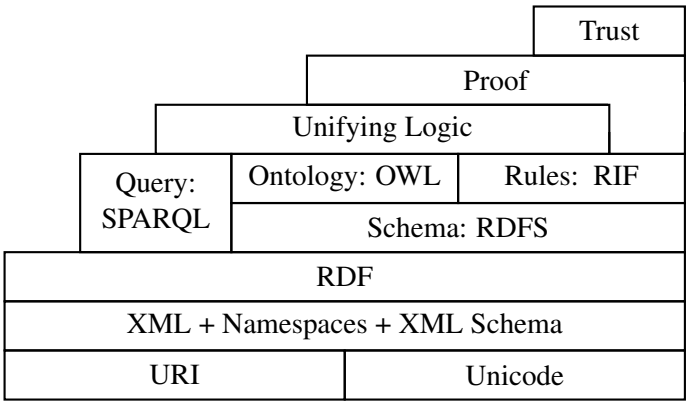


Figure 2.1: Semantic Web Stack based on [4, p. 14]

URI

While Uniform Resource Locators (URLs) provide a network location for accessing documents in the Web, Uniform Resource Identifiers (URIs) are a more generic unique identifier for describing any kind of resource in the world. Therefore, URLs are a subset of URIs [7, p. 3]. The URI syntax was standardized in RFC 3986 by IETF in 2005 [5].

Unicode

URIs are encoded using the ASCII character set [5]. In order to cover a wider set of characters exceeding ASCII, the Internationalized Resource Identifier (IRI) standard was introduced. An IRI is an URI using the Unicode character set as defined in ISO 10646. The IRIs syntax and standard for mapping IRIs to URIs is defined in RFC 3987. IRIs are a generalization of URIs as they allow a larger character set [23]. Hence, the two URI and Unicode layers in the Semantic Web Stack could be replaced with a single IRI layer.

XML, Namespaces, and XML Schema

Extensible Markup Language (**XML**) is the W3C standard format for data exchange and is based on the ISO 8879 definition for Standard Generalized Markup Language (SGML). XML uses arbitrarily definable tags and attributes to describe the different elements within the document [11].

Namespaces in XML allow to reference and reuse common vocabularies for element and attribute names. W3C defines a *vocabulary* as a *set of terms* that are "used to describe and represent an area of concern" [71]. Those *terms* define named

concepts, as well as relationships and constraints between those concepts [71]. Namespaces are defined using prefixes and are identified using URIs such as *xsd:* for referencing `http://www.w3.org/2001/XMLSchema#` [12]. Table 2.1 lists some examples of namespace prefixes, URIs, and the correspondent vocabularies.

Namespace prefix	Namespace URI	RDF vocabulary
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	Built-in RDF vocabulary
rdfs	`http://www.w3.org/2000/01/rdf-schema#`	RDFS vocabulary
xsd	`http://www.w3.org/2001/XMLSchema#`	RDF-compatible XSD types
owl	`http://www.w3.org/2002/07/owl#`	Web ontology standard

Table 2.1: Example Namespace Prefixes, URIs, and RDF vocabularies [39]

An **XML Schema** defines valid data types, elements, and attributes for an XML document. XML Schema is an extension of a Document Type Definition (DTD) as it allows more flexible declarations and is an XML document itself [66]. A DTD defines valid elements for XML documents [11].

RDF

The W3C standard format for defining typed relationships between single data items is Resource Description Framework (RDF). These links are similar to traditional hyperlinks in the Web but have two main differences:

1. RDF connects data items and not HTML documents.
2. RDF links have an explicit type [31, p. 4].

A single statement in RDF is written as **RDF triple** and has the *subject, predicate, object* order as illustrated in Figure 2.2.

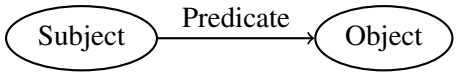


Figure 2.2: An RDF Triple [39]

W3C defines an **RDF graph** as "a set of RDF triples" [39]. The subjects and objects are the **nodes** of the RDF graph, while the predicates define the binary relation between the subjects and the objects (also called **property** or attribute). The number of edges that leave a node denotes the *outdegree* of the respective node, whereas the number of incoming edges specifies its *indegree*. The official W3C definition for the three components and types of an RDF triple are:

- **subject**: URI or blank node
- **predicate**: URI
- **object**: URI, literal or blank node [39]

Any URI or literal is called a **resource** or **entity** that "denotes something in the world" [39].

Blank nodes are used to state that a resource has a specific relationship "without explicitly naming it" [39]. Instead of a URI, a local identifier is used for the node [30].

"**Literals** are used for values such as strings, numbers, and dates". Every literal consists of a lexical form and a URI datatype such as *"1"^^xsd:integer* and *"text"^^xsd:string* or a language tag for strings such as *"english text"@en*. XML Schema Definition Language (**XSD**) defines those datatypes that are compatible with RDF [39].

RDF is based on the following principles:

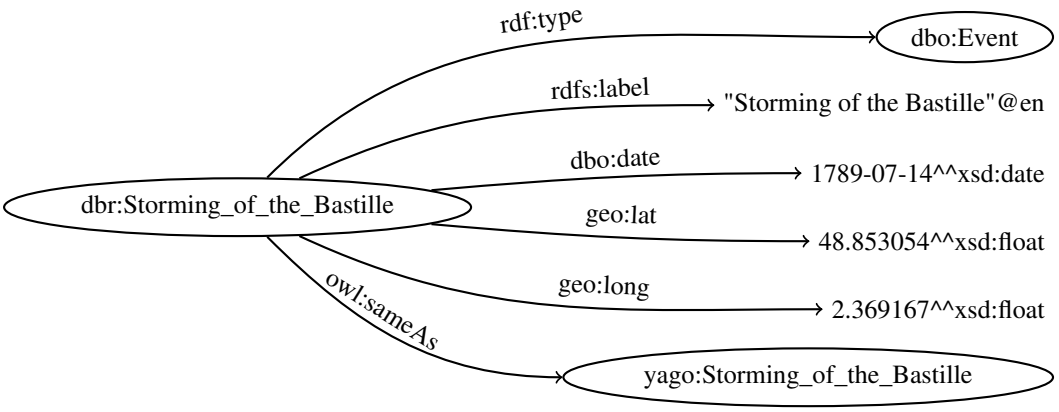
- **Anyone can make statements about any resource** [38].
- **Open World Assumption:** the provided information about resources might be incomplete [32, p. 119].
- **Non-Unique Naming Assumption:** different URIs can refer to the same real-world entity [32, p. 65].

Figure 2.3 shows an example RDF graph with resources, properties, literals and their datatypes or language tag with data that is used for the project of the thesis. The *dbr:Storming_of_the_Bastille* resource is an instance of the *dbo:Event* class in DBpedia. In addition, an extract of properties about the label, date, and coordinates are shown. The *owl:sameAs* property links the resource to the equivalent real-world entity *yago:Storming_of_the_Bastille* in YAGO.

RDFS

While RDF defines the basic data model to describe triples, RDF Vocabulary Description Language (**RDFS**) is a first basic vocabulary that is expressed in RDF format. A **vocabulary** is a "collection[] of classes and properties" [7, p. 4]. **Classes** are groups of resources and are resources themselves, whose members are called **instances** of this specific class. This relation is expressed using the *rdf:type* property as shown in Figure 2.3 [13].

Therefore, RDFS allows to describe "groups of related resources and the relationships between these resources" using pre-defined classes such as *rdfs:Resource*



Namespace Prefixes

dbo:	http://dbpedia.org/ontology/
dbr:	http://dbpedia.org/resource/
geo:	http://www.w3.org/2003/01/geo/wgs84_pos#
owl:	http://www.w3.org/2002/07/owl#
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
xsd:	http://www.w3.org/2001/XMLSchema#
yago:	http://yago-knowledge.org/resource/

Figure 2.3: Example of a simple RDF graph

or *rdfs:Class* and properties such as *rdf:type*, *rdfs:range*, *rdfs:domain*, *rdfs:subClassOf*, or *rdfs:subPropertyOf* [13].

SPARQL

SPARQL Protocol and RDF Query Language (**SPARQL**) is a query language for RDF with a similar syntax as SQL [43, pp. 307-308]. The basic principle of SPARQL is "graph pattern matching" [29]. A **basic graph pattern** is "a set of triple patterns" and **triple patterns** are RDF triples that may contain variables such as *?i rdf:type dbo:Event* for all instances of the dbo:Event class in DBpedia [29]. Appendices A.1 to A.3 show sample SPARQL queries that are used for the project.

OWL

Web Ontology Language (**OWL**) is the official ontology language by W3C. An **ontology** is a "formalized vocabular[y] of terms" for a certain domain and hereby defines the used terms and describes "their relationships with other terms in the ontology" [69]. Similar to RDFS, OWL is written in RDF but allows to express more complex knowledge and reasoning capabilities. The current version OWL 2 has three different **profiles** that are "sub-languages of OWL 2"[69]. These profiles with their main purpose are defined as:

- **OWL 2 EL** (Expressive Language): basic reasoning on ontologies with many properties and classes.
- **OWL 2 QL** (Query Language): fast query answering for "large volumes of instance data".
- **OWL 2 RL** (Rule Language): scalable reasoning with trading some expressiveness. [44]

RIF

Rule Interchange Format (**RIF**) is the W3C standard for rule exchange [10]. A popular rule language is Semantic Web Rule Language (**SWRL**) that allows to add Horn-like rules to OWL in order to increase the expressiveness [33].

Logic

The **unified logic** layer stands for the overall concept of writing rules and making implicit knowledge explicit through inference [56]. **Inference** is the automatic discovery of new relationships using reasoning techniques [70].

Proof and Trust

As RDF follows the principles of the Open World and Non-Unique-Naming assumptions, **proof** points out the importance of provenance in the Semantic Web. **Provenance** provides information about the origin of the data. **Trust** is based on this provenance and decides how accurate the provided information is. In addition, the trust layer points out the importance of agreements and the development of a technical infrastructure for establishing trustworthiness [43, p. 300] [61].

2.1.2 Principles of Linked Data

The four Linked Data principles as defined by Tim Berners-Lee are [2]:

1. Use URIs for naming things.
2. Use URLs "so that people can look up those names".
3. Provide useful information and use the standards such as RDF, RDFS, and SPARQL.
4. "Include links to other URIs."

Those four principles should lead to interconnected data in a "single global data space", the **Web of Data** [7, p. 2]. The goal of the W3C Linked Open Data (**LOD**) project is to identify open license data, convert it to RDF using the four Linked Data principles, and publish it on the Web [31, p. 30]. The LOD cloud diagram shows the distinct datasets in the Semantic Web and is illustrated in Figure 2.4.

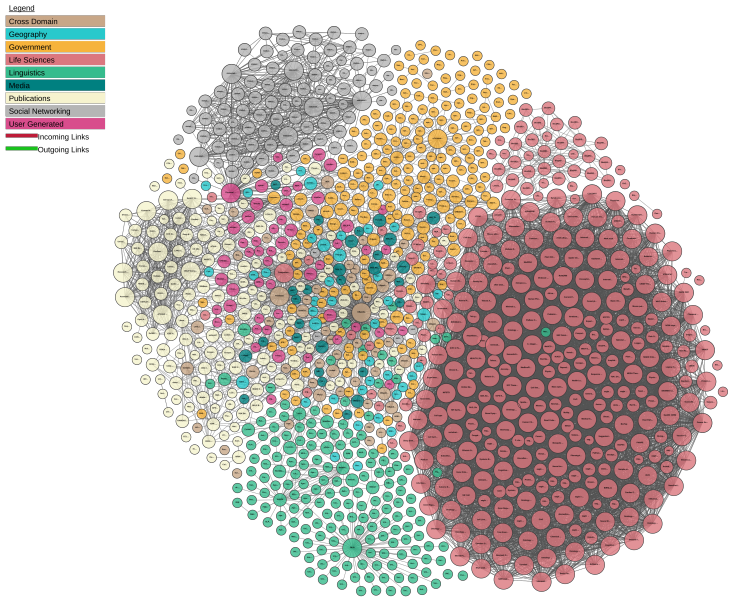


Figure 2.4: Linking Open Data Cloud Diagram 2017, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. `http://lod-cloud.net/`

Each node in the LOD cloud represents a dataset with a color indicating the main category. Edges connect datasets that share at least one link while the node

size illustrates the number of edges, indicating the overall connectivity of the dataset.

2.1.3 Challenges of the Semantic Web

With the rapidly growing amount of published datasets in the Semantic Web, new challenges regarding the structure, size, access, and quality of the data arise:

- **Data Structure:** different formats, schemata, data models, and encodings for the semi-structured data.
- **Data Size:** the increasing number of triples requires scalable solutions.
- **Data Access:** availability of public SPARQL endpoints.
- **Data Quality:** inconsistencies, duplicates, errors, and missing values [7, pp. 17-20] [31, p. 104] [43, p. 319].

Regarding these challenges and the diversity in the Semantic Web "the integration of datasets is one of the key points of LOD" [17, p. 1]. Therefore, the different steps of the data integration process have to resolve the challenges.

2.2 Data Integration

The goal of data integration is to "offer uniform access to a set of autonomous and heterogeneous data sources" [19, p. 6]. Due to the **autonomy** of data sources, different types of **heterogeneity** have to be resolved using the steps of the data integration process: data collection, schema matching and schema mapping, identity resolution, and data fusion. These four steps are illustrated in Figure 2.5 [43, p. 60]. The following subchapters will define each step in more detail.

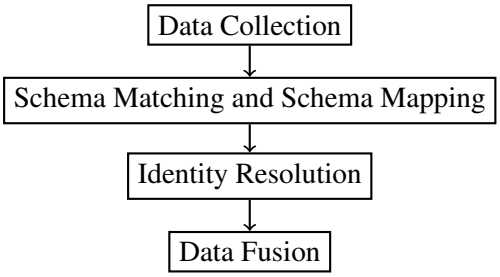
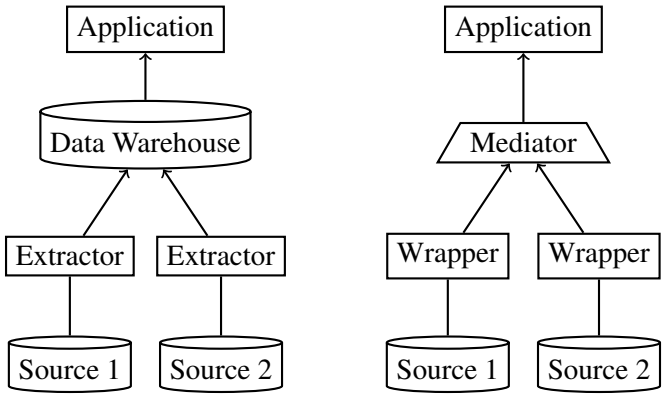


Figure 2.5: The Data Integration Process based on [31, p. 99]

2.2.1 Data Collection

The data collection step resolves the technical, syntactical, and data model heterogeneity so the data can be accessed and is represented in the same data model. **Technical heterogeneity** comprises all kinds of differences for accessing the data but not the data itself. This involves the communication protocol, the data exchange format, the query language, and potential restrictions for accessing the data [43, p. 62]. All encoding differences regarding character, number, and delimiter formats are referred to as **syntactical heterogeneity** [43, pp. 64-65]. **Data model heterogeneity** includes differences in the underlying data representation model such as relational, object-oriented, or XML [43, pp. 65-66].

Depending on the data storage location, different data integration architectures are used [43, p. 87]. Figure 2.6 shows the two basic data integration architectures.



(a) Materialized Integration (b) Virtual Integration

Figure 2.6: Data Integration Architectures based on [43, p. 87]

Extract, Transform, Load (ETL) tools are used for accessing the data in the **Materialized Integration** architecture. The extracted data is then combined in a central Data Warehouse [19, p. 11] [43, pp. 86-87].

Wrappers access the data in the **Virtual Integration** architecture. For each data source, a single **Wrapper** is responsible for accessing the underlying data. In contrast to the Materialized Integration, the data is not collected and stored at a central Data Warehouse. The data resides at the sources and is queried dynamically on demand. Afterwards, a **Mediator** translates the received data from all Wrappers to a mediated target schema. This **mediated schema** contains a logical view of the relevant data in the target schema and thus provides a single point of access to the application [19, pp. 10-11] [43, p. 97].

Hybrid Integration architectures combine the Materialized and Virtual Integration architectures and are used if the data is gathered from Data Warehouses as well as dynamically from Mediators [43, p. 91].

2.2.2 Schema Matching and Schema Mapping

For providing a single logical view of the data, the schema matching and schema mapping step resolves the structural and schema-related semantic heterogeneity by:

1. **Schema Matching:** Find correspondences of different schema attributes.

2. **Schema Mapping:** Translate source data to target schema [19, p. 121] [43, p. 123].

Resolving schema-related **semantic heterogeneity** aims to find attributes in the source schema that can be linked to the target schema. These links are called **correspondences** and might involve a transformation of the mapped values. **Structural heterogeneity** comprises these transformations due to differences in the source and target schema [43, p. 123]. Figure 2.7 illustrates the schema matching and schema mapping main tasks of finding correspondences and creating queries for the transformation of the data.

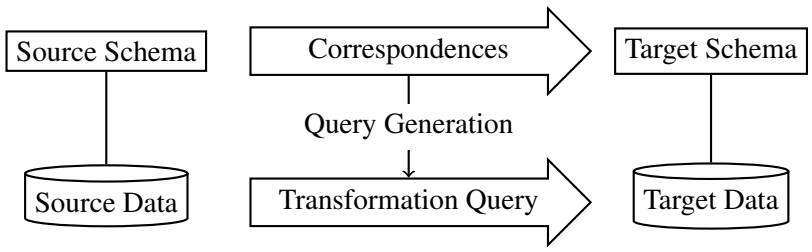


Figure 2.7: Schema Matching and Schema Mapping Overview based on [43, p. 125]

The correspondences are usually defined by domain experts but there is a lot of research for finding correspondences automatically or at least reduce the required effort of defining those links by hand [27, pp. 1-2]. After linking the source schema attributes to the target schema attributes with correspondences, the schema mapping is executed by the ETL tools or the Mediator when bringing this step into context of Figure 2.6.

2.2.3 Identity Resolution

Identity resolution tries to detect duplicates, thereby resolving **semantic heterogeneity** on the object level. **Duplicates** are records that describe the same real-world entity [19, p. 173] [45, p. 1].

The first well-known article about identity resolution, "Automatic Linkage of Vital Records" by Newcombe *et al.* [46], was published in 1959 and describes the research problem and proposed first matching solutions [19, p. 205]. Based on this work, Fellegri and Sunter developed the first mathematical model to provide a theoretical framework for identity resolution in their "A Theory of Record Linkage" article from 1969 [26] [43, p. 366].

Contrary to the aim of resolving ambiguity, the identity resolution concept has many synonyms such as duplicate detection, record linkage, object/data/instance/entity matching or identification, deduplication, and entity resolution [45, p. 8]. This thesis will stick to the term identity resolution.

Figure 2.8 illustrates the basic two-step process of identity resolution. First, an algorithm tries to find candidate pairs of records among the input data. This step tries to reduce the quadratic complexity of comparing each record with all other records and is known as **blocking**. Afterwards, a **matching rule** calculates a similarity score sim_{MR} for the candidate pair. This similarity indicates the probability of a match and is compared to a predefined threshold θ for deciding whether these two records are considered to be duplicates [45, pp. 2-3]. As identity resolution is the main focus of the thesis, these two steps are described in detail in Chapters 2.4 and 2.5.

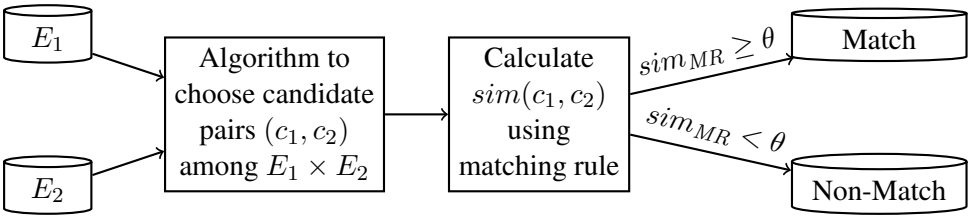


Figure 2.8: The Identity Resolution Process based on [45, p. 3]

2.2.4 Data Fusion

The goal of the data fusion step is to combine the data values of the found duplicates so that each real-world entity is only represented once. Depending on the existence of values for each attribute of the duplicates, this leads to one of the following data fusion cases:

- **Data equality:** identical data values for the same attribute.
- **Data enrichment**: complementary attributes enrich the fused entity.
- **Data conflict**: diverse values for the same attribute have to be resolved [43, pp. 343-344].

The fused entity is a combination of the found duplicates and combines all data values depending on the data fusion strategy, thereby reducing redundancy and resolving contradictions. Data conflicts can arise due to e.g. misspelling, incorrect calculation, timeliness, and different interpretations [20, p. 107]. Table 2.2 shows these cases for two example entities from DBpedia and YAGO that describe the same real-world entity. The English labels are identical (data equality), the dates have diverse values (data conflict) and only one entity has a value for the place attribute (data enrichment).

URI	English Label	Date	Place
dbr:64th_Berlin_International_Film_Festival	64th Berlin International Film Festival	2014-02-16	Berlin, Germany
yago:64th_Berlin_International_Film_Festival	64th Berlin International Film Festival	2014-##-##	-

Table 2.2: Data Fusion Example

While the data equality and data enrichment cases can be resolved easily, data conflicts require more advanced resolution strategies in order to decide which value is assumed to be correct. Therefore, the quality of the data is addressed using different heuristics including **provenance** information and other measures for estimating its "fitness for use" [19, p. 359] [43, pp. 353-354].

2.3 Data Integration on the Web

"Since the value of data explodes when it can be linked and fused with other data" the integration of large datasets from the Semantic Web becomes crucial for reaching the promise to "make valuable, data-driven decisions to alter all aspects of society" [20, p. 1]. This vision can be realized if the challenges that arise with the increasing amount and size of datasets in the Semantic Web are addressed properly [45, pp. 10-11]. These challenges and opportunities are outlined in the following subchapters.

2.3.1 Challenges

The main challenges of integrating big data sources from the Semantic Web are summarized using the popular "V" dimensions [20, pp. 11-13]:

- **Volume:** size and amount of data sources.
- **Velocity:** dynamic content and fast update rate.
- **Variety:** diverse and heterogeneous structure.
- **Veracity:** different quality, coverage, accuracy, and timeliness.

For creating an easily accessible and linked Semantic Web, the data integration process has to solve these challenges of the "V" dimensions.

2.3.2 Opportunities

The challenges of integrating large data sources from the Semantic Web are addressed with the following prospects [20, pp. 27-29]:

- **Big Data Platforms**
- **Long Data:** collection of data values over time.
- **Data Redundancy:** overlapping data from multiple sources.

The development of scalable big data platforms that parallelize all steps of the data integration process addresses the volume challenge. A current example is the MapReduce programming model using the Apache Hadoop framework [20, p. 29].

Since data values can evolve over time and, consequently, the truth can change, existing data fusion techniques have to be extended. Long data addresses the velocity challenge with the goal to find the current true value or correct values for a specific period [20, p. 133].

Data redundancy addresses the variety and veracity challenges. Overlapping data values from different sources indicate equivalent attributes and thus support finding correspondences for schema matching and creating a target schema thereby solving variety [20, p. 28]. In addition, the "collective wisdom from the sources" supports the truth and error discovery [20, p. 109]. The probability to identify the correct values for data fusion rises with the amount of diverse sources [20, p. 28].

2.4 Matching Rules

This subchapter introduces the matching rules that are applied for calculating a similarity score of two entities during the identity resolution step. Given a predefined threshold θ and two entities x and y from two entity sets X and Y so that $x \in X$ and $y \in Y$, the similarity score of a matching rule sim_{MR} determines whether a pair (x, y) is assumed to describe the same real-world entity. The formal definition is [19, p. 173] [45, p. 23]:

$$classify(x, y) = \begin{cases} x \text{ and } y \text{ are duplicates} & \text{if } sim_{MR}(x, y) \geq \theta \\ x \text{ and } y \text{ are non-duplicates} & \text{otherwise} \end{cases} \tag{2.1}$$

where $sim_{MR}(x, y) \in [0, 1]$ and $\theta \in [0, 1]$.

Matching rules can be a single or a linearly weighted combination of multiple similarity measures $sim(x, y)$. A linearly weighted matching rule for n similarity measures with predefined weights α is defined as [19, p. 175]:

$$sim_{MR}(x, y) = \sum_{i=1}^{n} \alpha_i \times sim_i(x, y) \tag{2.2}$$

where $sim_i(x, y) \in [0, 1]$, $\alpha_i \in [0, 1]$, $\sum_{i=1}^{n} \alpha_i = 1$, and $i = 1, ..., n$.

When using a distance measure $dist(x, y)$ instead of a similarity measure, the similarity score can be calculated using [45, p. 23]:

$$sim(x, y) = 1 - dist(x, y) \tag{2.3}$$

where $dist(x, y) \in [0, 1]$.

The following chapters will describe basic string (2.4.1) and numeric (2.4.2) similarity measures as well as evaluation metrics for matching rules (2.4.3).

2.4.1 String Similarity Measures

String similarity measures take two strings as input and return a similarity score between 0 and 1. The higher the score of a similarity measure, the more likely the two strings match. String matching faces two major challenges [19, p. 95]:

- **Accuracy:** misspelling and typos, different formats, abbreviations, and shuffled word order make it difficult to compare two strings.
- **Scalability:** computational cost of calculating the similarity measure.

This chapter will present different categories of string similarity measures and define the most common ones.

Edit-based Similarity

Edit-based similarity measures calculate the cost of transforming one string to the other. These transformations or "edit operations" include insertion, deletion, replacement and character swaps [19, p. 96] [45, p. 30].

The **Levenshtein** distance $dist_{Levenshtein}(x, y)$ calculates the minimum insertion, deletion, and replacement operations of transforming a string x to a string y and can be computed using dynamic programming with the following recurrence equation [19, p. 97]:

$$d(i,j) = min \begin{cases} d(i-1,j-1) & \text{if } x_i = y_i \text{ //copy} \\ d(i-1,j-1)+1 & \text{if } x_i \neq y_i \text{ //substitute} \\ d(i-1,j)+1 & \text{//delete } x_i \\ d(i,j-1)+1 & \text{//insert } y_i \end{cases} \tag{2.4}$$

where x_i and y_j are prefixes of the strings $x = x_1x_2...x_n$ and $y = y_1y_2...y_m$. The length of the substrings is determined by i and j. Therefore, $d(i, j)$ is the minimum transformation cost for the strings x_i and y_j where i and j indicate the length of the prefixes of the strings [43, p. 335].

The conversion of the Levenshtein distance $dist_{Levenshtein}(x, y)$ into a similarity function $sim_{Levenshtein}(x, y)$ is calculated with scaling the edit operations to the length of the longest string [19, p. 97]:

$$sim_{Levenshtein}(x,y) = 1 - \frac{dist_{Levenshtein}(x,y)}{max(|x|,|y|)} \tag{2.5}$$

As an example, the edit distance for the strings *Marla* and *Maria* is 1 for the substitution transformation of the characters *l* and *i*. The two strings *Marla* and *Mira* have an edit distance of 2 for the substitution of *a* with *i* at the second position and the deletion of the *l* character. The similarity scores are calculated as:

$$sim_{Levenshtein}(Marla, Maria) = 1 - \tfrac{1}{max(5,5)} = 0.8$$

$$sim_{Levenshtein}(Marla, Mira) = 1 - \tfrac{2}{max(5,5)} = 0.6$$

The computational cost for computing the Levenshtein distance is $O(|x||y|)$ and is approximated as quadratic due to the often similar length of strings x and y [19, p. 98]. Other edit-based string measures such as Needleman-Wunch, Affine Gap, and Smith-Waterman extend or generalize the Levenshtein distance with other scoring models and gap handling strategies [19, pp. 98-103].

The **Jaro** similarity measure compares two strings x and y with finding common characters and calculating the number of transpositions. Common characters are defined as *identical characters that are close to each other* [19, p. 103]:

$$|i - j| \leq \frac{min(|x|, |y|)}{2} \tag{2.6}$$

where $x_i = y_j$. The number of common characters in x and y is defined as c. With Equation 2.6, two identical characters are determined as close if their position does not "differ by more than half of the length of the shorter string" [45, p. 32].

Those common character strings are then used to count the number of transpositions t. There is a transposition if the ith common character of x does not match the ith common character of y. The Jaro similarity score sim_{Jaro} is then calculated with [19, p. 103]:

$$sim_{Jaro}(x, y) = \frac{1}{3} \times \left(\frac{c}{|x|} + \frac{c}{|y|} + \frac{c - \frac{t}{2}}{c} \right) \tag{2.7}$$

The Jaro similarity score was developed for comparing short strings such as names with minor spelling errors [45, p. 32].

The strings *Marla* and *Maria* have 4 common characters in the same order and at the same position. Thus, the number of transpositions equals 0. The Jaro similarity score is calculated as:

$$sim_{Jaro}(Marla, Maria) = \tfrac{1}{3} \times \left(\tfrac{4}{5} + \tfrac{4}{5} + \tfrac{4-0}{4} \right) \approx 0.87$$

Another example having transpositions are the two strings *Marla* and *Mira* that have the 3 common characters *M*, *a*, and *r*. Using equation 2.6, the first *a* of *Marla* at position 2 is within the common character distance of 2 with the *a* at the fourth position in *Mira*. Hence, the subsequences of common characters that are compared are *Mar* and *Mra*, which results in two transpositions for the characters *a* and *r*. The Jaro similarity score is calculated as:

$$sim_{Jaro}(Marla, Mira) = \tfrac{1}{3} \times \left(\tfrac{3}{5} + \tfrac{3}{4} + \tfrac{3-1}{3} \right) \approx 0.67$$

The computational costs for calculating the Jaro similarity score is $O(|x||y|)$ and therefore quadratic [19, p. 104].

The **JaroWinkler** similarity measure $sim_{JaroWinkler}$ extends the Jaro score, with putting a weight w on a common prefix [19, p. 104]:

$$sim_{JaroWinkler}(x, y) = (1 - |p| \times w) \times sim_{Jaro}(x, y) + |p| \times w \tag{2.8}$$

where p is the common prefix and w is the scaling factor. This weighting of a common prefix follows the idea that names can contain additional suffixes such as middle names that may have different abbreviations [45, p. 33]. The JaroWinkler similarity score for the strings *Marla* and *Maria* as well as *Marla* and *Mira* with a scaling factor of $w = 0.1$ is calculated as:

$$sim_{JaroWinkler}(Marla, Maria) = (1 - 3 \times 0.1) \times 0.87 + 3 \times 0.1 = 0.91$$

$$sim_{JaroWinkler}(Marla, Mira) = (1 - 1 \times 0.1) \times 0.67 + 1 \times 0.1 = 0.71$$

Marla and *Maria* have the common prefix *Mar* with the length of 3, whereas *Marla* and *Mira* only share the first character for the prefix. The approximated results of the Jaro similarities are used for the calculations.

Token-based Similarity

Token-based similarity measures split every string into a set of *tokens* and then compare these sets of tokens with each other. A common tokenization function uses whitespace characters as delimiter for splitting the strings. Other functions might extract *q-grams* from the strings where *q* defines the length of the token [19, p. 104] [45, p. 24].

The **Jaccard** similarity score measures the overlap of shared tokens in comparison to the overall number of tokens for two sets of tokens B_x and B_y [19, p. 104]:

$$sim_{Jaccard}(x, y) = \frac{|B_x \cap B_y|}{|B_x \cup B_y|} \tag{2.9}$$

Using a basic tokenization function on whitespace characters, the Jaccard score for the strings *Marla Singer* and *Singer Marla* is 1.0 whereas the Jaccard score for the strings *Marla Singer* and *Samuel Weller Singer* is 0.25.

The Term Frequency / Inverse Document Frequency (**TF/IDF**) similarity measure is used in information retrieval to "find documents that are relevant to keyword queries" [19, p. 105]. The underlying concept is that two strings are assumed to be "similar if they share distinguishing terms" [19, p. 105]. The Term Frequency (TF) and Inverse Document Frequency (IDF) scores are calculated as [19, p. 105] [43, p. 339]:

- For each term t and each document d: compute $tf(t, d)$ as the total number that t occurs in d.
- For each term t: compute $idf(t)$ as fraction of the total number of documents ($|d|$) to the number of documents that contain the term t.

A high idf value indicates a more distinguishing term t for the document it occurs in. Afterwards, each document d is converted to a feature vector v_d. For each term t, v_d has a feature $v_d(t)$ which is a "function of the TF and IDF scores" such as [19, pp. 105-106] [45, p. 27]:

$$v_d(t) = \log(tf(t,d) + 1) \times \log(idf(t)) \tag{2.10}$$

Feature vectors v_d that are close to each other indicate similar documents d. The TF/IDF score of two strings x and y can then be computed as cosine similarity of the feature vectors for x and y [19, p. 106]:

$$sim_{TF/IDF}(x,y) = \frac{\sum_{t \in T} v_x(t) \times v_y(t)}{\sqrt{\sum_{t \in T} v_x(t)^2} \times \sqrt{\sum_{t \in T} v_y(t)^2}} \tag{2.11}$$

where T is the set of all terms t. The feature vector v_d can be normalized to length 1 with [19, p. 106]:

$$v_d(t) = \frac{v_d(t)}{\sqrt{\sum_{t \in T} v_d(t)^2}} \tag{2.12}$$

Hybrid Similarity Measures

Similarity measures that combine the edit- and token-based methods are called hybrid similarity measures. Those similarity measures *soften* the exact matching requirements of the token-based methods and use edit-based similarity measures for finding *close* terms. Popular examples for hybrid similarity measures are the **Generalized Jaccard**, **Soft TF/IDF**, and **Monge-Elkan** methods [19, pp. 106-109].

Phonetic-based Similarity

Phonetic-based similarity measures such as **Soundex** are based on the sound of words. Thereby, the first character of the string is kept and all non-vowel characters of the strings are replaced with up to three digits that stand for similar sounds. Zeros are used for filling if the strings have less than three non-vowel characters. Those four character codes are then compared with each other to find matches [19, pp. 109-110] [45, p. 39]. The string *Marla* becomes *M640* and *Maria* becomes *M600* where the 6 stands for the *r* character and the 4 represents the *l* character. Phonetic-based similarity measures are mostly used for matching surnames that sound similar but are potentially spelled differently by mistake [19, p. 109].

2.4.2 Numeric Similarity Measures

All presented string matching methods do not work well for numerical data values: e.g. the edit distances for comparing the year date 2017 with 2016 and 1017 are 1 for both cases. Nevertheless, the years 2017 and 2016 are much closer when regarding the numeric difference. Therefore, different numeric similarity measures have to be used when comparing timestamps, dates, or coordinates. It is important to define the meaning of the difference in numbers for diverse domains: depending on the context a difference of e.g. a second can be large or little. The numeric similarity function and the meaning of the difference is defined explicitly for the specific use case [45, p. 39].

2.4.3 Matching Rule Evaluation

Given a set of candidate pairs from two input entity collections, a matching rule declares each pair as duplicate or non-duplicate. Depending on this declaration and its correctness, there are four different types of pairs [43, pp. 331-332]:

1. **True positives (TP):** correctly declared duplicates.

2. **False positives (FP):** incorrectly declared duplicates.

3. **True negatives (TN):** correctly declared non-duplicates.

4. **False negatives (FN):** incorrectly declared non-duplicates.

Table 2.3 shows these four types for detecting duplicates. The goal of a matching rule is to maximize the intersection of the true and declared duplicates. For evaluating the success of a matching rule there are different success measures that can be calculated using the cardinalities of the sets corresponding to the four types [45, p. 61].

	True Duplicate	**True Non-Duplicate**
Declared Duplicate	True Positives	False Positives
Declared Non-Duplicate	False Negatives	True Negatives

Table 2.3: Error Types in Identity Resolution based on [43, p. 332]

Precision

Precision "measures the ratio of correctly identified duplicates compared to all declared duplicates" [45, p. 61]:

$$\mathrm{p} = \frac{\mid \mathrm{TP} \mid}{\mid \mathrm{TP} \mid + \mid \mathrm{FP} \mid} = \frac{\mid \mathrm{TP} \mid}{\mid \text{declared duplicates} \mid} \tag{2.13}$$

A high precision value indicates that the matching rule is confident when declaring two entries as duplicates [43, p. 332].

Recall

Recall "measures the ratio of correctly identified duplicates compared to all true duplicates" [45, p. 61]:

$$\mathrm{r} = \frac{\mid \mathrm{TP} \mid}{\mid \mathrm{TP} \mid + \mid \mathrm{FN} \mid} = \frac{\mid \mathrm{TP} \mid}{\mid \text{true duplicates} \mid} \tag{2.14}$$

A high recall value indicates that many of the duplicates are found by the matching rule. Nevertheless, as the number of False Positives is not involved, declaring many non-duplicates as duplicates and will not affect the recall score [43, p. 332].

F-measure

With the diverging goals of precision and recall, the F-measure is a combination of the two and calculates the harmonic mean [43, p. 333] [45, p. 62]:

$$\mathrm{f} = \frac{2 \times \mathrm{r} \times \mathrm{p}}{\mathrm{r} + \mathrm{p}} \tag{2.15}$$

In order to achieve high values for the F-measure, "both precision and recall must be high" [45, p. 62].

Gold Standard

For evaluating the success of the matching rule, the knowledge of all duplicates within the dataset is assumed. A so-called gold standard labels entity pairs as duplicates or non-duplicates. If this gold standard is not known before, it has to be created manually. As the complexity of labeling all entities from the two datasets E_1 and E_2 is quadratic ($|E_1| \times |E_2|$), smaller subsets of the data are used or artificial methods are applied to create this gold standard automatically [45, p. 65].

Sign Test

The **sign test** is a quantitative measurement for the significance of differences. The data consists of n' bivariate pairs $(X_1, Y_1), (X_2, Y_2), ..., (X_{n'}, Y_{n'})$ [16, p. 122] [63, p. 68]. Each pair (X_i, Y_i) of the random variables X and Y is categorized as one of three classes [16, p. 123]:

- ***Plus*** **(+)**, if $X_i < Y_i$.
- ***Minus*** **(−)**, if $X_i > Y_i$.
- ***Tie*** **(0)**, if $X_i = Y_i$.

The sign test assumes that the random pairs are mutually independent and each pair is internally consistent and has an ordinal scale so that it can be classified as plus, minus, or tie. Using P to denote the probability, the hypotheses for the two-tailed test are [16, p. 123]:

$$\begin{aligned} H_0 &: P(+) = P(-) \\ H_1 &: P(+) \neq P(-) \end{aligned} \tag{2.16}$$

H_0 expects that the number of pairs for the *plus* and *minus* classes are the same and H_1 states the contradicting statement indicating that there is a significant change for one of the classes. The number of *plus* pairs is denoted as n_+ while n_- represents the number of *minus* pairs. n stands for the sum of *plus* and *minus* pairs without the ties [16, pp. 123-124]:

$$n = n_+ + n_- \tag{2.17}$$

The approximate desired level of significance is denoted as α. Depending on the amount of n, two different calculations for the critical value t have to be used [16, p. 124]:

- $n \leq 20$: Select the value nearest to $\alpha/2$ in the bionominal distribution table for $p = 0.5$ and the respective value for n. The y value of the table is assigned for t.
- $n > 20$: t is approximated using Equation 2.18.

$$t = \tfrac{1}{2}\left(n + w_{\alpha/2}\sqrt{n}\right) \tag{2.18}$$

where $w_{\alpha/2}$ is obtained from the normal distribution table. For $\alpha = 0.05$, Equation 2.18 can be approximated as $t = \frac{n}{2} - \sqrt{n}$ with $w_{\alpha/2} = -1.96$. The hypothesis H_0 is rejected if $n_+ \leq t$ or $n_+ \geq n - t$ [16, p. 124].

2.5 Blocking

As illustrated in Figure 2.8, the identity resolution step takes two entity collections as input, creates a set of candidate pairs, and passes those pairs to the matching rule which then decides whether the pair is assumed to be a duplicate. The set of candidate pairs has quadratic complexity as every entity of the first collection E_1 has to be compared with every entity of the second collection E_2: $|E_1| \times |E_2|$ [25, p. 11] [45, p. 16].

There are two different types of input entity collections [47, p. 12]:

- **Clean:** duplicate-free.
- **Dirty:** contain duplicates.

For finding all duplicates in a single dirty entity collection E, the complexity of comparing all n entities e with each other is n^2. This number of required comparisons can be reduced without missing duplicates by assuming [45, p. 16]:

- **Reflexive similarity:** $sim(e_1, e_1) = 1$, entities do not have to be compared with themselves. This corresponds to the blue diagonal line in the matrix in Figure 2.9.
- **Symmetric similarity:** $sim(e_1, e_2) = sim(e_2, e_1)$, entities only have to be compared once. This refers to the white squares on the bottom left of the matrix in Figure 2.9.

Those assumptions reduce the complexity to $\frac{(n^2-n)}{2}$. Figure 2.9 shows this example for 20 entities, where the number of comparisons can be reduced from 400 $\left(20^2\right)$ to 190 $\left(\frac{20^2-20}{2}\right)$ [43, p. 333] [45, p. 16].

The orange blocks illustrate all comparisons, blue blocks show the unnecessary symmetric comparisons, and white blocks indicate the saved reflexive pairs. Although many comparisons can be saved, the complexity is still quadratic. Hence, blocking algorithms trade some of the effectiveness in order to further reduce the number of candidate pairs. In this case, effectiveness describes the number of detected duplicates.

For scaling the identity resolution step to large datasets, blocking groups entities into blocks and only compares those entities that are in the same block. Therefore, the complexity is still quadratic but only within blocks and not on the size of the whole dataset.

Figure 2.10 illustrates the same example of 20 entities of a single dirty entity collection with a sample blocking key that reduces the 190 comparisons to 47 [45, pp. 43-44]. Each color indicates a different block.

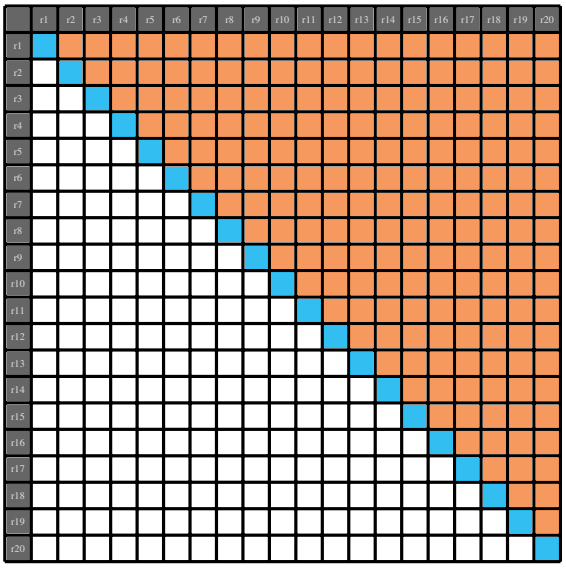


Figure 2.9: Duplicate Candidates Matrix based on [45, p. 17]

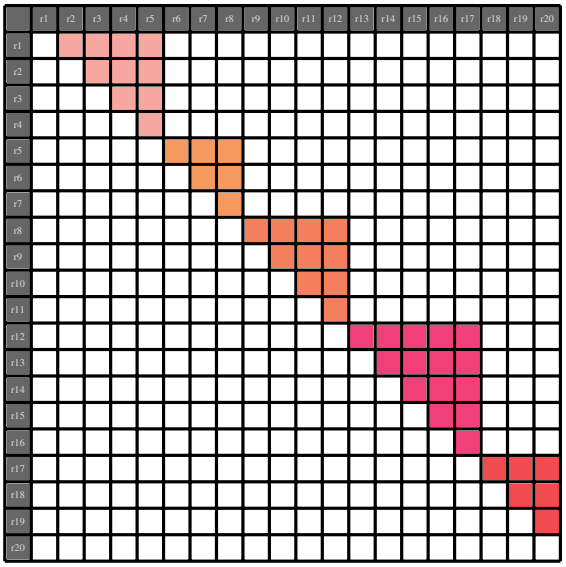


Figure 2.10: Duplicate Candidates Matrix with Blocking based on [45, p. 44]

2.5.1 The Sub-tasks of Blocking

Blocking consists of three consecutive sub-tasks as illustrated in Figure 2.11: Block Building (BlBu), Block Cleaning (BlCl), and Comparison Cleaning (CoCl), where BlCl and CoCl are optional.

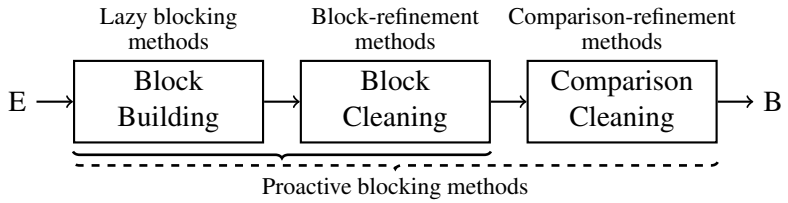


Figure 2.11: The Sub-tasks of Blocking based on [55, p. 3]

BlBu takes one dirty or several clean entity collections E as input, creates blocks based on a blocking key, and assigns entities to zero, one, or multiple blocks. The parameters of BlBu define the blocking key and the key use. The blocking key definition determines how the keys are extracted from the attribute values. The key use specifies how the support of the blocking keys is measured and might discard keys that do not fulfill a certain threshold. Hence, some entities might not be assigned to any block at all. The final block collection B consisting of all blocks with the assigned entities is passed to the next step.

BlCl receives this block collection and tries to remove unnecessary comparisons, which can be:

- **Redundant:** "repeated comparisons across different blocks" [55, p. 3].
- **Superfluous:** "involve non-matching entities" [55, p. 3].

As superfluous comparisons can only be detected when matching the entities, BlCl trades some recall for efficiency. The parameters of BlCl either operate on:

- **Block-level:** constraints that affect "individual blocks (e.g. maximum entities per block)" [55, p. 3].
- **Entity-level:** constraints that affect "individual entities (e.g. maximum blocks per entity)" [55, p. 3].

Therefore, the output of BlCl is a refined block collection having fewer redundant and superfluous comparisons.

CoCl takes this refined block collection and creates a set of candidate pairs. In addition, it trades some recall while trying to avoid further unnecessary comparisons. "Instead of targeting entire blocks [...], it operates at a finer level of

granularity, targeting individual comparisons" [55, p. 3]. The parameters of CoCl either address redundant or superfluous comparisons.

2.5.2 Blocking Methods

As illustrated in Figure 2.11, there are four different categories of blocking methods depending on the addressed sub-tasks [55, p. 3]:

- **Lazy blocking methods:** BlBu only.
- **Block-refinement methods:** BlCl only.
- **Comparison-refinement methods:** CoCl only.
- **Proactive blocking methods:** BlBu, BlCl, and optionally CoCl which is illustrated with the dashed bracket in Figure 2.11.

The example entities from Table 2.2 are extended in Table 2.4 and are used as running example to show the differences of the blocking methods. The six entities are limited to the URI, English label and the date properties. Table 2.5 provides an overview of the used blocking methods of this thesis. These methods are defined in the following with the use of the example entities when using the English label and date properties for building the blocks. The entities d_1 and y_1 as well as d_2 and y_2 represent duplicates of the different KGs.

KG	ID	URI	Label	Date
DBpedia	d_1	dbr:64th_Berlin_International_Film_Festival	64th Berlin International Film Festival	2014-02-16
DBpedia	d_2	dbr:Battle_of_Bouvines	Battle of Bouvines	1214-07-27
DBpedia	d_3	dbr:Battle_of_Waterloo	Battle of Waterloo	1815-06-18
YAGO	y_1	yago:64th_Berlin_International_Film_Festival	64th Berlin International Film Festival	2014-##-##
YAGO	y_2	yago:Battle_of_Bouvines	Battle of Bouvines	1214-07-27
YAGO	y_3	yago:Waterloo_Festival_for_Animated_Cinema	Waterloo Festival for Animated Cinema	-

Table 2.4: Example Entities for Blocking

Standard Blocking

Standard Blocking (StBl) is the most common lazy method for building blocks. For every distinct token t_i in all attribute values, a block b_i is created. All entities that contain t_i in any of the attribute values are assigned to b_i. Thus, StBl does not keep track of the attribute names and where the token was extracted. Blocks

Block Building	**Block Cleaning**	**Comparison Cleaning**
Standard Blocking (StBl)		
Attribute Clustering (ACl)		
	Block Filtering (BlFi)	
		Meta Blocking (MeBl)

Table 2.5: Blocking Methods

that only contain a single entity are discarded. StBl is parameter-free but might be modified with changing the tokenization function [47, p. 67] [55, p. 3]. Table 2.6 shows the created blocks for every distinct token in the English label and date properties for the example entities of Table 2.4. The blocks uniquely identified by the block tokens *for*, *Animated*, *Cinema*, *2014-02-16*, *1815-06-18*, and *2014-##-##* are discarded as they only contain a single entity.

Block Token	Assigned Entities	Extracted from Property
64th	d_1, y_1	Label
Berlin	d_1, y_1	Label
International	d_1, y_1	Label
Film	d_1, y_1	Label
Festival	d_1, y_1, y_3	Label
Battle	d_2, d_3, y_2	Label
of	d_2, d_3, y_2	Label
Bouvines	d_2, y_2	Label
Waterloo	d_3, y_3	Label
for	y_3	Label
Animated	y_3	Label
Cinema	y_3	Label
2014-02-16	d_1	Date
1214-07-27	d_2, y_2	Date
1815-06-18	d_3	Date
2014-##-##	y_1	Date

Table 2.6: Created Blocks for the Example Entities using Standard Blocking

Attribute Clustering

Attribute Clustering (ACl) is another lazy BlBu method where attribute names are clustered based on the "similarity of their values" [55, p. 3]. StBl is then applied for each cluster independently. This approach should lead to smaller blocks and therefore higher efficiency than StBl. ACl can be configured with different representation models and similarity measures. The representation model determines how the attribute names are modeled using tokenization or n-gram functions. The similarity measure then calculates a score based on the created attribute name sequences [47, pp. 69-71] [55, p. 3]. An **n-gram** is a substring having n consecutive characters [51, p. 9]. For the word "label" the bigrams ($n = 2$) are *la*, *ab*, *be*, and *el*, while the trigrams ($n = 3$) are *lab*, *abe*, and *bel*. Established techniques for the representation models and their similarity measures are:

- **Term Vector:** representation of the attribute names by vectors. With the distinct tokens of the attribute names as input, the TF/IDF similarity measure is used to create a single vector for each attribute name. The similarity of the attributes is calculated as cosine similarity of the term vectors [51, p. 9].
- **Character n-grams**: each attribute name is transformed to a set of n-grams. The similarity of those sets is then calculated using the Jaccard similarity measure [51, p. 9].
- **n-gram Graphs:** similar to the character n-grams model but connects neighboring n-grams with edges using contextual information. Neighboring n-grams "lie within a sliding windows of n characters" to each other. The edges are weighted with the distance of the n-grams in the original attribute name. When combining the n-gram graphs for each attribute name, the mean values of the edge weights are used. *Value similarity* is a graph metric that calculates the similarity of two n-gram graphs by evaluating the ratio of common edges with the same weights [28, p. 14] [51, p. 9].

As the attribute values of the examples of Table 2.4 are very diverse, the created blocks using Attribute Clustering with different representation models are the same as listed in Table 2.6 for the blocks of StBl. In addition, the *label* and *date* attribute names do not contain similar bi- or trigrams. Thus, the created blocks using ACl and Term Vector, Character n-grams or n-gram Graphs with bi- or trigrams as representation models, do not differ from Table 2.6. Nevertheless, the created clusters for the distinct attribute names are clearly separated.

As a counterexample, some labels contain year numbers such as *1974 FIFA World Cup*. The created block for the token *1974* only contains entities that have this specific token in the label attribute. Contrariwise, the *1974* block that might be

created using the date property consists of entities that possess *1974* as date value. Even though the two blocks are created using the same token *1974*, the assigned entities would differ which leads to overall smaller blocks.

Block Filtering

Block Filtering (BlFi) is a block-refinement method that operates on entity-level. The core assumption is that "the larger a block is, the less important it is for its entities" as it mainly contains redundant comparisons [55, p. 4]. Hence, entities are removed from its largest blocks. Blocks are globally sorted based on their cardinality and every entity e_i is retained "in the N_i smallest blocks in which it appears" based on "$N_i = \lfloor r \times |B_i| \rfloor$, where B_i stands for the set of blocks containing e_i and $r \in [0, 1]$ specifies a percentage" [55, p. 4] [54]. The cardinalities of the blocks listed in Table 2.6 are ascending sorted in Table 2.7. Due to the small number of examples and the similar attributes, the number of executed comparisons rises from 9 for the brute-force approach to 13 when applying StBl without any block-refinement methods.

Block Token	Assigned Entities	Executed Comparisons	Cardinality
64th	d_1, y_1	(d_1, y_1)	1
Berlin	d_1, y_1	(d_1, y_1)	1
International	d_1, y_1	(d_1, y_1)	1
Film	d_1, y_1	(d_1, y_1)	1
Bouvines	d_2, y_2	(d_2, y_2)	1
Waterloo	d_3, y_3	(d_3, y_3)	1
1214-07-27	d_2, y_2	(d_2, y_2)	1
Festival	d_1, y_1, y_3	$(d_1, y_1), (d_1, y_3)$	2
Battle	d_2, d_3, y_2	$(d_2, y_2), (d_3, y_2)$	2
of	d_2, d_3, y_2	$(d_2, y_2), (d_3, y_2)$	2

Table 2.7: Sorted Blocks for the Example Entities using Standard Blocking

Entities d_1 and y_1 both appear in the *64th*, *Berlin*, *International*, *Film*, and *Festival* blocks that are sorted in ascending order. With an example ratio of 0.5 every entity is retained in 50% of its smallest blocks. Therefore, when keeping the input order for blocks that have the same cardinalities, the blocks *64th* and *Berlin* are retained for the entities while they are removed from the *International*, *Film*, and *Festival* blocks. Entity d_2 is retained in the *Bouvines* and *1214-07-27* blocks and is removed from the *Battle* and *of* blocks. The same applies for entity y_2. Due to the applied floor function, d_3 and y_3 are only kept in the *Waterloo* block.

The *International*, *Film*, *Festival*, *Battle* and *of* blocks are then discarded for this example as no entities are retained in those blocks. This saves many redundant without missing required comparisons. Depending on the internal sorting function of blocks that have the same cardinality, the retained blocks for d_1 and y_1 differ. Table 2.8 lists the retained blocks after applying Block Filtering with a ratio of 0.5 for the example entities. Thus, the number of executed comparisons decreases from 17 to 5 due to the saved redundant comparisons.

Block Token	Assigned Entities	Executed Comparisons	Cardinality
64th	d_1, y_1	(d_1, y_1)	1
Berlin	d_1, y_1	(d_1, y_1)	1
Bouvines	d_2, y_2	(d_2, y_2)	1
Waterloo	d_3, y_3	(d_3, y_3)	1
1214-07-27	d_2, y_2	(d_2, y_2)	1

Table 2.8: Retained Blocks for the Example Entities using StBl and BlFi with a ratio of 0.5

Meta Blocking

Meta Blocking (MeBl) is a comparison-refinement method that "targets redundant and superfluous comparisons in *redundancy-positive* block collections" [55, p. 4]. The idea is to use this redundancy as evidence for the probability of two entities to be duplicates: two entities likely match if they share many blocks [24, p. 4]. MeBl has four consecutive steps [47, pp. 97-102]:

1. **Graph Building:** "creates an undirected graph, where the nodes correspond to entities and the edges connect the co-occuring entities" within the blocks [55, p. 4].
2. **Edge Weighting:** calculates edge weights using a weighting scheme.
3. **Graph Pruning:** removes edges using a pruning algorithm.
4. **Block Collecting:** collects the remaining blocks.

As there are no parallel edges, all redundant comparisons are eliminated automatically when building the graph. Different weighting schemes and pruning algorithms therefore try to reduce the number of superfluous comparisons. A weighting scheme calculates edge weights based on the shared blocks of entities. The pruning algorithm then discards comparisons based on these edge weights [47, pp. 97-104] [55, p. 4].

Weighting schemes for MeBl are Aggregate Reciprocal Comparisons Scheme (ARCS), Common Blocks Scheme (CBS), Enhanced Common Blocks Scheme (ECBS), Jaccard Scheme (JS), and Enhanced Jaccard Scheme (EJS):

- $\mathit{ARCS}(e_i, e_j, B) = \sum_{b_k \in B_{ij}} \frac{1}{||b_k||}$
- $\mathit{CBS}(e_i, e_j, B) = |B_{ij}|$
- $\mathit{ECBS}(e_i, e_j, B) = \mathit{CBS}(e_i, e_j, B) \times \log \frac{|B|}{|B_i|} \times \log \frac{|B|}{|B_j|}$
- $\mathit{JS}(e_i, e_j, B) = \frac{|B_{ij}|}{|B_i|+|B_j|-|B_{ij}|}$
- $\mathit{EJS}(e_i, e_j, B) = \mathit{JS}(e_i, e_j, B) \times \log \frac{|V_B|}{|v_i|} \times \log \frac{|V_B|}{|v_j|}$

where the set of shared blocks by the entities e_i and e_j are denoted by B_{ij}, "the total number of nodes in the blocking graph of B" is $|V_B|$, and the node degree of e_i is $|v_i|$ [55, p. 4].

The pruning algorithm then decides which edges of the graph are retained:

- Cardinality Edge Pruning (CEP): "top-k weighted edges of the graph" [55, p. 4].
- Cardinality Node Pruning (CNP): "top-k weighted edges from the neighborhood of each node" [55, p. 4].
- Reciprocal Cardinality Node Pruning (ReCNP): "edges that are among the top-k weighted ones for both adjacent entities" [55, p. 4].
- Reciprocal Weight Node Pruning (ReWNP): "edges that exceed the local threshold of both adjacent node neighborhoods" [55, p. 4].

The names of the different pruning algorithms implicitly indicate two characteristics [47, p. 102]:

- Thresholds: cardinality- or weight-based.
- Centricity: edge or node.

In addition, the scope of the pruning algorithms differs: CEP, CNP, and ReCNP calculate global cardinalities, whereas ReWNP only uses local weights that are compared to the threshold [24, p. 4]. The threshold value of the pruning algorithm is determined automatically [55, p. 4]. Main challenge of MeBl is the high computational complexity for weighting the edges [47, p. 104]. Figure 2.12 illustrates

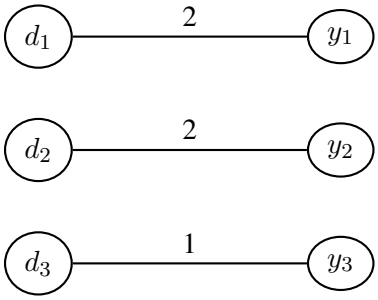


Figure 2.12: MeBl Undirect Graph Example using the CBS Weighting Scheme

the undirected graph for the results of the BlFi example in Table 2.8 using CBS for weighting the edges.

The undirected graph discards all redundant comparisons and therefore decreases the number of overall comparisons to 3 without pruning the graph. A pruning algorithm such as CEP that retains the top-k weighted edges of the graph prunes the superfluous comparisons of the nodes d_3 and y_3 for $k = 2$, thereby decreasing the number of overall comparisons to 2. Note that k is determined automatically by the pruning algorithm. Hence, all redundant and superfluous comparisons are discarded for the example entities of Table 2.4 using Standard Blocking, Block Filtering with a ratio of 0.5, and Meta Blocking.

2.5.3 Blocking Evaluation

For evaluating the performance of blocking algorithms, three different effectiveness measures are used: Pairs Completeness (PC), Pairs Quality (PQ), and Reduction Ratio (RR). They "take values in the interval $[0, 1]$, with higher values indicating higher effectiveness" [55, p. 2]. All measures are defined for matching two clean entity collections E_1 and E_2.

Pairs Completeness

The PC measure for E_1, E_2, and their set of blocks B is defined as:

$$PC(B, E_1, E_2) = \frac{\mid D(B) \mid}{\mid D(E_1 \cap E_2) \mid} \tag{2.19}$$

where $D(B)$ denotes the "set of co-occuring duplicate entities" in the blocks and $D(E_1 \cap E_2)$ is the intersection of the existing duplicates in the entity collections [55, p. 2]. Thus, PC assesses the ratio of detected matches to the existing matches and can be regarded as "*optimistic recall*" [47, p. 22].

Pairs Quality

PQ "estimates the portion of non-redundant comparisons that involve matching entities" [55, p. 2]:

$$PQ(B) = \frac{\mid D(B) \mid}{\mid\mid B \mid\mid} \tag{2.20}$$

where $||B||$ refers to the "total number of pairwise comparisons in B":

$$||B|| = \sum_{b_i \in B} ||b_i|| \tag{2.21}$$

and $||b_i||$ denotes the number of comparisons in block i:

$$||b_i|| = |b_{i,1}| \times |b_{i,2}| \tag{2.22}$$

"where $b_{i,1} \subseteq E_1$ and $b_{i,2} \subseteq E_2$ are the disjoint, inner blocks of b_i" [55, p. 2].

Therefore, PQ calculates the portion of the detected matches to the number of executed comparisons and can be regarded as "*pessimistic precision*" since blocking usually involves many redundant comparisons [47, p. 22].

Reduction Ratio

The RR measure "estimates the portion of comparisons that are avoided in B with respect to the naïve brute-force approach" [55, p. 2]:

$$RR(B, E_1, E_2) = 1 - \frac{\mid\mid B \mid\mid}{\mid E_1 \mid \times \mid E_2 \mid} \tag{2.23}$$

High RR values indicate efficient blocking algorithms that scale well to large datasets.

Trade-off between PC and PQ-RR

The initial aim of blocking is to identify "all existing duplicates with the minimum number of comparisons", thereby maximizing all three effectiveness measures [55, p. 2]. Nevertheless, a blocking method with a high redundancy increases PC but decreases PQ and RR, and vice versa [50, p. 4].

Hence, blocking should try to balance the effectiveness measures with maximizing PC and RR simultaneously [55, p. 6]. This combined measure is denoted as α [50, p. 6]:

$$\alpha(B, E_1, E_2) = RR(B, E_1, E_2) \times PC(B, E_1, E_2) \tag{2.24}$$

The use of this combined measure α should lead to a balanced increase in the overall effectiveness that arises from "the careful removal of unnecessary comparisons between irrelevant entities, rather than from a blind process" [51, p. 4].

Efficiency Measures

In addition to the effectiveness measures of the blocking algorithm, the Overhead Time (OTime) and Resolution Time (RTime) efficiency measures estimate the duration of the blocking algorithm:

- **OTime:** "is the time between receiving the input entity collection(s) and returning the set of blocks as output" [55, p. 2].
- **RTime:** "adds to OTime the time required to perform all pair-wise comparisons in B with an entity matching technique" [55, p. 2].

Lower values for the OTime an RTime measures indicate a higher efficiency. For calculating the approximate RTime a basic similarity measure such as Jaccard or an estimate for executing a single comparison can be used [55, p. 2].

2.6 Related Work

Data integration and the Semantic Web are two current research topics that are combined more and more. With the rising amount and size of datasets, the Semantic Web offers a great variety of data from different domains to integrate.

Integrating Data from the Semantic Web

The size, structure, and noise of those datasets lead to many challenges. Hence, cleaning big and heterogeneous data sources is an important step before integrating the data into productive systems. Rahm and Do [58] present different approaches and tools for data cleaning. Weis *et al.* [72] propose a benchmark for identity resolution with a main focus on XML data. Cluster-based approaches for scaling the integration to many different data sources are introduced by Rahm [57]. These data sources might be extracted from the Semantic Web as well.

For dealing with the size of the datasets, MapReduce techniques are adapted for data integration. Benbernou *et al.* [1] introduce query rewriting strategies that are based on MapReduce to query big data sources in RDF format. An approach for parallelizing the full data integration process using relational learning techniques is presented by Costa and Oliveira [17]. Existing identity resolution approaches are evaluated on real-world datasets from the bibliographic and e-commerce domain by Köpcke *et al.* [41].

An approach for detecting missing *owl:sameAs* links for duplicates within the Semantic Web that only uses the stripped URIs of the entities is proposed by Papadakis *et al.* [48]. Therefore, the URIs are split into the *PI(S)* scheme that stands

for the *Prefix-Infix(-Suffix)* order. This technique of removing the URI prefix is also used in the implementation part of the thesis.

A current survey for identifying duplicates within Knowledge Graphs is presented by Shen *et al.* [62]. The need of efficient and scalable systems for integrating data from the Semantic Web is one of the main findings of their survey. In addition, the demand for benchmark datasets to test the different frameworks and methods with comparable results is highlighted.

Similarity Measures

Elmagarmid *et al.* [25] present a literature review of different string similarity measures, matching rules, as well as blocking algorithms. This paper provides an eligible introduction to the research field of duplicate detection.

A comparison of those string similarity measures for matching names and records and the implementation of the Java *Second String*[1] library is introduced by Cohen *et al.* [15]. The paper also provides performance measures of the different matching methods. Among others, the Levenshtein, Jaro, JaroWinkler, Jaccard, and TF/IDF string similarity measures are analyzed. The *Second String* library is also used for comparing strings within the experiments of this thesis.

Blocking

The concept of blocking is a hot current research topic. Christen [14] presents a survey of six blocking techniques with twelve variations in total. The main finding of the paper is the importance of a proper blocking key definition that fits the underlying data.

Draisbach and Naumann [22] combine the concepts of blocking and windowing and present the *Sorted Blocks* algorithm that requires less comparisons in total than basic blocking or windowing methods. Windowing techniques create descriptive blocking keys based on attribute values, sort the entities by those keys and then compare each entity with a fixed number of previous and successive entities.

Kim and Lee [37] introduce *HARRA* (HAshed RecoRd linkAge), a blocking algorithm for large datasets that uses *Iterative Locality-Sensitive Hashing* techniques in order to increase the performance. *HARRA* hashes the entities based on the attribute values and then uses approximate nearest neighbor algorithms to assign similar entities to the same blocks. The use of hash tables makes the approach scalable for large input datasets.

Papadakis *et al.* [49] present an attribute-agnostic block building mechanism that is combined with efficient block processing techniques. The attribute-agnostic

[1] `http://secondstring.sourceforge.net/`

block building method does not depend on the same underlying schema of the input data and places "entities that contain the same token in any attribute [...] in one block" [49]. The approach is evaluated on large heterogeneous real-world datasets for showing the effectiveness and efficiency of the novel method.

These findings are enhanced by Papadakis *et al.* [50] with a dataset of 182 million entities. In addition, new effectiveness measures of blocking such as Pairs Completeness and Reduction Ratio are introduced. These blocking-specific effectiveness measures are used for evaluating the blocking methods as defined in Chapter 2.5.3.

Another efficient blocking algorithm that does not depend on the user definition of the blocking key is MIFIBlocks as presented by Kenig and Gal [36]. MIFIBlocks uses the approach of selecting maximum frequent itemsets. For showing the effectiveness, the paper evaluates the algorithm to other blocking methods on different synthetic and real-world datasets.

The Meta Blocking comparison-refinement method is introduced by Papadakis *et al.* [52]. The performance of Meta Blocking is evaluated on three large real-world datasets that indicate a significant enhancement regarding efficiency with trading some effectiveness in comparison to attribute-agnostic blocking method as defined in [49].

For further increasing the efficiency of Meta Blocking, Efthymiou *et al.* [24] as well as Papadakis *et al.* [53] [54] propose different approaches. Papadakis *et al.* [53] introduce three new pruning algorithms for increasing precision while also reducing the Overhead Time. The experiments are tested on 19 large and heterogeneous datasets and best configurations for different datasets are demonstrated. Additional pruning algorithms are presented and tested on six real-world datasets by Papadakis *et al.* [54]. For parallelizing the execution of the Meta Blocking method, Efthymiou *et al.* [24] introduce scalable algorithms using MapReduce techniques that are tested on four real-world datasets.

Costa and Oliveira [18] present the CER-blocking scheme that uses inverted index techniques and data evidence information in order to cope with the heterogeneous and noisy data of the Semantic Web. Main finding of the conference paper is that the more descriptive information is contained in data triples, the better the effectiveness of blocking becomes. Nevertheless, the noise of the extracted data remains a challenging task and has to be considered carefully.

The key related work of this thesis is the *Comparative Analysis of Approximate Blocking Techniques for Entity Resolution* conference paper by Papadakis, Gal, and Palpanas, where they present an empirical survey with "17 state-of-the-art blocking methods and 6 popular datasets" [55]. The two largest real-world datasets of the experiments in the paper deal with the matching of clean entity collections. Therefore, they possess similar characteristics as the datasets that are used for the

experiments of this thesis [55]:

1. Movies extracted from DBpedia and IMDB[2]
2. Two different DBpedia versions (3.0rc and 3.4)

Both movie datasets contain 27,615 and 23,182 entities, leading to 6.4×10^8 comparisons that are executed by the brute-force approach. The datasets of the DBpedia versions consist of approximately 1.2×10^6 (3.0rc) and 2.2×10^6 (3.4) entities, and consequently would require 2.58×10^{12} comparisons without applying any blocking methods.

Supervised Learning of Matching Rules

After creating the set of candidate pairs, matching rules decide which candidates are assumed to describe duplicates. Defining those rules by hand is difficult. Thus, genetic programming algorithms and active learning methods are used for creating those matching rules automatically.

Bilenko and Mooney [6] introduce MARLIN (Multiply Adaptive Record Linkage with INduction) that uses edit distance and Support Vector Machine (SVM) methods to learn classifiers that predict duplicates. The experimental results on six datasets indicate a significant improvement in regards to primitive decision tree learners.

Isele and Bizer [34] [35] present the supervised GenLink algorithm. The GenLink algorithm outperforms other current genetic programming approaches with reaching a higher accuracy [34]. The extended ActiveGenLink algorithm uses active learning and requires only a small number of labeled candidate pairs that the user has to confirm or decline [35].

GALER is another genetic algorithm that uses active learning as introduced by Sun *et al.* [65]. GALER uses different specialized crossover operators for aggregations, similarity functions, and thresholds in order to classify the entities. The paper presents the detailed implementation of the algorithms and compares GALER to other supervised approaches on two real-world as well as one synthetic dataset to demonstrate the improvement.

Data Fusion

An introduction of the data fusion step in the date integration process is provided by Bleiholder and Naumann [8]. The paper presents the main goals and challenges of data fusion. In addition, integration systems that perform data fusion

[2] `http://www.imdb.com/`

tasks are compared to each other. Bleiholder and Naumann [9] present high-level descriptions of different conflict resolution functions and respective strategies for data fusion. The developed HumMer (Humboldt-Merger) system implements the presented functions.

A more recent data fusion approach focuses on data that is extracted from Knowledge Graphs. Dong *et al.* [21] describe the limitations of existing data fusion approaches for finding the true values and therefore improve existing data fusion methods for dealing with noise and large data from the Semantic Web. The experiments are evaluated on 1.6 billion data triples.

Frameworks

An introduction of different frameworks for entity matching is presented by Köpcke and Rahm [40]. The eleven examined frameworks are analyzed based on provided functions for blocking, matching, and potentially learning algorithms. In addition, the results of the frameworks are compared on real-world datasets.

The underlying Java Blocking Framework[3] of Papadakis *et al.* [55] with implemented methods for BlBu, BlCl, and CoCl is publicly available and is used for testing different blocking methods for the experiments of the thesis. Thus, the obtained results are comparable to the outcomes of the paper.

The *Scaling Entity Resolution to Large, Heterogeneous Data with Enhanced Meta-blocking* conference paper by Papadakis, Pastefanatos, Palpanas, and Koubarakis describe the Block Filtering and Meta Blocking algorithms in detail [54]. This paper is therefore used as basis for implementing these algorithms.

The *Silk Linked Data Integration Framework*[4] uses the genetic programming algorithm *GenLink* for learning matching rules. *GenLink* learns "linkage rules from a set of existing reference links" and automatically [34]:

- selects properties that are discriminative,
- normalizes the data,
- selects distance or similarity measures with suitable thresholds, and
- aggregates the results.

For this thesis, the Silk framework is used to learn matching rules that are compared to a user-defined baseline rule. The Java Web Data Integration Framework[5]

[3] http://l3s.de/~papadakis/erFramework.html

[4] http://silkframework.org/

[5] http://dws.informatik.uni-mannheim.de/en/teaching/courses-for-master-candidates/ie670webdataintegration/

that is provided for the *Web Data Integration* course at the University of Mannheim by Prof. Dr. Christian Bizer is used as basic structure for the full data integration process for the experiments of the thesis and the implemented Web application.

Chapter 3

Data Source Selection and Schema Mapping

This chapter provides an overview of the used datasets for the experiments of the thesis and the mapping to the target schema. The target schema properties are outlined in Chapter 3.1. Chapter 3.2 describes the basic requirements of the data sources and provides basic definitions and characteristics for each KG. The target schema matching with the data sources is defined in Chapter 3.3. In Chapter 3.4, the approach of collecting and translating the data of the selected sources to the target schema is outlined.

3.1 Target Schema Properties

The event target schema has the following properties:

- URI: unique identifier of the event instance.
- Label: all English labels of the event.
- Links: instances from other KGs that describe the same real-world entity.
- Date: temporal information of the event.
- Coordinates: latitude and longitude according to the World Geodetic System 1984 (WGS84) coordinate positioning system [68].
- Location: spatial instances that are connected to the event.

The *location* itself can contains labels, coordinates, and links to other instances. Due to the uncertainty in the Semantic Web, all properties (except the URI) can

have zero, one or multiple values. The URI property is used as unique identifier of the event instances form the different KGs and labels are limited to the English language.

3.2 Source Selection

As the final Web application that is developed for the thesis is based on a virtual architecture, the main requirement for the data sources is the availability of a public SPARQL endpoint. Therefore, the DBpedia [42], YAGO [64], and Wikidata [67] KGs are analyzed regarding available event data. Those events should usually have spatial and temporal properties, making the integration of this data interesting and challenging. For mapping the source data to the target schema, the most promising properties regarding availability should be found.

3.2.1 DBpedia

The public SPARQL endpoint at `http://dbpedia.org/sparql` is used for querying the DBpedia KG [1]. The event class in the DBpedia ontology is identified as *dbo:Event*. Thus, instances of the event class are defined as:

$$\textit{?event rdfs:type dbo:Event} \tag{3.1}$$

With this definition, there are $76{,}022$ distinct events in the DBpedia KG using the current 2016-04 dataset and the query found in A.1. After analyzing different counts and expressiveness measures (such as indegrees and outdegrees), the following properties are chosen for DBpedia:

- *rdfs:label* for the label.
- *owl:sameAs* for the link to instances from other KGs.
- *dbo:date* for the date property.
- *geo:lat* and *geo:long* for the coordinates.
- *dbo:place* for the location.

The *rdfs:label*, *owl:sameAs*, *geo:lat*, and *geo:long* properties are also used for the *location* for gathering additional information about the *dbo:place* instances.

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

3.2.2 YAGO

As DBpedia and YAGO[2] both extract information from Wikipedia, they are pretty similar in nature. The SPARQL endpoint at `https://linkeddata1.calcul.u-psud.fr/sparql` is used to query YAGO. In the YAGO ontology the event class is identified by *yago:wordnet_event_100029378* so event instances are defined as:

?event rdfs:type yago:wordnet_event_100029378 (3.2)

With the query found in A.2, there are 417,276 distinct events in the current YAGO version 3 KG. Similar to DBpedia, the following properties were chosen for YAGO:

- *rdfs:label* for the label.
- *owl:sameAs* for the link to instances from other KGs.
- *yago:happenedOnDate* for the date property.
- *yago:hasLatitude* and *yago:hasLongitude* for the coordinates.
- *yago:isLocatedIn* for the location.

For the *location* the *rdfs:label*, *owl:sameAs*, *yago:hasLatitude*, and *yago:hasLongitude* properties are used as well to receive additional data values from the YAGO KG.

3.2.3 Wikidata

The SPARQL endpoint at `https://query.wikidata.org/` is used for querying Wikidata[3]. A significant difference in contrast to DBpedia and YAGO is that the Wikidata endpoint is updated every few seconds and does not have major releases every several months or years. The event class in Wikidata is called *wd:Q1656682* and instances are assigned using the *instance of* (*wdt:P31*) property:

?event wdt:P31 wd:Q1656682 (3.3)

With the query from A.3, 3934 distinct events are received on November 22, 2016. The most frequent location-related property is *wdt:P17* with 168 instances having this *country* property. In addition, only 49 events have the most frequent time-related property *point in time* (*wdt:P585*) and there are no direct links to the YAGO or DBpedia KGs. Therefore, Wikidata as data source is discarded due to limited number of event instances, their low expressiveness, and the missing links to DBpedia and YAGO that are required for creating a gold standard.

[2] `http://yago-knowledge.org/`
[3] `https://www.wikidata.org`

3.3 Target Schema Matching

For translating the event instances of the data sources to a homogeneous target schema, the matches and mappings are defined by hand. Figures 3.1 and 3.2 illustrate the relations of the data source properties to the target schema for DBpedia and YAGO respectively. As the structure of DBpedia and YAGO is very similar, no complex transformations are executed. Hence, all properties are gathered and passed to the specific target schema property.

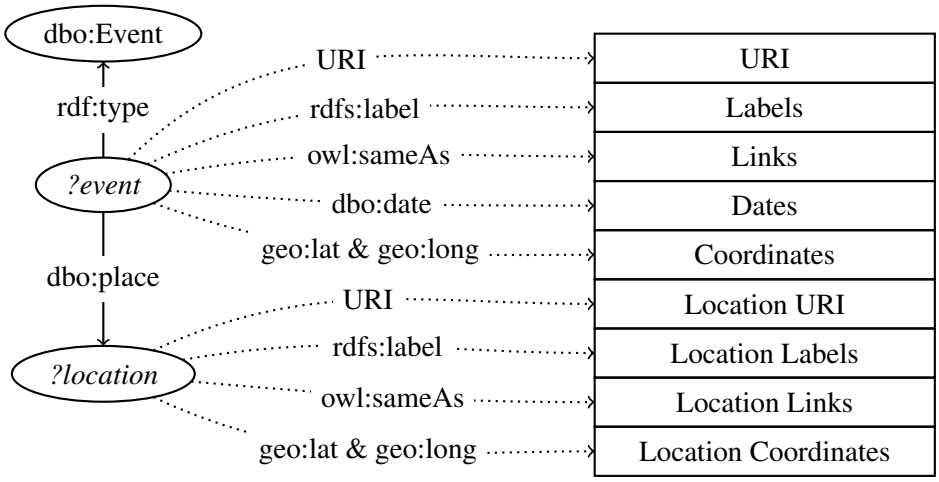


Figure 3.1: Target Schema Correspondences for DBpedia

3.4 Data Collection and Schema Mapping

For evaluating the blocking algorithm and matching rules, offline datasets in XML format are created for DBpedia and YAGO. Therefore, a Java program queries the SPARQL endpoints for each event URI and then combines the gathered data in the target schema. The only requirement of the event instances is the availability of at least one English label. After cleaning the non-parsable event URIs, there are $56{,}725$ distinct DBpedia events and $417{,}113$ distinct YAGO events left. Table 3.1 gives and overview of the dataset properties and their rounded densities.

Every instance in the Semantic Web is required to have a URI as unique identifier. In addition, an English label is defined as mandatory. Thus, the densities for these two properties are 1.0. All other property densities indicate that DBpedia indeed has far fewer event instances but they possess more properties on average. The average densities are 60.0% for DBpedia and 49.5% for YAGO.

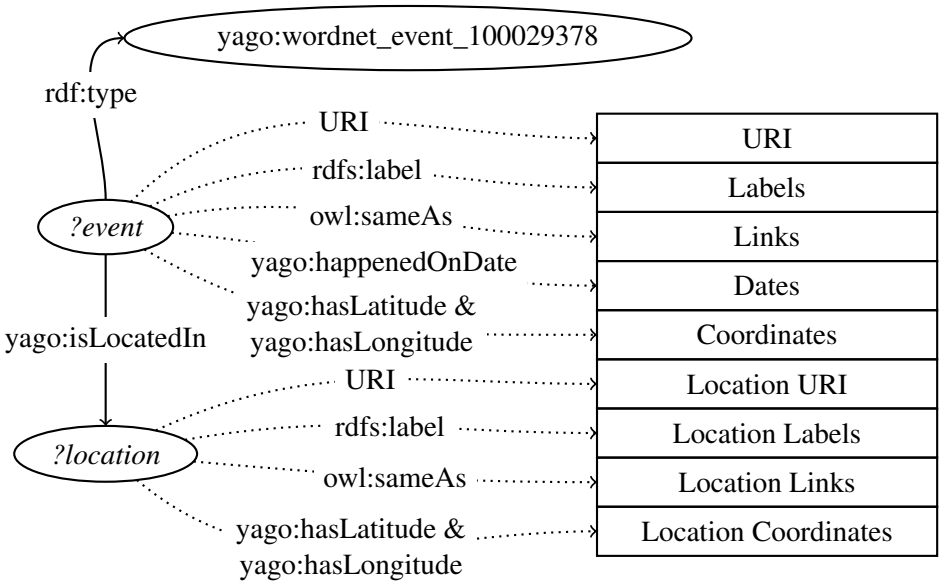


Figure 3.2: Target Schema Correspondences for YAGO

Property	DBpedia	YAGO
URI	1.000	1.000
Label	1.000	1.000
Links	0.981	0.723
Date	0.253	0.049
Coordinates	0.137	0.031
Location	0.230	0.168

Table 3.1: Dataset Property Densities

Chapter 4

Identity Resolution Experiments

This chapter describes the blocking and matching rule experiments as well as the implemented Web application. The experimental setup is outlined in Chapter 4.1 while Chapter 4.2 provides the results and their discussion. The results for the blocking methods and matching rules are presented successively as the matching rules are tested on the full datasets using the best blocking method. In addition, the implemented Web Application for integrating the event data from DBpedia and YAGO is presented in Chapter 4.3.

4.1 Experimental Setup

The general setup of the experiments is presented in Chapter 4.1.1 and the used frameworks are outlined in Chapter 4.1.2. Chapter 4.1.3 describes how the gold standard is created for measuring the performance of the experiments. The different analyzed datasets and parameters for the blocking methods are outlined in Chapter 4.1.4 while Chapter 4.1.5 describes the matching rules experiments.

4.1.1 General Setup

All experiments are executed on a Debian version 8 server using a x86_64 architecture. The server has 24 CPUs of the type *Intel Xeon E5-2630 v2* with 2.60 GHz each and 378 GB RAM in total. Java version 1.8.0_121 is used to run the programs. When measuring the runtime of algorithms, the measurements are repeated ten times and the average duration is reported.

4.1.2 Frameworks

Three different frameworks were used for the experiments of the thesis:

1. **Java Blocking Framework**[1]**:** foundation for running all blocking experiments.
2. **Silk**[2]**:** automatically learning matching rules using the *GenLink* genetic programming algorithm.
3. **Java Web Data Integration Framework**[3]**:** provides the basis for implementing the full data integration process. The best blocking method is implemented in the framework to test the matching rules on the full (blocked) datasets.

4.1.3 Gold Standard Creation

The gold standard is required for testing the performance of the blocking methods and matching rules. Thus, entity pairs are labeled as positives (duplicates) or negatives (non-duplicates). The PC, PQ, and RR effectiveness measures of blocking and the precision, recall, F-measure, and sigh test scores for the matching rules are calculated based on this labeled gold standard.

The existing *owl:sameAs* links of the entities are used to create the positive examples automatically. Therefore, the correctness of those direct links from DBpedia instances to YAGO instances and vice versa is assumed. The *owl:sameAs* property is excluded for learning matching rules as it is used for creating the gold standard: The preservation of the property would lead to overfitting, as the rules could simply check the values of the *owl:sameAs* properties.

For creating the negative examples automatically as well, it is assumed that all entities in the set of positives are different from each other except to the linked ones. In order to make the negative examples more significant and to decrease the size of examples for performance reasons, only negatives examples that are similar to each other are included in the gold standard. This similarity is defined as edit distance of less than 3 for the stripped URI values. A stripped URI is the remaining string without the prefix that is distinct for the KG. The prefixes for the KGs are:

- DBpedia: `http://dbpedia.org/resource/`
- YAGO: `http://yago-knowledge.org/resource/`

Thus, the value for the stripped URI of `http://dbpedia.org/resource/Woodstock_1999` is *Woodstock_1999*. With this approach, only difficult negative

[1] `http://l3s.de/~papadakis/erFramework.html`
[2] `http://silkframework.org/`
[3] `http://dws.informatik.uni-mannheim.de/en/teaching/courses-for-master-candidates/ie670webdataintegration/`

examples are added to the gold standard which should lead to more robust matching rules.

The gold standard, that is created under these assumptions, has 9709 positive and 6000 negative examples. Nevertheless, due to the *Open World* and *Non-Unique Naming Assumptions* of the Semantic Web, the automatic creation of the gold standard might lead to some incorrect positive or negative examples: *owl:sameAs* links might be incorrect and there might be missing links of instances describing the same real-world entity even though the instance is linked to another instance. These potential errors would even decrease the performance of a perfect matching rule. As perfect matching rules normally indicate overfitting, this potential small amount of errors in the gold standard can be discarded.

Gold Standard Characteristics

In order to get an overview of the gold standard characteristics and to create a baseline for the matching rules, the similarity of the stripped URIs is calculated for the positive and negative examples. The scaled Levenshtein is used as similarity measure. All similarity scores are rounded to two digits after the decimal point and the entity pairs are added to buckets in a histogram to visualize the distribution of entities. Thus, 101 buckets from 0.00 to 1.00 with steps of size 0.01 are created and the entities are added to the respective bucket based on their rounded similarity score. Figure 4.1 shows the histogram with the green bars indicating entities that are linked and therefore considered as duplicates and red bars that illustrate the entities in the gold standard that are not linked and are assumed to be non-matches. The buckets of the histogram are shown on the x-axis while the y-axis indicates the number of entities in the block. The bucket orientation of the y-axis does not specify negative counts but separates the duplicates (top) and non-duplicates (bottom).

Figure 4.1 clearly shows that the stripped URIs of the linked entities are identical in most cases, which is indicated with the large green bar with a similarity score of 1. The non-duplicates are much more diverse but have a peak between 0.7 and 0.8. In order to view the distribution of the smaller block as well, Figure 4.2 illustrates the histogram with a modified y-axis using a natural logarithm with base e on the counts.

The logarithmic y-axis highlights the small number of outliers, i.e. duplicates with a low similarity score or non-duplicates with a very high similarity score. Predicting those outliers might be difficult for a matching rule, while very similar duplicates and non-duplicates having a high distance score are easy to predict. The baseline matching rule will be created with the scaled Levenshtein similarity measure on the stripped URI. All learned rules will then be compared to this baseline

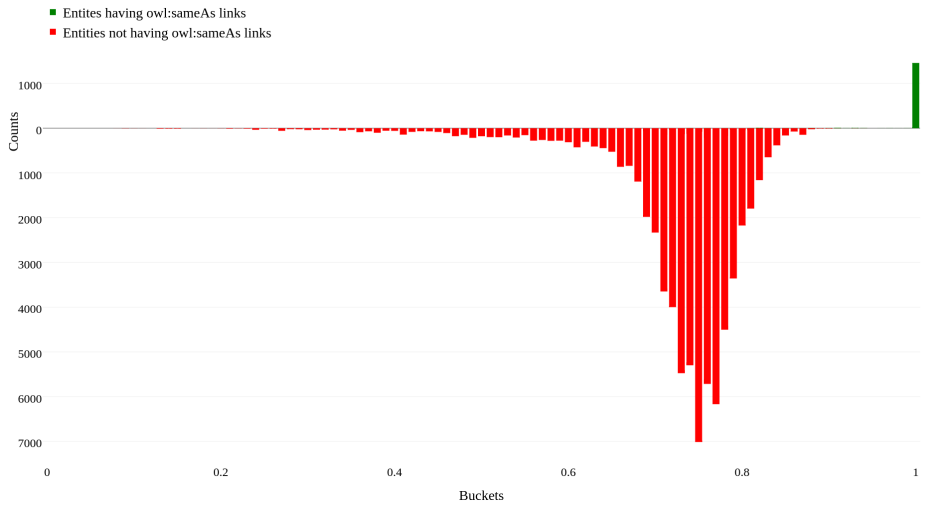


Figure 4.1: URI Similarity Distribution of the Gold Standard using Scaled Levenshtein

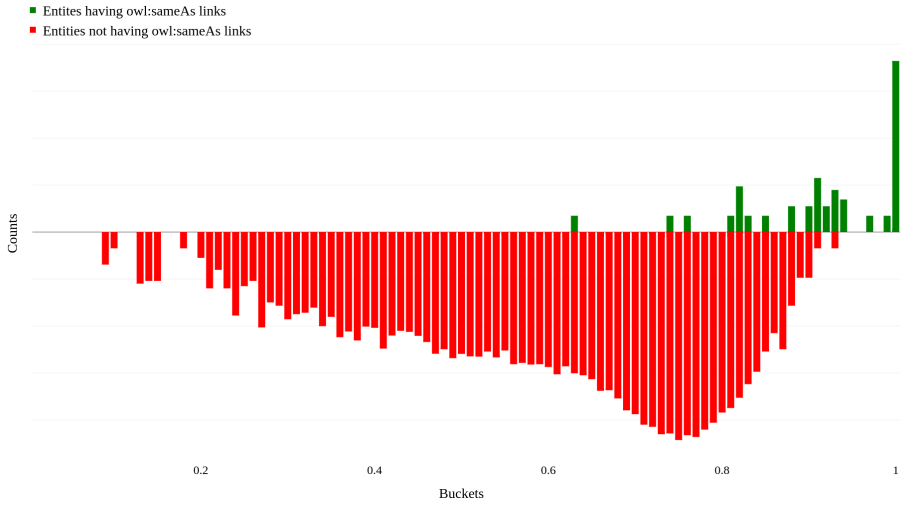


Figure 4.2: Logarithmic URI Similarity Distribution of the Gold Standard using Scaled Levenshtein

rule. The histogram already indicates that the threshold of the baseline rule will probably be between 0.9 and 1 as this score clearly separates the duplicates and the non-duplicates when regarding Figure 4.1.

4.1.4 Blocking Methods

The goal of blocking is to maximize the α value of the blocking method as defined in Chapter 2.5.3. The α value combines the number of saved comparisons (RR) as well as the completeness of the blocking method (PC). Therefore, α is an indicator for the time efficiency and the recall. All blocking methods listed in Table 2.5 are tested with different parameter and datasets which are described in the following.

Subsets

The blocks of the algorithms are created based on the attribute values of the entity collections. Hence, five subsets of the datasets are created to test the usefulness of the different attributes. The subsets are identified with the numbers of the list and include the following attributes:

1. Stripped URI
2. Stripped URI + Labels
3. Stripped URI + Links
4. Stripped URI + Labels + Links
5. All attributes

The runtime of building the blocks decreases with a smaller number of attributes as entities have to be assigned to fewer blocks. More attributes indicate an increasing number of diverse values and thus more block assignments for the entities. In addition, the number of blocks is reduced with fewer attribute values since different attributes include more diverse values. The idea of creating those subsets is to examine whether it is sufficient for the blocking methods to create the blocks with fewer attributes and therefore increase the time efficiency.

Parameter

Table 4.1 gives an overview of the blocking methods, the sub-tasks that they work on, the parameters that have to be specified, the scope of the blocking method, and the number of different settings that are tested for the experiments of this thesis. All blocking methods are analyzed for the five different subsets independently. As the sub-tasks of blocking are consecutive, the methods are tested successively for each of the subsets.

First, the best parameters for the BlBu task are determined. StBl builds blocks with the tokenized attribute values and is therfore parameter-free. For ACl, five

Sub-task	Method	Parameter	Scope	#settings
BlBu	StBl	parameter-free	-	1
BlBu	ACl	5 representation models	blocking key use	5
BlCl	BlFi	$r \in [0.05, 1.0]$ with steps of size 0.05	entity-level	20
CoCl	MeBl	5 weighting schemes 4 pruning algorithms	superfluous comparisons	20

Table 4.1: Parameters of the Blocking Methods

different representation models are analyzed for finding the model with the highest α value for each subset:

- Term Vector
- Character Bigrams
- Character Trigrams
- Bigram Graphs
- Trigram Graphs

The output of BlBu serves as input for the BlCl sub-task. As two different BlBu methods are tested, both methods are analyzed independently. 20 different values between 0.05 and 1 for the ratio are tested for the BlFi method that operates on the entity-level. This corresponds to the settings of the comparative analysis of blocking techniques by Papadakis *et al.* [55, p. 5] to obtain comparable results. Therefore, parameters that discard almost all entity block assignments ($r = 0.05$) to those that retain all assignments ($r = 1$) are analyzed. The refined blocks with the ratio that maximizes α for each subset are then passed to the CoCl method.

For MeBl all five weighting schemes are examined with the four different pruning algorithms to reduce superfluous comparisons. This leads to 20 different combinations for each of the five subsets. The ARCS, CBS, ECBS, JS, and EJS weighting schemes are combined with the CEP, CNP, ReCNP, and ReWNP pruning algorithms.

As the BlCl and CoCl steps are consecutive and optional sub-tasks of blocking, there are three different groups of blocking methods that are analyzed:

- BlBu only: StBl or ACl for creating the blocks.
- BlBu and BlCl: apply BlFi on the blocks.
- BlBu, BlCl, and CoCl: apply MeBl on the refined blocks.

For the final comparison of the blocking methods, the α values of the blocking methods with the best parameters are examined. Hereby, the three different groups of performed sub-tasks are observed for the five subsets and with the two different BlBu methods. Hence, the overall best α value determines the final BlBu method, the executed sub-tasks, the parameters of the methods, and the subset of attributes.

4.1.5 Learning Matching Rules with Silk

The baseline rule is created with a optimized threshold for a scaled Levenshtein similarity measure on the stripped URIs. In addition, four different subsets with special characteristics are created for DBpedia and YAGO. The aim is to learn more robust matching rules with subsets that contain examples that are difficult to detect. Those subsets are then imported into Silk to learn matching rules automatically. Figure 4.2 illustrate the easy and hard cases for the matching rule based on the URIs. The entities having links that have a low similarity measure (green bars on the left) as well as the entities that do not have links having a high similarity (red bars on the right) are difficult to learn for a matching rule. On the contrary, matches that have a high similarity and non-matches that have a low similarity are easy to distinguish and are covered by the basic baseline rule.

Figure 4.1 indicates that the easy cases are covered when picking entities randomly due to the high counts of those entities. The four subsets for DBpedia and YAGO consist of 1000 different event instances: 500 matches and 500 non-matches. Those matches and non-matches then consist of entities with different characteristics.

- Random matches and random non-matches (RR)
- Random matches and high similarity of non-matches (RH)
- Low similarity of matches and random non-matches (LR)
- Low similarity of matches and high similarity of non-matches (LH)

The full DBpedia and YAGO XML files are used to create the subsets. Entities having links are collected and the Levenshtein edit distance is calculated between the stripped URIs. For obtaining the matches having a low similarity, the entities are sorted based on the Levenshtein edit distance and the 500 matches with the highest distance scores are returned. The random matches are received using a built-in random function on the set of entities having links.

As Standard Blocking is used to speed-up the runtime of the script, those blocks are also used to receive the assumed non-matches having a high similarity score.

The underlying idea is that entities sharing at least one token have a higher similarity score than entities that do not share a token at all. Therefore, non-linked entities from the same block but different KGs are collected and ordered by their Levenshtein edit distance. The 500 non-matches with the lowest edit distance are returned for high similarity pairs of non-matches. Random non-matches are picked by chance from all entities that are not linked to each other.

For each of the four cases, the script returns subsets of the full XML datasets in *N-Triples* format (.nt) with only the required entities as well as an XML file that specifies the relation (match or non-match) between the entities. The three files are imported into Silk to learn the matching rules. Due to the random initial population of the *GenLink* genetic programming algorithm, three runs are executed for determining the matching rule that maximizes the F-measure on the gold standard.

4.2 Results and Discussion

This chapter presents the results for the blocking methods in Chapter 4.2.1 and the outcomes of the matching rules in Chapter 4.2.2.

4.2.1 Blocking Methods

Without blocking, the total number of comparing the $56{,}725$ DBpedia entities with the $417{,}113$ YAGO entities is $23{,}660{,}734{,}925$ ($56{,}725 \times 417{,}113$). As the *Java Web Data Integration Framework* keeps all candidate pairs in-memory, the server is not able to process this large number of comparisons with the provided framework. The duration for calculating a basic Levenshtein similarity measure on the URI for two sample files with $10{,}000$ entries each takes 686.427 ms. Hence, one out of the $10{,}000^2$ comparisons approximately takes 0.007 ms. With this estimation, the full 2.37E+10 comparisons would take 46 hours. A more complex matching rule would even increase this required resolution time.

The results of the blocking sub-tasks are presented in the following. First, the results of testing different parameters for the blocking methods are discussed. Afterwards, the results with the best parameters for each blocking method and the performed sub-tasks are compared for determining the final blocking method. All PC, RR, and α values are rounded to three digits after the decimal point.

BlBu

As StBl is parameter-free, the results only vary when changing the input data. Table 4.2 shows these results of StBl for the five subsets (s). In addition to the PC,

PQ, RR, and α effectiveness measures, the number of blocks and the total number of required comparisons are presented.

s	#blocks	total comparisons	PC	PQ	RR	α
1	**19517**	**1.34E+09**	**0.994**	**3.70E-06**	**0.936**	**0.9301**
2	19550	1.35E+09	0.994	3.69E-06	0.935	0.9299
3	25694	7.00E+10	1.000	7.13E-08	-1.980	-1.9801
4	25857	7.00E+10	1.000	7.13E-08	-1.980	-1.9803
5	32790	7.55E+10	1.000	6.60E-08	-2.205	-2.2048

Table 4.2: StBl Results for the five Subsets (s)

For Table 4.2, the α value is rounded to four digits after the decimal point to highlight the deviation of the first two lines that would not be visible if rounding to three digits only. The bold row highlights the subset that maximizes α.

The more attributes the BlBu method receives as input, the higher the number of blocks. This behavior is expected as different attributes contain diverse values that lead to an increasing number of blocks and a larger number of comparisons. In addition, more attributes increase the PC but decrease the PQ and RR, which also becomes negative in some cases. A negative RR score indicates that more comparisons than the brute-force approach are executed. This occurs due to the redundant comparisons when comparing the same two entities multiple times for each block that they are assigned to.

When comparing the results for the subsets 1 and 2, it is interesting to see that adding the label attribute to the URI does not increase α. This is because the values of the stripped URI and the English labels are very similar. Therefore, this attribute could be discarded as it provides mostly redundant information.

On the contrary, using two or more attributes with diverse values such as subsets 3, 4, or 5, leads to a maximal PC score as all matching entities are compared in at least one block. Nevertheless, due to the many redundant comparisons, the RR and PQ are very low and even negative for the RR scores.

For subsets 1 and 2, the RR scores are already very high even though many redundant comparisons have to be execute when running BlBu without any cleaning methods. This indicates that the URI and label attributes are a good duplicate indicator for the used datasets and simultaneously help to discard many superfluous comparisons. Nevertheless, the low PQ scores imply the high number of redundant comparisons that occur when running BlBu methods solely without BlCl and CoCl. This is evident for the subsets 3, 4, and 5 that execute around twice as many comparisons as the naïve brute-force approach.

s	representation model	#blocks	total comparisons	PC	PQ	RR	α
1	**Term Vector**	**19517**	**1.34E+09**	**0.994**	**3.70E-06**	**0.936**	**0.930**
1	**Character Bigrams**	**19517**	**1.34E+09**	**0.994**	**3.70E-06**	**0.936**	**0.930**
1	**Character Trigrams**	**19517**	**1.34E+09**	**0.994**	**3.70E-06**	**0.936**	**0.930**
1	**Token Bigram Graphs**	**19517**	**1.34E+09**	**0.994**	**3.70E-06**	**0.936**	**0.930**
1	**Token Trigram Graphs**	**19517**	**1.34E+09**	**0.994**	**3.70E-06**	**0.936**	**0.930**
2	**Term Vector**	**19550**	**1.35E+09**	**0.994**	**3.69E-06**	**0.935**	**0.930**
2	Character Bigrams	39067	2.67E+09	0.994	1.86E-06	0.872	0.867
2	Character Trigrams	39067	2.67E+09	0.994	1.86E-06	0.872	0.867
2	**Token Bigram Graphs**	**19550**	**1.35E+09**	**0.994**	**3.69E-06**	**0.935**	**0.930**
2	**Token Trigram Graphs**	**19550**	**1.35E+09**	**0.994**	**3.69E-06**	**0.935**	**0.930**
3	**Term Vector**	**25694**	**7.00E+10**	**1.000**	**7.13E-08**	**-1.980**	**-1.980**
3	**Character Bigrams**	**25694**	**7.00E+10**	**1.000**	**7.13E-08**	**-1.980**	**-1.980**
3	**Character Trigrams**	**25694**	**7.00E+10**	**1.000**	**7.13E-08**	**-1.980**	**-1.980**
3	**Token Bigram Graphs**	**25694**	**7.00E+10**	**1.000**	**7.13E-08**	**-1.980**	**-1.980**
3	Token Trigram Graphs	40820	6.93E+10	1.000	7.19E-08	-2.002	-2.002
4	**Term Vector**	**25857**	**7.00E+10**	**1.000**	**7.13E-08**	**-1.980**	**-1.980**
4	**Character Bigrams**	**25857**	**7.00E+10**	**1.000**	**7.13E-08**	**-1.980**	**-1.980**
4	Character Trigrams	45244	7.13E+10	1.000	6.99E-08	-2.036	-2.036
4	**Token Bigram Graphs**	**25857**	**7.00E+10**	**1.000**	**7.13E-08**	**-1.980**	**-1.980**
4	Token Trigram Graphs	40853	6.93E+10	1.000	7.19E-08	-2.001	-2.001
5	**Term Vector**	**32447**	**1.87E+10**	**0.995**	**2.65E-07**	**0.119**	**0.118**
5	Character Bigrams	38174	7.54E+10	1.000	6.61E-08	-2.208	-2.208
5	Character Trigrams	55051	7.70E+10	1.000	6.48E-08	-2.266	-2.266
5	Token Bigram Graphs	37343	7.53E+10	1.000	6.63E-08	-2.204	-2.204
5	Token Trigram Graphs	52154	7.46E+10	1.000	6.68E-08	-2.227	-2.227

Table 4.3: Testing different Representation Models for ACl and the five Subsets (s)

Table 4.3 presents the results for all five subsets and the five different representation models for ACl. Bold rows indicate the best representation model for the subset regarding α. All results are very similar to StBl due to the equal attribute names. Adding the label attribute to the URIs does not increase α and even decreases the PQ and RR scores when using the *Character Bigrams* or *Trigrams* representation models. The higher number of attributes in subsets 3, 4, and 5 leads to more diverse values and thus decreases the RR scores that are even negative for most representation models. Main difference to the StBl method is that the Term Vector model works very well for dataset 5 leading to positive RR and α scores

while trading some PC for it. As the Term Vector representation model is among the best models for all five subsets, it is considered as best overall model for ACl.

Both BlBu methods shows that when using less but meaningful attributes, some PC is traded for a much higher RR and therefore a lower number of executed comparisons. This effect is also used for the further subsequent block and comparison cleaning sub-tasks. In addition, BlBu without any block or comparison cleaning methods is mostly not applicable with two or more diverse attributes as this would require more comparisons than the brute-force approach. Only the Term Vector representation model for ACl works well for the full dataset and is slightly better than the brute-force approach. All in all, more attributes with diverse values lead to a larger number of blocks and thus more (potentially redundant and superfluous) comparisons. In addition, some PC can be traded for higher PQ and RR. This contradicting relation of PC with PQ and RR is also used for the block and comparison cleaning methods.

Block-refinement: BlFi

As described in Chapter 4.1.4, 20 different parameters for the BlFi ratio are tested for each subset and each underyling BlBu method. Hence, 200 different α scores are compared in total to determine the best BlFi ratio for each subset and BlBu method ($20 \times 5 \times 2$).

Table 4.4 presents the α scores for each subset with StBl as BlBu method. Figure 4.3 illustrates those results with the ratios on the x-axis and the α scores on the y-axis. The lines for subsets 1 and 2 as well as the lines for subsets 3 and 4 are mostly on top of each other as they are so similar in nature. Table 4.5 and Figure 4.4 show the same results when using ACl with Term Vector as BlBu method. The full results including PC, PQ, and RR for StBl and ACl and the five subsets are provided in the Appendices B.1 to B.10.

The first column of the Tables 4.4 and 4.5 show the 20 different values for the ratio of the BlFi method from 0.05 to 1 with steps of size 0.05. All other columns represent the respective subsets. Therefore, the ratio with the highest α score is determined for each subset, which is indicated by the bold values.

All BlFi ratios that maximize α are between 0.4 and 0.55. This indicates that α is maximized when removing the entities from roughly half of their assigned blocks with the highest cardinalities.

A ratio of 1.0 indicates that all block assignments are retained so the results are the same as executing no BlCl at all. For ratios smaller or equal to 0.8, the α scores for all subsets and both BlBu methods become positive. For the dataset at hand, this implies that blocking saves comparisons with regards to the brute-force approach when removing the entity assignments from at least 20% of the largest

r	1	2	3	4	5
0.05	1.99E-04	1.99E-04	0.030	0.043	0.729
0.1	0.175	0.174	0.929	0.921	0.888
0.15	0.542	0.539	0.957	0.950	0.935
0.2	0.923	0.917	0.969	0.969	0.954
0.25	0.978	0.974	0.976	0.975	0.972
0.3	0.979	0.977	0.984	0.984	0.982
0.35	0.979	0.978	0.985	0.986	0.986
0.4	0.981	0.981	**0.986**	**0.987**	0.989
0.45	0.980	0.980	0.985	0.985	0.990
0.5	**0.985**	**0.985**	0.985	0.985	**0.991**
0.55	0.984	0.984	0.984	0.984	0.988
0.6	0.983	0.983	0.980	0.980	0.980
0.65	0.983	0.983	0.954	0.954	0.961
0.7	0.982	0.982	0.906	0.906	0.926
0.75	0.979	0.979	0.648	0.648	0.768
0.8	0.977	0.976	0.399	0.399	0.583
0.85	0.970	0.970	-0.029	-0.028	0.203
0.9	0.950	0.950	-0.644	-0.643	-0.350
0.95	0.933	0.933	-1.165	-1.164	-1.047
1	0.930	0.930	-1.980	-1.980	-2.205

Table 4.4: α for different BlFi Ratios and Subsets with StBl

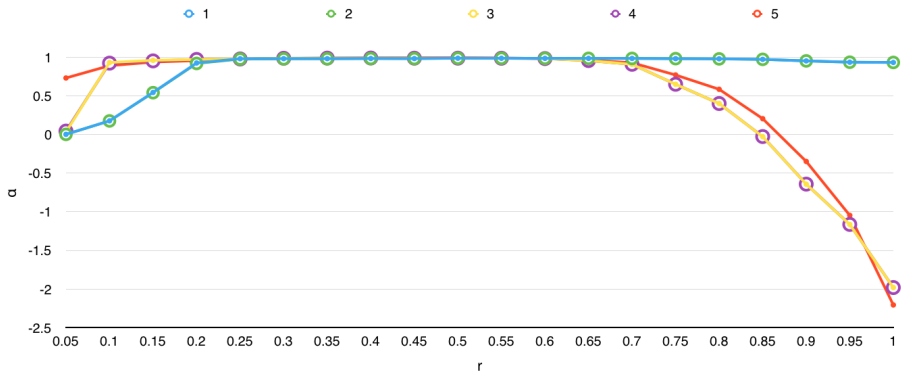


Figure 4.3: BlFi Ratio Parameter Tuning for StBl using the five Subsets

r	1	2	3	4	5
0.05	1.99E-04	1.99E-04	0.030	0.043	0.332
0.1	0.175	0.174	0.929	0.921	0.855
0.15	0.542	0.539	0.957	0.950	0.903
0.2	0.923	0.917	0.969	0.969	0.947
0.25	0.978	0.974	0.976	0.975	0.961
0.3	0.979	0.977	0.984	0.984	0.967
0.35	0.979	0.978	0.985	0.986	0.970
0.4	0.981	0.981	**0.986**	**0.987**	0.977
0.45	0.980	0.980	0.985	0.985	0.978
0.5	**0.985**	**0.985**	0.985	0.985	0.978
0.55	0.984	0.984	0.984	0.984	**0.980**
0.6	0.983	0.983	0.980	0.980	0.976
0.65	0.983	0.983	0.954	0.954	0.972
0.7	0.982	0.982	0.906	0.906	0.961
0.75	0.979	0.979	0.648	0.648	0.932
0.8	0.977	0.976	0.399	0.399	0.884
0.85	0.970	0.970	-0.029	-0.028	0.759
0.9	0.950	0.950	-0.644	-0.643	0.553
0.95	0.933	0.933	-1.165	-1.164	0.340
1	0.930	0.930	-1.980	-1.980	0.118

Table 4.5: α for different BlFi Ratios and Subsets with ACl

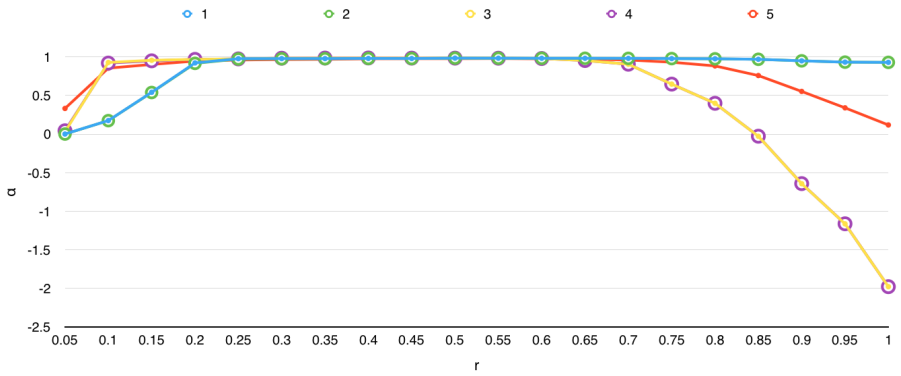


Figure 4.4: BlFi Ratio Parameter Tuning for ACl using the five Subsets

block in all cases: the number of redundant comparisons becomes smaller than the number of saved superfluous comparisons of the entities that do not share blocks.

The best BlFi ratios (r) for the different subsets (s) and BlBu methods are summarized in Table 4.6. TV stands for the Term Vector representation model of ACl.

BlBu	s	r	#blocks	total comparisons	PC	PQ	RR	α
StBl	1	0.5	19333	7.54E+07	0.988	6.54E-05	0.996	0.985
StBl	2	0.5	19366	7.53E+07	0.988	6.54E-05	0.996	0.985
StBl	3	0.4	25620	5.89E+07	0.989	8.38E-05	0.997	0.986
StBl	4	0.4	25786	5.87E+07	0.990	8.41E-05	0.997	0.987
StBl	**5**	**0.5**	**32775**	**2.10E+08**	**1.000**	**2.38E-05**	**0.991**	**0.991**
ACl (TV)	1	0.5	19333	7.54E+07	0.988	6.54E-05	0.996	0.985
ACl (TV)	2	0.5	19366	7.53E+07	0.988	6.54E-05	0.996	0.985
ACl (TV)	3	0.4	25620	5.89E+07	0.989	8.38E-05	0.997	0.986
ACl (TV)	4	0.4	25786	5.87E+07	0.990	8.41E-05	0.997	0.987
ACl (TV)	5	0.55	32426	1.26E+08	0.986	3.90E-05	0.994	0.980

Table 4.6: Best Ratios for BlFi for the different BlBu Methods and the Subsets

The α scores for all subsets and BlBu methods indicate that blocking becomes very efficient when using BlFi with an optimized ratio. A reduction ratio of more than 99% for all subsets and BlBu methods implies a high increase in efficiency in regards to the brute-force approach. In order to reach such a high RR score, almost all methods trade some PC. Only StBl with subset 5 retains the perfect 1.0 score for PC but thus has the lowest PQ score of all best methods that perform BlFi. The overall low PQ scores originate from the still high number of redundant and superfluous comparisons that arise when building the blocks.

StBl with a ratio of 0.5 for subset 5 with all attributes provides the best α score of all BlBu methods with BlFi and the optimized parameter. This method indeed has the lowest RR of 0.991 but therefore scores the best PC where all examples of the gold standard appear in at least on block even after removing the entities from 50% of the block assignments.

Comparison-refinement: MeBl

Table 4.7 presents the best MeBl weighting schemes (WS) for each subset (s) and the two different BlBu methods. ReWNP is used as pruning algorithm as it scores the best α values for all configurations. The complete results for the different weighting schemes and pruning algorithms are provided in Appendices B.11 to

B.20. For Table 4.7 the PC, RR, and α scores are rounded to five digits after the decimal point to show the marginal deviations of the values. Term Vector (TV) is the representation model for all ACl BlBu methods.

BlBu	BlFi ratio	s	WS	#blocks	total comparisons	PC	PQ	RR	α
StBl	**0.50**	**1**	**CBS**	**56525**	**2.75E+07**	**0.97514**	**1.77E-04**	**0.99864**	**0.97382**
StBl	0.50	2	CBS	56526	2.77E+07	0.97514	1.76E-04	0.99863	0.97381
StBl	0.40	3	ARCS	56707	1.71E+07	0.96652	2.83E-04	0.99924	0.96578
StBl	0.40	4	ARCS	56707	1.67E+07	0.96652	2.90E-04	0.99918	0.96573
StBl	0.50	5	ARCS	56717	3.98E+07	0.96612	1.21E-04	0.99831	0.96448
ACl (TV)	**0.50**	**1**	**CBS**	**56525**	**2.75E+07**	**0.97514**	**1.77E-04**	**0.99864**	**0.97382**
ACl (TV)	0.50	2	CBS	56526	2.77E+07	0.97514	1.76E-04	0.99863	0.97381
ACl (TV)	0.40	3	ARCS	56707	1.71E+07	0.96652	2.83E-04	0.99924	0.96578
ACl (TV)	0.40	4	ARCS	56707	1.67E+07	0.96652	2.90E-04	0.99918	0.96573
ACl (TV)	0.55	5	ARCS	56717	2.29E+07	0.96612	2.11E-04	0.99890	0.96505

Table 4.7: Best MeBl Weighting Schemes (WS) and for the five Subsets and the two different BlBu Methods using ReWNP as Pruning Algorithm

As MeBl removes all redundant comparisons by definition, the PQ scores increase by one magnitude in comparison to the best BlFi methods. Nevertheless, PQ remains quite small due to the superfluous candidate pairs that cannot be prevented when comparing large datasets. The high RR and strongly increased PQ scores prove the purpose of comparison cleaning to trade some PC for gaining PQ and RR. The RR scores of approximately 99.9% for all subsets and BlBu methods indicate the boost in efficiency in comparison to the brute-force approach even though the PQ values remain small.

The final scores for the subsets 1 to 4 are the same for StBl and ACl. With weighting and pruning block assignments, MeBl results in the same number of blocks and comparisons as well as identical values for PC, PQ, RR, and α, no matter which BlBu method is used. Nevertheless, using ACl as BlBu method for subset 5 leads to slightly better results for PQ and RR when applying BlFi and MeBl in regards to StBl. ACl thus works better for many diverse attributes when running block and comparison cleaning methods as well.

The CBS weighting scheme is used for both BlBu methods with the subsets 1 and 2. This implies that CBS works well for datasets having only one or multiple very similar attributes. ARCS is used as weighting schemes for all other subsets and therefore works better when using many diverse attributes.

The ReWNP is determined to be the best pruning algorithm for all subsets

and both BlBu methods. Hence, this algorithm is considered to work well for the dataset at hand that is extracted from the Semantic Web.

Overall Best Parameter

Table 4.8 shows the blocking methods with the overall best parameters that maximize α. The results are grouped based on the executed sub-tasks indicated with the double horizontal lines. The first group only builds the blocks but does not apply any block or comparison cleaning sub-tasks. The second group executes BlFi as block cleaning sub-tasks after BlBu. MeBl is performed as comparison cleaning sub-task after BlBu and BlCl in the third group. As two different BlBu methods and five subsets are analyzed, ten results are presented for each group. The method that overall maximizes α will be determined as final blocking method.

The PC, PQ, and RR scores for the three different groups highlight the purpose of block and comparison cleaning methods. When using two or more diverse attributes, simple BlBu methods solely do not increase the performance of identity resolution. The performance is even worse as the number of additional redundant comparisons is higher than the number of saved superfluous comparisons.

Nevertheless, when executing block cleaning methods after building the blocks, some PC is traded for PQ and RR. For the dataset at hand, the PQ score is increased by up to three magnitudes with an optimized ratio for BlFi. In addition, the RR value becomes positive and is even higher than 99% in all cases. This rise in performance is also indicated with the decreased number of total comparisons. The number of blocks is slightly reduced due to blocks that contain only one or none entity after removing the block assignments. Other BlCl methods that operate on the block-level would remove more blocks but retain all entity assignments within those blocks.

With MeBl all redundant comparisons are discarded due to the creation of an undirected graph. The weighting and pruning of edges additionally trades some PC for PQ and RR for removing further superfluous comparisons. The PQ score increases by another magnitude and the RR rise. In return, the PC decreases up to 0.034.

The overall blocking method that maximizes α is StBl for building the blocks and BlFi for block cleaning the blocks with a ratio of 0.5. This method retains the perfect PC score of 1.0 even though half of the entity assignments to blocks are removed. For subset 5, this block cleaning on entity-level improves the RR score from -220.5% for StBl without BlFi to 99.1%.

BlBu	BlCl	r	CoCl	s	#blocks	total comparisons	PC	PQ	RR	α
StBl	-	-	-	1	19517	1.34E+09	0.994	3.70E-06	0.936	0.930
StBl	-	-	-	2	19550	1.35E+09	0.994	3.69E-06	0.935	0.930
StBl	-	-	-	3	25694	7.00E+10	1.000	7.13E-08	-1.980	-1.980
StBl	-	-	-	4	25857	7.00E+10	1.000	7.13E-08	-1.980	-1.980
StBl	-	-	-	5	32790	7.55E+10	1.000	6.60E-08	-2.205	-2.205
ACl (TV)	-	-	-	1	19517	1.34E+09	0.994	3.70E-06	0.936	0.930
ACl (TV)	-	-	-	2	19550	1.35E+09	0.994	3.69E-06	0.935	0.930
ACl (TV)	-	-	-	3	25694	7.00E+10	1.000	7.13E-08	-1.980	-1.980
ACl (TV)	-	-	-	4	25857	7.00E+10	1.000	7.13E-08	-1.980	-1.980
ACl (TV)	-	-	-	5	32447	1.87E+10	0.995	2.65E-07	0.119	0.118
StBl	BlFi	0.50	-	1	19333	7.54E+07	0.988	6.54E-05	0.996	0.985
StBl	BlFi	0.50	-	2	19366	7.53E+07	0.988	6.54E-05	0.996	0.985
StBl	BlFi	0.40	-	3	25620	5.89E+07	0.989	8.38E-05	0.997	0.986
StBl	BlFi	0.40	-	4	25786	5.87E+07	0.990	8.41E-05	0.997	0.987
StBl	**BlFi**	**0.50**	**-**	**5**	**32775**	**2.10E+08**	**1.000**	**2.38E-05**	**0.991**	**0.991**
ACl (TV)	BlFi	0.50	-	1	19333	7.54E+07	0.988	6.54E-05	0.996	0.985
ACl (TV)	BlFi	0.50	-	2	19366	7.53E+07	0.988	6.54E-05	0.996	0.985
ACl (TV)	BlFi	0.40	-	3	25620	5.89E+07	0.989	8.38E-05	0.997	0.986
ACl (TV)	BlFi	0.40	-	4	25786	5.87E+07	0.990	8.41E-05	0.997	0.987
ACl (TV)	BlFi	0.55	-	5	32426	1.26E+08	0.986	3.90E-05	0.994	0.980
StBl	BlFi	0.50	MeBl	1	56525	2.75E+07	0.975	1.77E-04	0.999	0.974
StBl	BlFi	0.50	MeBl	2	56526	2.77E+07	0.975	1.76E-04	0.999	0.974
StBl	BlFi	0.40	MeBl	3	56707	1.71E+07	0.967	2.83E-04	0.999	0.966
StBl	BlFi	0.40	MeBl	4	56707	1.67E+07	0.967	2.90E-04	0.999	0.966
StBl	BlFi	0.50	MeBl	5	56717	3.98E+07	0.966	1.21E-04	0.998	0.964
ACl (TV)	BlFi	0.50	MeBl	1	56525	2.75E+07	0.975	1.77E-04	0.999	0.974
ACl (TV)	BlFi	0.50	MeBl	2	56526	2.77E+07	0.975	1.76E-04	0.999	0.974
ACl (TV)	BlFi	0.40	MeBl	3	56707	1.71E+07	0.967	2.83E-04	0.999	0.966
ACl (TV)	BlFi	0.40	MeBl	4	56707	1.67E+07	0.967	2.90E-04	0.999	0.966
ACl (TV)	BlFi	0.55	MeBl	5	56717	4.75E+07	0.969	1.02E-04	0.998	0.967

Table 4.8: Blocking Methods with Overall Best Parameter

Runtime of the Best Blocking Methods

The runtime of the best overall methods from Table 4.8 is listed in Table 4.9. The Overhead Time (OTime) of the blocking methods can be determined using the

sum of the BlBu, BlCl, and CoCl times. Using the approximated duration for one comparison of 0.007 ms, the estimated Resolution Time (RTime) is calculated with:

$$RTime = OTime + c * 0.007ms \tag{4.1}$$

where c denotes the total number of comparisons of the blocking method as listed in Table 4.8. The RTime is compared to the brute-force approach with estimated 46 hours to calculate the saved time when using the respective blocking method. Figure 4.5 illustrates the estimated RTime (blue bars) and highlights the saved time (green bars) of the blocking methods in regards to the brute-force approach. The red line indicates the duration that is required by the brute-force approach.

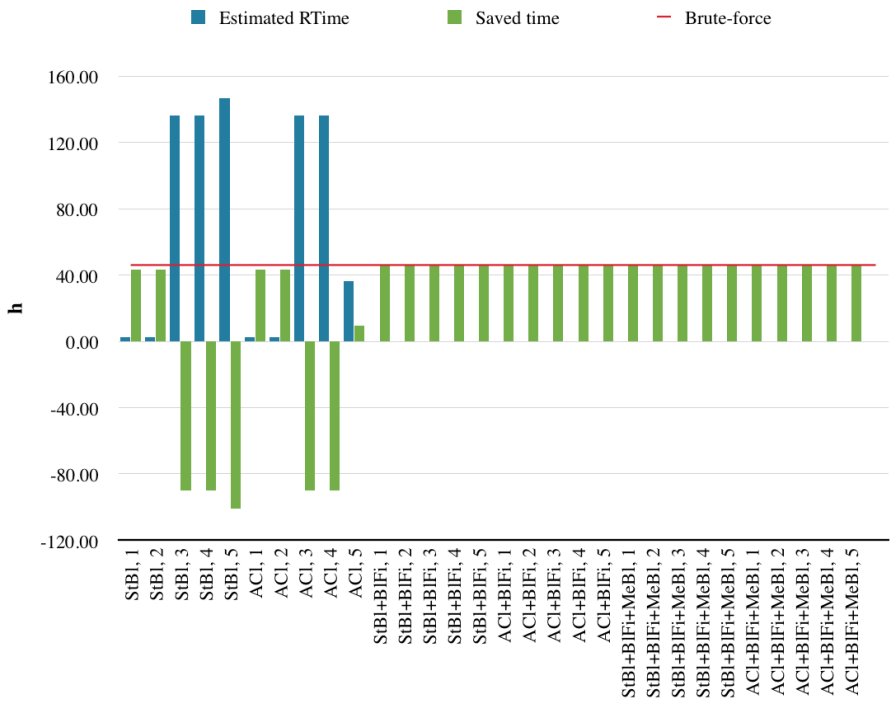


Figure 4.5: Runtime of Blocking Methods in Comparison to the Brute-force Approach

All blocking methods that take longer than those 46 hours have a negative value for the *saved time*. Figure 4.5 clearly shows that as soon as block cleaning methods are performed, the required RTime decreases remarkably. In addition to the RR, that only considers the number of comparisons, the RTime also takes the time

BlBu	BlCl	r	CoCl	s	BlBu time	BlCl time	CoCl time	estimated RTime
StBl	-	-	-	1	4224.8	-	-	9.38E+06
StBl	-	-	-	2	7402.4	-	-	9.43E+06
StBl	-	-	-	3	11033.4	-	-	4.90E+08
StBl	-	-	-	4	15450.9	-	-	4.90E+08
StBl	-	-	-	5	22379.8	-	-	5.29E+08
ACl (TV)	-	-	-	1	4740.1	-	-	9.38E+06
ACl (TV)	-	-	-	2	8797.8	-	-	9.43E+06
ACl (TV)	-	-	-	3	12391.6	-	-	4.90E+08
ACl (TV)	-	-	-	4	16229.1	-	-	4.90E+08
ACl (TV)	-	-	-	5	24054.0	-	-	1.31E+08
StBl	BlFi	0.50	-	1	4224.8	56.3	-	5.32E+05
StBl	BlFi	0.50	-	2	7402.4	56.1	-	5.35E+05
StBl	BlFi	0.40	-	3	11033.4	94.4	-	4.23E+05
StBl	BlFi	0.40	-	4	15450.9	112.5	-	4.26E+05
StBl	BlFi	0.50	-	5	22379.8	183.6	-	1.49E+06
ACl (TV)	BlFi	0.50	-	1	4740.1	51.9	-	5.33E+05
ACl (TV)	BlFi	0.50	-	2	8797.8	58.4	-	5.36E+05
ACl (TV)	BlFi	0.40	-	3	12391.6	90.0	-	4.25E+05
ACl (TV)	BlFi	0.40	-	4	16229.1	91.5	-	4.27E+05
ACl (TV)	BlFi	0.55	-	5	24054.0	144.8	-	9.06E+05
StBl	BlFi	0.50	MeBl	1	4224.8	56.3	48886.2	2.46E+05
StBl	BlFi	0.50	MeBl	2	7402.4	56.1	47981.1	2.49E+05
StBl	BlFi	0.40	MeBl	3	11033.4	94.4	135216.5	2.66E+05
StBl	BlFi	0.40	MeBl	4	15450.9	112.5	142825.8	2.75E+05
StBl	BlFi	0.50	MeBl	5	22379.8	183.6	373799.1	6.75E+05
ACl (TV)	BlFi	0.50	MeBl	1	4740.1	51.9	55313.8	2.53E+05
ACl (TV)	BlFi	0.50	MeBl	2	8797.8	58.4	56937.5	2.60E+05
ACl (TV)	BlFi	0.40	MeBl	3	12391.6	90.0	169378.4	3.01E+05
ACl (TV)	BlFi	0.40	MeBl	4	16229.1	91.5	171479.5	3.04E+05
ACl (TV)	BlFi	0.55	MeBl	5	24054.0	144.8	396961.5	7.53E+05

Table 4.9: Runtime of Blocking Methods with Overall Best Parameter

to build and clean the block into account. This is an important factor as the duration of the blocking sub-tasks with the additional remaining comparisons might take longer than executing the brute-force approach. Nevertheless, this experiment certainly shows the efficiency of blocking when applying cleaning sub-tasks.

Final Evaluation of Blocking Methods

The presented efficiency scores and runtimes of the different blocking methods show that when using multiple diverse attributes, block and optionally comparison cleaning methods are essential. Lazy blocking methods only become efficient for large datasets when applying at least block-refinement methods such as BlFi. The optimized ratios for BlFi are in between 0.4 and 0.55 for both BlBu methods and all five subsets. This implies that around half of the entity assignments to the largest blocks are dispensable and can be pruned to significantly increase the performance with trading only a small amount of PC.

The required time for building and cleaning the blocks with the additional duration of comparing the remaining candidate pairs is much lower than executing the brute-force approach. Hence, blocking methods are crucial when comparing or integrating large datasets.

For each consecutive cleaning step, some PC is traded for PQ and RR. Depending on the goal of the application and the use case at hand, the importance of PC in comparison to RR has to be determined. The more PC can be discarded, the higher the potential increase in RR.

4.2.2 Matching Rules

First, an optimized baseline rule is created using the gold standard file. Then Silk is used to learn rules for subsets of the gold standard that have special characteristics. Those learned rules are then compared to the baseline rule to determine the final matching rule. Goal of the matching rules is to maximize the F-measure.

Baseline

In order to calculate the optimal threshold of the scaled Levenshtein baseline rule, the precision, recall, and F-measure values are calculated for $t \in [0, 1]$ in steps of size 0.1. The results are evaluated on the created gold standard and are listed in Table 4.10a. Figure 4.6 illustrates these results for $t \in [0, 1]$. Table 4.10a indicates that the maximum F-measure is between 0.9 and 1.0. Therefore, a second run with $t \in [0.9, 1]$ in steps of size 0.01 is performed to obtain a more exact threshold. These results are listed in Table 4.10b. The lowest threshold that maximizes the F-measure is **0.96**, which is determined as threshold for the baseline rule using the scaled Levenshtein similarity measure on the stripped URIs.

t	Precision	Recall	F-measure
1	1.000	0.970	0.985
0.9	0.907	0.970	0.937
0.8	0.744	0.990	0.850
0.7	0.398	0.990	0.567
0.6	0.086	0.990	0.159
0.5	0.031	1.000	0.060
0.4	0.018	1.000	0.035
0.3	0.016	1.000	0.031
0.2	0.012	1.000	0.025
0.1	0.010	1.000	0.020
0	0.010	1.000	0.020

(a) For $t \in [0, 1]$ with steps of size 0.1

t	Precision	Recall	F-measure
1.00	1.000	0.970	0.985
0.99	1.000	0.970	0.985
0.98	1.000	0.970	0.985
0.97	1.000	0.970	0.985
0.96	**1.000**	**0.970**	**0.985**
0.95	0.960	0.970	0.965
0.94	0.960	0.970	0.965
0.93	0.960	0.970	0.965
0.92	0.960	0.970	0.965
0.91	0.960	0.970	0.965
0.90	0.907	0.970	0.937

(b) For $t \in [0.9, 1]$ with steps of size 0.01

Table 4.10: Matching Rule Baseline for Scaled Levenshtein on the Stripped URI

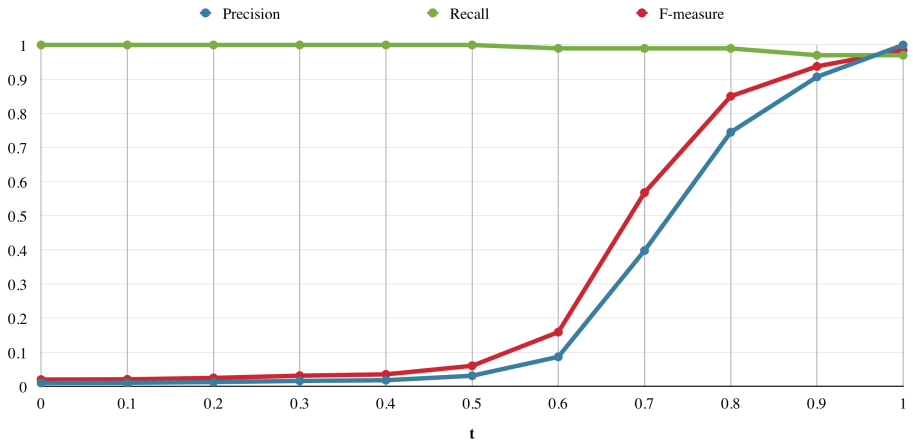


Figure 4.6: Precision, Recall, and F-Measure for the Baseline Matching Rule with $t \in [0, 1]$

SILK Learned Rules

For learning matching rules with Silk, the four subsets with special characteristics are used as input files. The best rule for each subset is shown in Figures 4.7 to 4.10. For matching two datasets in Silk, the source and target schema have to be defined. As the DBpedia and YAGO input files are both in the same mediated

schema structure as defined in Chapter 3, the order of this definition does not make a difference. For the experiments, YAGO is used as source and DBpedia as target path. Hence, the YAGO entities are mapped to the DBpedia file.

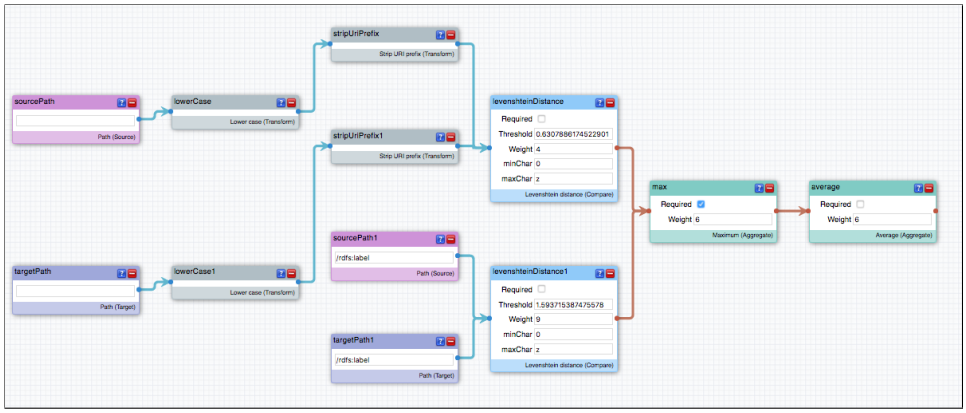


Figure 4.7: Learned Rule for Random Matches and Random Non-matches (RR)

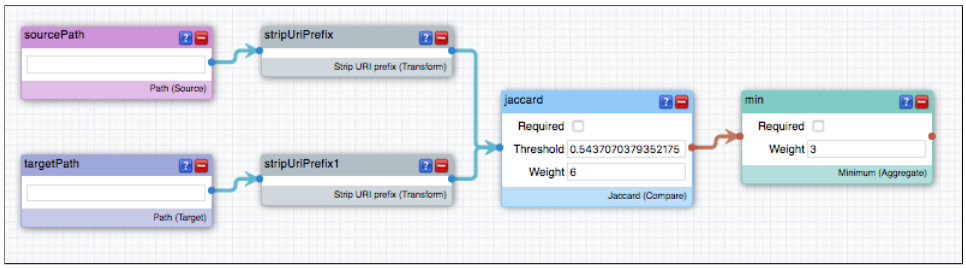


Figure 4.8: Learned Rule for Random Matches and High Similarity of Non-matches (RH)

When using random matches and random non-matches (RR) as input files, the learned rule that performs best on the samples is shown in Figure 4.7. The empty source and target paths indicate the URI of the entities that is used as unique identifier. The Levenshtein comparator with a threshold of 0.63 is used for the lowercase and stripped URIs. As the absolute Levenshtein distance is used in Silk this edit distance of less than 1 corresponds to perfect equality of the resulting strings. In addition, the English labels of the entries are compared using a Levenshtein distance with a threshold of 1.59. Therefore, one character insertion, deletion, or substitution is permitted. The two results are then combined using the maximum aggregator. This aggregator evaluates on the highest confidence of the results to de-

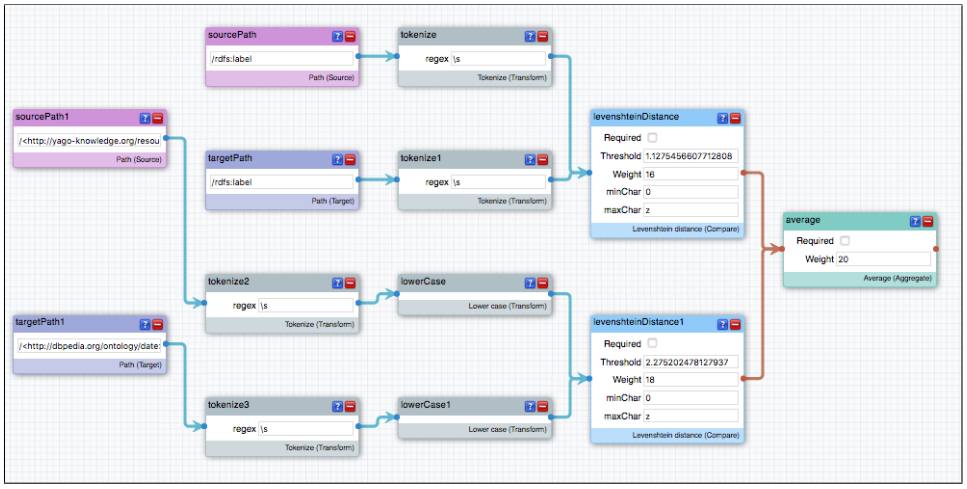


Figure 4.9: Learned Rule for Low Similarity of Matches and Random Non-matches (LR)

cide whether the entries should be regarded as duplicates. Hence, the learned RR rule is similar to the baseline rule but also takes the English labels into account.

Figure 4.8 shows the learned rule for the random matches and the non-matches having a high similarity (RH). Hence, all input entities are very similar regarding the URI. Nevertheless, the best learned rule only considers the URIs, stripes the prefix and uses the Jaccard distance coefficient with a threshold of 0.54 to determine the duplicates. Note that Silk always uses the distance and not the similarity measures for comparisons. As those two measures are interchangeable using the $sim(x, y) = 1 - dist(x, y)$ formula as defined in equation 2.4, this difference of Silk and the Web Date Integration framework are not problematic. When comparing two entities with the Jaccard similarity coefficient instead of the distance coefficient, the threshold has to be converted using equation 2.4 as well: A maximum threshold of 0.54 for the distance coefficient becomes a minimum threshold of 0.46 $(1-0.54)$ for the similarity coefficient. As the input entities are very similar regarding the stripped URIs and the RH rule only considers this property to compare the entities, the learned rule declares duplicates less strict than the baseline rule. This characteristic will increase the number of declared duplicates, which are potentially incorrect. Hence, the RH rule will probably find more true duplicates but simultaneously declare more non-duplicates as matches.

The learned rule for matches having a low similarity score and random non-matches (LR) is shown in Figure 4.9. The rule tokenizes the English labels and compares them with an absolute Levenshtein distance measure having a maximum

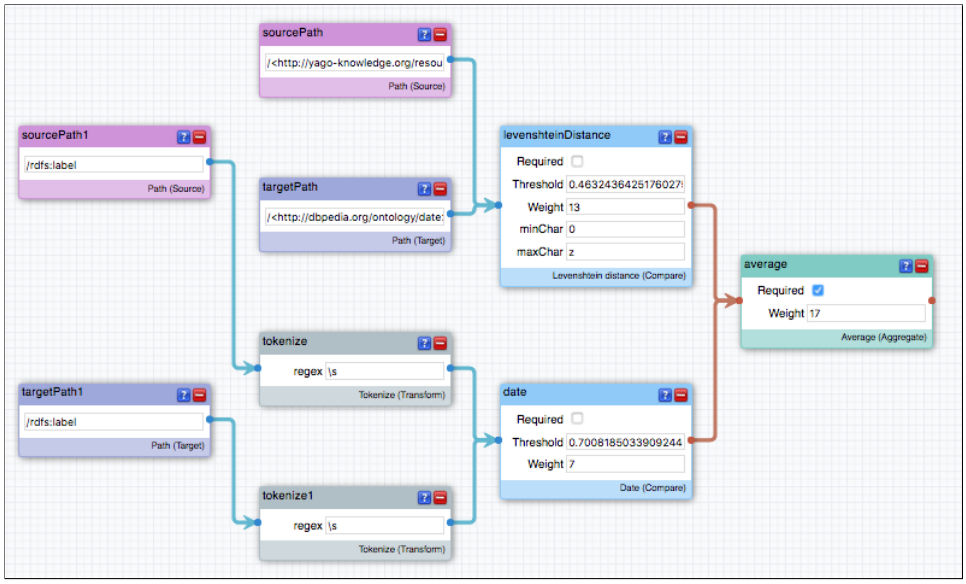


Figure 4.10: Learned Rule for Low Similarity of Matches and High Similarity of Non-matches (LH)

threshold of 1.13. The weight for this first comparator is set to 16. For the second comparator, the tokenized and lowercased date values are examined with a Levenshtein distance with a threshold of 2.28 and a weight of 18. Those two comparators are then combined with the average aggregator that uses the weights of the input comparators to assess their importance. Therefore, the maximum number of character transformations is set to one for the labels and two for the dates. As the positive matches in the subset have a low similarity score of the URIs, this rule uses the of the entities to find duplicates. In addition, the values of the date property are considered as equal for up to two edit transformations. This edit distance similarity measure on date values seems unusual but serves two purposes. On the one hand, this similarity measure takes the many partly missing month or day values for dates such as *2014-07-##* in YAGO into account. In addition, this deals with small variations for the date property values of events that took place over a period of time, i.e. several days. An example is the *Battle of Arcole* event that has (among others) the *1796-11-17* date value for the dbr:Battle_of_Arcole event in DBpedia and the *1796-11-15* value for the respective yago:Battle_of_the_Bridge_of_Arcole event in YAGO. On the other hand, this similarity measure matches events with diverse labels that happened on the same day such as dbr:Operation_Leopard_(1969) and yago:Defense_of_Umuahia that both happened on the *1969-04-22*. Due to the higher

weight of the date similarity measure, those two events are correctly considered to describe duplicates. Nevertheless, this characteristic has to be considered carefully as all events that have the same date are matches as duplicates by the LR rule.

Figure 4.10 shows the fourth learned rule in Silk for matches with a low similarity and non-matches with a high similarity (LH). It thus has the most difficult input data to learn a rule. The threshold for the absolute Levenshtein distance is set to 0.46 for the date values having a weight of 13. In addition, the tokenized English labels are compared using the date comparator with a threshold of 0.7 and a weight of 7. The date comparator calculates the difference in days so a threshold of less than 1 implies exact equality. The two comparators are then combined using the average aggregator. Hence, the rule looks for entities having the exact same date and, at the same time, tries to extract year dates from the labels. Many recurring events in DBpedia and YAGO have year dates in the URIs and label values such as *Woodstock 1999* to distinguish those events that happen every (few) year(s). Similar to the LR rule, due to the higher weight of the Levenshtein similarity measure on the date, the LH rule matches all events that have the same value. In addition, all events that have the same year number in the label value are declared as duplicates. For example, the dbr:United_Kingdom_general_election,_2005_(Scotland) event is correctly considered matched with yago:2005_United_Kingdom_general_election_results_in_Scotland. Note that all events having 2005 in the label will be considered as duplicates.

Precision, Recall, and F-Measure

The precision, recall, and F-measure scores for the baseline and the learned matching rules on the gold standard indicate the effectiveness of the rules to separate the matches and the non-matches of the gold standard. Table 4.11 lists these results.

Matching Rule	Precision	Recall	F-measure
RR	1.00	0.95	0.97
RH	1.00	0.95	0.97
LR	0.79	0.71	0.75
LH	0.98	0.15	0.25

Table 4.11: Effectiveness of the Learned Matching Rules in Silk

The simpler cases with random and therefore highly similar URIs for the matches (RR and RH) score high precision and recall values. The more difficult cases for matches having a low similarity score (LR, LH) struggle more to separate the matches and the non-matches within the gold standard which is implied by the

lower precision and recall scores. Note that Table 4.11 lists the overfitted effectiveness results of rules that are trained and tested on the gold standard. This Table only provides the confidence of the rules on the gold standard.

As already seen in Figures 4.1 and 4.2, the URI is a highly effective indicator of matches and non-matches. Beating the optimized baseline is thus difficult for the datasets at hand. This significant descriptiveness of the URIs is also used by the RR an RH subsets, to learn the rules. RR consists of random entities and therefore has similar characteristics as the full gold standard. The learned rule thus uses a high threshold similar to the baseline. It even requires perfect equality of the URIs and allows up to one character transformation on the English labels. The RH rule reduces this threshold as all input entities have a very high similarity on the URIs. Hence, the URI effectively separates the matches that mostly have a very high similarity and the non-matches that have a lower similarity than the matches. As only matches having a low similarity on the URI are used for the LR and LH subsets, the URI is not distinctive for these input files. Therefore, the date and label properties are used for the rules learned on the LR and LH subsets.

For evaluating the learned rules and comparing them to the baseline, the matching rules are replicated in the Web Data Integration framework. The rules are then evaluated on the full datasets using StBl with BlFi ($r = 0.5$) as blocking method. Table 4.12 lists the results of the baseline and the four learned rules on the full datasets. The matching rule that maximizes the F-measure is then determined as final rule.

Matching Rule	Precision	Recall	F-measure
Baseline	**0.9141**	**0.3881**	**0.5448**
RR	0.7163	0.3894	0.5045
RH	0.5497	0.3924	0.4579
LR	0.7177	0.3884	0.5040
LH	0.9993	0.3033	0.4653

Table 4.12: Effectiveness of the Learned Matching Rules on the Dataset

The baseline rule that uses the scaled Levenshtein similarity measure on the stripped URI is indeed the best matching rule with a score of 0.54 for the F-measure. The distribution of the gold standard as illustrated in Figure 4.1 illustrates that this baseline is hard to beat as a similarity measure on the URIs with a high threshold separates the matches and the non-matches pretty well. The rule learned from the random entities from the gold standard (RR) scores slightly better in recall but loses precision. Interesting to see is that the learned rule for matches having a low similarity and random non-matches (LR) as well as the most difficult case

of matches having a low similarity and non-matches with a high similarity (LH) score better than expected. The LH case learns a very robust rule with a precision of 99.9% but therefore also misses many matches as indicated by the low recall score.

Sign Test on True Positives

The sign test is used as additional effectiveness analysis on the number of True Positives (TP). Hence, the different matching rules are compared regarding the TPs that are found by both rules and by either of the rules. Nevertheless, the use of statistical tests for evaluating the significance of effectiveness measures has to be considered carefully due to the underlying assumptions it makes [59, p. 136].

The gold standard contains 9707 TPs in total. Table 4.13 lists the number of declared and true positives for each matching rule.

Matching Rule	Declared Duplicates	True Positives
Baseline	4121	3767
RR	5277	3780
RH	6929	3809
LR	5253	3770
LH	2946	2944

Table 4.13: Declared Duplicates and True Positives pf the Matching Rules

As defined in Equation 2.13, the ratio of TPs to the declared duplicates is the precision of the matching rules and corresponds to the results in Table 4.12. For analyzing the TPs of the different matching rules, Table 4.14 lists the values that are require for the sign test. X and Y stand for the respective matching rules that are compared to each other. Ten sign tests are calculated for comparing all five matching rules with each other.

n_+ indicates the number of TPs that are only identified by X while n_- denotes the number of TPs that are only found by Y. The ties (n_0) are all TPs that are found by both matching rules. n denotes the sum of n_+ and n_- as defined in Equation 2.17.

H_0 is the hypothesis that both matching rules identify roughly the same number of TPs. The contradicting hypothesis H_1 states that one of the matching rules finds significantly more TPs than the other one. For a significance level of $\alpha = 0.05$ the critical value t is either determined by Equation 2.18 or by table lookup for $n \leq 20$ [16, pp. 433-444]. t is rounded to two digits after the decimal point for Table 4.14.

X	Y	n_+	n_-	n_0	n	t	Reject H_0
Baseline	RR	1	14	3766	15	3.00	True
Baseline	RH	1	43	3766	44	15.37	True
Baseline	LR	11	14	3756	25	7.50	False
Baseline	LH	869	46	2898	915	427.25	True
RR	RH	1	30	3779	31	9.93	True
RR	LR	10	0	3770	10	1.00	True
RR	LH	876	40	2904	916	427.73	True
RH	LR	40	1	3769	41	14.10	True
RH	LH	880	15	2929	895	417.58	True
LR	LH	866	40	2904	906	422.90	True

Table 4.14: Sign Test for the Matching Rules

The hypothesis H_0 is rejected and H_1 holds, if $n_+ \leq t$ or $n_+ \geq n - t$. H_1 states that the matching rule with a larger number of found TPs finds significantly more TPs than the other matching rule.

When looking at the n_+, n_-, and n_0 numbers, Table 4.14 clearly shows that, apart from the LH rule, all matching rules find mostly the same duplicates (n_0) and only identify a small number of diverging TPs. Regarding this finding, the difference of 42 duplicates for the baseline and the RH rule is the largest variation.

As H_0 is rejected for the baseline and the LR matching rules, the number of found TPs for those two rules does not differ significantly. Therefore, the rules are considered to find approximately the same TPs. Nevertheless, as listed in Table 4.13, the baseline rule declares a lot less duplicates, which leads to a much higher precision score as shown in Table 4.12.

Overall, the RH matching rule finds most TPs with a significant difference in regards to the RR rule. This high number of correct TPs comes with the cost of declaring many incorrect duplicates as well and a resulting precision of 54.97%, which is the lowest score of all matching rules.

The RR rule finds the significantly second most TPs of all matching rules with 3780 identified TPs with declaring 5277 duplicates. As the RR rule declares much fewer duplicates than the RH rule, the precision score is higher with a similar recall.

The baseline and LR rules identify significantly less TPs than the RH rule but at the same time significantly more TPs than the LH rule. Depending on the aim of the date integration process, finding significantly more TPs can be attained with a potentially higher number of incorrectly declared duplicates. This effect can also be seen when comparing the precision and recall scores.

Final Findings

The baseline and LR rules are considered to find the same number of TPs, whereas the baseline rule declares 1132 fewer duplicates than the LR rule. Therefore, the precision of the baseline rule is almost 20% higher. Both rules find significantly more TPs than the LH rule.

The LH rule declares the fewest duplicates of all matching rules and thus also has the smallest number of TPs. Nevertheless, all but two out of the 2946 declared duplicates are correct which leads to a precision of 99.9%.

Depending on the aim of the data integration application, the number of TPs as well as the precision, recall, and F-measure scores have to be considered. The RH rule should be used, if the number of found TPs or the recall is most important. If errors are expensive and should be avoided, the LH rule that has the highest precision score, should be chosen. The baseline rule scores the highest F-measure and therefore best combines precision and recall. In addition, the number of TPs is high when regarding the low number of declared duplicates leading to a precision score of 91.41%. Following Occam's razor principle, due to the simplicity and the best score in the F-measure, the baseline rule with a threshold of 0.96 is determined as final matching rule.

4.3 Web Application for Integrating the Event Data

A Web application using the best blocking method and matching rule is implemented using Apache Tomcat 8.5.6 as Web server and running Java 1.8. The JavaScript library D3.js is used to illustrate the results on a world map. D3.js version 4.2.7 is used for this project. Figure 4.11 shows a screenshot of the Web application.

The different dot colors in the world map indicate the data source. Blue dots display events that are extracted from DBpedia, yellow dots illustrate YAGO events, and red dots indicate fused events from both data sources. Information of the events are shown on mouseover. The user can use the extracted local data or query the data from the public SPARQL endpoints dynamically. For both cases, date or keyword filters can be applied. Start and end dates, as well as case-insensitive keywords on the English label can be specified to filter the data. Hereby, it is important to mention that when using the date parameter, all events that do not have a date property are discarded as they will not match the query. In addition, the user can switch between showing all results or the fused events only.

The map in Figure 4.11 illustrates that integrating two data sources of the same domain that are extracted from the Semantic Web are very useful to enrich the data. For the example events that are received for all events from January 1, 2000 up-

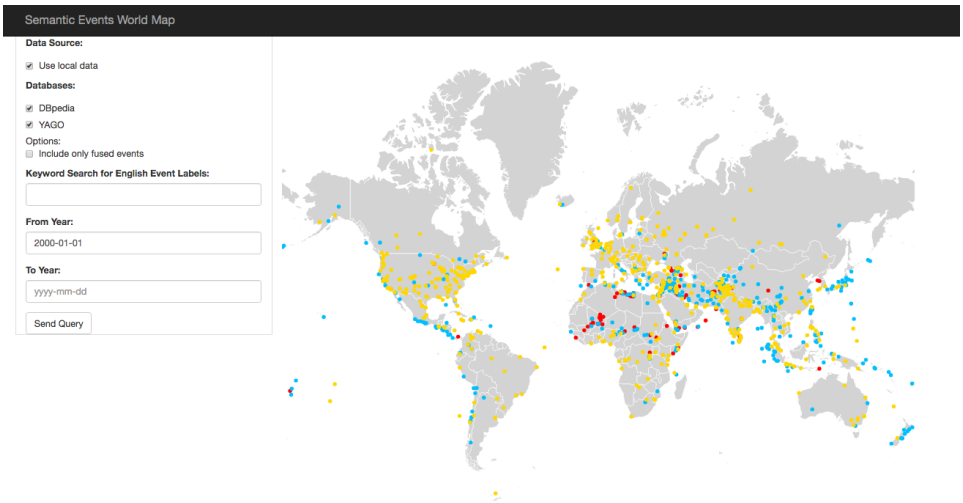


Figure 4.11: Screenshot of the Implemented Web Application

wards, the two data sources complement each other with is implied by the many blue and yellow dots. In addition, duplicates (red dots) are found using the Standard Blocking and Block Filtering methods with a ratio of 0.5 and the baseline matching rule.

Chapter 5

Conclusions

The results of the experiments are summarized in Chapter 5.1. In addition, potential future work is outlined in Chapter 5.2.

5.1 Summary

Integrating Web data from different sources is a challenging but rewarding problem. The more duplicates that describe the same real-world entities from different sources are linked to each other, the more powerful the Semantic Web becomes. With the increasing size and amount of datasets published in the Semantic Web, the integration and linkage of entities becomes a recurring and computational expensive task. As the complexity of comparing each entity from the first dataset with every entity of the second dataset is quadratic, blocking algorithms try to reduce this complexity with removing superfluous comparisons. When building the blocks, a large number of redundant comparisons can arise that might even lead to a reduction regarding efficiency. Hence, additional block and comparison cleaning methods aim to remove those redundant and further superfluous comparisons with the tradeoff to potentially miss a small amount of duplicates.

For the datasets with event data which are extracted from DBpedia and YAGO, the blocking experiments clearly show that using basic block building methods, such as Standard Blocking or Attribute Clustering, lead to a higher number of overall comparisons. This effect can occur when the number of additional redundant comparisons is higher than the number of saved superfluous comparisons. Nevertheless, as soon as block and optionally comparison cleaning methods are applied on the built blocks, the efficiency of blocking increases extensively. Therefore, one of the key findings of the experiments is that block and comparison cleaning methods should be applied when integrating large datasets to boost the performance

of finding duplicates in the datasets with trading just a small amount of potential missed matches. This is especially important if many diverse attributes are used. Depending on the application or the goal of the data integration, the importance of finding all matches in contrast to reducing the number of overall comparisons has to be assessed. A small amount of Pairs Completeness can be traded for a large increase in Pairs Quality and Reduction Ratio.

Subsets with special characteristics are extracted from the data to learn robust matching rules using the Silk framework. Nevertheless, the URIs of the linked entities in the gold standard are very similar and often identical. Thus, an optimized matching rule using the Levenshtein similarity measure and a high threshold of 0.96 works best for the datasets in this experiment.

5.2 Future Work

Potential future work includes different aspects of the experiments regarding the datasets, the frameworks, and the methods.

For the datasets, more diverse attributes could be added to analyze their usefulness. Therefore, more subsets having different attributes could be created. The integration of additional and more diverse datasets from the Semantic Web would lead to richer data. Nevertheless, when increasing the amount of different heterogeneous sources, data integration becomes even more challenging.

For integrating more and larger datasets, the frameworks have to be optimized to improve the runtime of the integration. The computational expensive tasks of the frameworks should be parallelized to allow the testing and integration of larger datasets. The bottlenecks of the frameworks should be found and enhanced if possible.

In addition to the executed experiments, more methods for all sub-tasks of blocking could be analyzed and compared to each other. This would lead to a more complete comparisons of different block building, block cleaning, and comparison cleaning methods.

Bibliography

[1] Salima Benbernou, Xin Huang, and Mourad Ouziri. Fusion of Big RDF Data: A Semantic Entity Resolution and Query Rewriting-Based Inference Approach. In *Web Information Systems Engineering – WISE 2015*, pages 300–307. Springer, Cham, November 2015. doi: 10.1007/978-3-319-26187-4_27.

[2] Tim Berners-Lee. Linked Data - Design Issues. `https://www.w3.org/DesignIssues/LinkedData.html`, July 2006.

[3] Tim Berners-Lee. The Next Web. `https://www.ted.com/talks/tim_berners_lee_on_the_next_web?language=en`, February 2009.

[4] Tim Berners-Lee. Semantic Web. `https://www.w3.org/2009/Talks/0120-campus-party-tbl/#(14)`, 2009.

[5] Tim Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. `http://www.ietf.org/rfc/rfc3986.txt`, 2005.

[6] Mikhail Bilenko and Raymond J. Mooney. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 39–48, New York, NY, USA, 2003. ACM. ISBN 978-1-58113-737-8. doi: 10.1145/956750.956759.

[7] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far:. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009. ISSN 1552-6283, 1552-6291. doi: 10.4018/jswis.2009081901.

[8] Jens Bleiholder and Felix Naumann. Data Fusion. *ACM Comput. Surv.*, 41(1): 1:1–1:41, January 2009. ISSN 0360-0300. doi: 10.1145/1456650.1456651.

[9] Jens Bleiholder, Jens Bleiholder, and Felix Naumann. Conflict Handling Strategies in an Integrated Information System. 2006.

[10] Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres, and Dave Reynolds. RIF Core Dialect (Second Edition). https://www.w3.org/TR/rif-core/, 2013.

[11] Tim Bray, Jean Paoli, and C.M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. https://www.w3.org/TR/1998/REC-xml-19980210, 1998.

[12] Tim Bray, Dave Hollander, Andrew Layman, Richard Tobin, and Henry S. Thompson. Namespaces in XML 1.0 (Third Edition). https://www.w3.org/TR/xml-names/, 2009.

[13] Dan Brickley, R.V. Guha, and Brian McBride. RDF Schema 1.1. https://www.w3.org/TR/rdf-schema/, February 2014.

[14] Peter Christen. A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *IEEE Trans. on Knowl. and Data Eng.*, 24(9): 1537–1555, September 2012. ISSN 1041-4347. doi: 10.1109/TKDE.2011.127.

[15] William W. Cohen, William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. pages 73–78, 2003.

[16] W. J Conover. *Practical Nonparametric Statistics*. Wiley, New York, 1971. ISBN 978-0-471-16851-5. OCLC: 119004.

[17] Gustavo De Assis Costa and José M. Parente de Oliveira. Large-Scale Entity Resolution for Semantic Web Data Integration. In *13th IADIS International Conference WWW/Internet 2014 (ICWI 2014)*, Porto, Portugal, 2014.

[18] Gustavo De Assis Costa and José M. Parente de Oliveira. A Blocking Scheme for Entity Resolution in the Semantic Web. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 1138–1145, March 2016. doi: 10.1109/AINA.2016.23.

[19] AnHai Doan, Alon Halevy, and Zachary Ives. *Principles of Data Integration*. Morgan Kaufmann, Saint Louis, US, 2012. ISBN 978-0-12-391479-8.

[20] Xin Luna Dong and Divesh Srivastava. Big Data Integration. *Synthesis Lectures on Data Management*, 7(1):1–198, February 2015. ISSN 2153-5418, 2153-5426. doi: 10.2200/S00578ED1V01Y201404DTM040.

[21] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Kevin Murphy, Shaohua Sun, and Wei Zhang. From Data Fusion to Knowledge Fusion. *arXiv:1503.00302 [cs]*, March 2015.

[22] Uwe Draisbach and Felix Naumann. A generalization of blocking and windowing algorithms for duplicate detection. pages 18–24. IEEE, September 2011. ISBN 978-1-4577-0865-7. doi: 10.1109/ICDKE.2011.6053920.

[23] M Duerst and M Suignard. Internationalized Resource Identifiers (IRIs). `http://www.ietf.org/rfc/rfc3987.txt`, 2005.

[24] Vasilis Efthymiou, George Papadakis, George Papastefanatos, Kostas Stefanidis, and Themis Palpanas. Parallel meta-blocking for scaling entity resolution over big heterogeneous data. *Information Systems*, 65:137–157, April 2017. ISSN 03064379. doi: 10.1016/j.is.2016.12.001.

[25] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, January 2007. ISSN 1041-4347. doi: 10.1109/TKDE.2007.250581.

[26] Ivan P. Fellegi and Alan B. Sunter. A Theory for Record Linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, December 1969. ISSN 0162-1459. doi: 10.1080/01621459.1969.10501049.

[27] Avigdor Gal. Uncertain Schema Matching. *Synthesis Lectures on Data Management*, 3(1):1–97, February 2011. ISSN 2153-5418. doi: 10.2200/S00337ED1V01Y201102DTM013.

[28] George Giannakopoulos, Vangelis Karkaletsis, George Vouros, and Panagiotis Stamatopoulos. Summarization system evaluation revisited: N-gram graphs. *ACM Transactions on Speech and Language Processing*, 5(3):1–39, October 2008. ISSN 15504875. doi: 10.1145/1410358.1410359.

[29] Steve Harris and Andy Seaborne. SPARQL 1.1 Query Language. `https://www.w3.org/TR/2013/REC-sparql11-query-20130321/`, 2013.

[30] Patrick J. Hayes and Peter F. Patel-Schneider. RDF 1.1 Semantics. `https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/#blank-nodes`, 2014.

[31] Tom Heath and Christian Bizer. *Linked Data:Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011. ISBN 978-1-60845-431-0.

[32] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure. *Semantic Web*. eXamen.press. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-33993-9. doi: 10.1007/978-3-540-33994-6.

[33] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. `https://www.w3.org/Submission/SWRL/`, 2004.

[34] Robert Isele and Christian Bizer. Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, 5(11):1638–1649, 2012.

[35] Robert Isele and Christian Bizer. Active Learning of Expressive Linkage Rules using Genetic Programming. *Web Semantics: Science, Services and Agents on the World Wide Web*, 23(0), 2013. ISSN 1570-8268.

[36] Batya Kenig and Avigdor Gal. MFIBlocks: An effective blocking algorithm for entity resolution. *Information Systems*, 38(6):908–926, September 2013. ISSN 03064379. doi: 10.1016/j.is.2012.11.008.

[37] Hung-sik Kim and Dongwon Lee. HARRA: Fast Iterative Hashed Record Linkage for Large-scale Data Collections. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 525–536, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-945-9. doi: 10.1145/1739041.1739104.

[38] Graham Klyne and Jeremy J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. `https://www.w3.org/TR/2002/WD-rdf-concepts-20021108/`, 2002.

[39] Graham Klyne, Jeremy J. Carroll, and Brian McBride. RDF 1.1 Concepts and Abstract Syntax. `https://www.w3.org/TR/rdf11-concepts/`, 2014.

[40] Hanna Köpcke and Erhard Rahm. Frameworks for Entity Matching: A Comparison. *Data Knowl. Eng.*, 69(2):197–210, February 2010. ISSN 0169-023X. doi: 10.1016/j.datak.2009.10.003.

[41] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of Entity Resolution Approaches on Real-world Match Problems. *Proc. VLDB Endow.*, 3 (1-2):484–493, September 2010. ISSN 2150-8097. doi: 10.14778/1920841.1920904.

[42] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. 6(2), 2013.

[43] Ulf Leser and Felix Naumann. *Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. dpunkt-Verlag, 2007. ISBN 978-3-89864-400-6. Google-Books-ID: vNGsGQAACAAJ.

[44] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language Profiles (Second Edition). `https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/#Introduction`, 2012.

[45] Felix Naumann and Melanie Herschel. An Introduction to Duplicate Detection. *Synthesis Lectures on Data Management*, 2(1): 1–87, January 2010. ISSN 2153-5418, 2153-5426. doi: 10.2200/S00262ED1V01Y201003DTM003.

[46] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic Linkage of Vital Records: Computers can be used to extract "follow-up" statistics of families from files of routine records. *Science*, 130(3381): 954–959, October 1959. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.130.3381.954.

[47] George Papadakis and Themis Palpanas. Blocking for large-scale Entity Resolution: Challenges, algorithms, and practical examples. pages 1436–1439. IEEE, May 2016. ISBN 978-1-5090-2020-1. doi: 10.1109/ICDE.2016.7498364.

[48] George Papadakis, Gianluca Demartini, Peter Fankhauser, and Philipp Kärger. The missing links: Discovering hidden same-as links among a billion of triples. page 453. ACM Press, 2010. ISBN 978-1-4503-0421-4. doi: 10.1145/1967486.1967557.

[49] George Papadakis, Ekaterini Ioannou, Claudia Niederée, and Peter Fankhauser. Efficient Entity Resolution for Large Heterogeneous Information Spaces. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 535–544, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0493-1. doi: 10.1145/1935826.1935903.

[50] George Papadakis, Ekaterini Ioannou, Claudia Niederée, Themis Palpanas, and Wolfgang Nejdl. Beyond 100 million entities: Large-scale blocking-based resolution for heterogeneous data. page 53. ACM Press, 2012. ISBN 978-1-4503-0747-5. doi: 10.1145/2124295.2124305.

[51] George Papadakis, Ekaterini Ioannou, Themis Palpanas, Claudia Niederee, and Wolfgang Nejdl. A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2665–2682, December 2013. ISSN 1041-4347. doi: 10.1109/TKDE.2012.150.

[52] George Papadakis, Georgia Koutrika, Themis Palpanas, and Wolfgang Nejdl. Meta-Blocking: Taking Entity Resolutionto the Next Level. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1946–1960, August 2014. ISSN 1041-4347. doi: 10.1109/TKDE.2013.54.

[53] George Papadakis, George Papastefanatos, Themis Palpanas, and Manolis Koubarakis. Boosting the Efficiency of Large-Scale Entity Resolution with Enhanced Meta-Blocking. *Big Data Research*, 6:43–63, December 2016. ISSN 2214-5796. doi: 10.1016/j.bdr.2016.08.002.

[54] George Papadakis, George Papastefanatos, Themis Palpanas, and Manolis Koubarakis. Scaling Entity Resolution to Large, Heterogeneous Data with Enhanced Meta-blocking. In *19th International Conference on Extending Database Technology*, pages 221–232, Bourdeaux, France, 2016. OpenProceedings. ISBN 978-3-89318-070-7. doi: 10.5441/002/edbt.2016.22.

[55] George Papadakis, Jonathan Svirsky, Avigdor Gal, and Themis Palpanas. Comparative Analysis of Approximate Blocking Techniques for Entity Resolution. In *ResearchGate*, September 2016.

[56] Heiko Paulheim. *Skalierbarkeit von Ontology-Matching-Verfahren*. Master thesis, TU Darmstadt, 2008.

[57] Erhard Rahm. The Case for Holistic Data Integration. In *Advances in Databases and Information Systems*, pages 11–27. Springer, Cham, August 2016. doi: 10.1007/978-3-319-44039-2_2.

[58] Erhard Rahm and Hong Hai Do. Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin*, 23:2000, 2000.

[59] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979. ISBN 978-0-408-70929-3.

[60] Daniel Ringler and Heiko Paulheim. One Knowledge Graph to Rule them All? - Analyzing the Differences between DBpedia, YAGO, Wikidata & co. *Forthcoming*, 2017.

[61] Riccardo Rosati. The upper layers of the Semantic Web. `https://www.dis.uniroma1.it/~rosati/krst-1415/sw-slides6.pdf`, 2014/2015.

[62] Wei Shen, Jianyong Wang, and Jiawei Han. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460, February 2015. ISSN 1041-4347. doi: 10.1109/TKDE.2014.2327028.

[63] Sidney Siegel. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, New York, 1956. ISBN 978-0-07-085689-9. OCLC: 166020.

[64] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242667.

[65] Chenchen Sun, Derong Shen, Yue Kou, Tiezheng Nie, and Ge Yu. A genetic algorithm based entity resolution approach with active learning. *Frontiers of Computer Science*, 11(1):147–159, February 2017. ISSN 2095-2228, 2095-2236. doi: 10.1007/s11704-015-5276-6.

[66] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. XML Schema Part 1: Structures Second Edition. `https://www.w3.org/TR/2004/REC-xmlschema-1-20041028/`, 2004.

[67] Denny Vrandečić and Markus Krötzsch. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57(10):78–85, September 2014. ISSN 00010782. doi: 10.1145/2629489.

[68] W3C. WGS84 Geo Positioning: An RDF vocabulary. `https://www.w3.org/2003/01/geo/wgs84_pos#`, 2009.

[69] W3C. OWL 2 Web Ontology Language Document Overview (Second Edition). `https://www.w3.org/TR/owl2-overview/`, December 2012.

[70] W3C. Inference - W3C. `https://www.w3.org/standards/semanticweb/inference`, 2017.

[71] W3C. Ontologies - W3C. `https://www.w3.org/standards/semanticweb/ontology`, 2017.

[72] Melanie Weis, Felix Naumann, and Franziska Brosy. A duplicate detection benchmark for XML (and relational) data. *Proc: of Workshop on Information Quality for Information Systems (IQIS)*, 2006.

Appendix A

Program Code

The full source code and the documentation is provided on the attached CD. The following SPARQL queries are used for counting the event instances for the respective Knowledge Graph implicitly specifying how events and their instances are defined.

```
SELECT COUNT(DISTINCT(?e)) WHERE {
 ?e rdf:type dbo:Event .
}
```

Listing A.1: SPARQL Query for Counting the Event Instances in DBpedia

```
SELECT COUNT(DISTINCT(?e)) WHERE {
 ?e rdf:type yago:wordnet\_event\_100029378 .
}
```

Listing A.2: SPARQL Query for Counting the Event Instances in YAGO

```
SELECT COUNT(DISTINCT(?e)) WHERE {
 ?e wdt:P31 wd:Q1656682 .
}
```

Listing A.3: SPARQL Query for Counting the Event Instances in Wikidata

Appendix B

Further Experimental Results

All results are also provided on the attached CD.

r	#blocks	total comparisons	PC	PQ	RR	α
0.05	345	172405	2.00E-04	5.80E-06	0.992	0.000
0.1	4954	928580	0.175	9.39E-04	0.999	0.175
0.15	9159	2126919	0.543	0.001	1.000	0.542
0.2	14742	3390766	0.923	0.001	1.000	0.923
0.25	17326	6660278	0.978	7.33E-04	1.000	0.978
0.3	17815	9609262	0.980	5.08E-04	0.999	0.979
0.35	17835	1.25E+07	0.980	3.91E-04	0.999	0.979
0.4	18457	2.22E+07	0.982	2.21E-04	0.999	0.981
0.45	18509	3.12E+07	0.982	1.57E-04	0.998	0.980
0.5	**19333**	**7.54E+07**	**0.988**	**6.54E-05**	**0.996**	**0.985**
0.55	19333	8.13E+07	0.988	6.06E-05	0.996	0.984
0.6	19346	1.01E+08	0.988	4.88E-05	0.995	0.983
0.65	19380	1.77E+08	0.992	2.80E-05	0.991	0.983
0.7	19392	2.21E+08	0.992	2.24E-05	0.989	0.982
0.75	19496	2.82E+08	0.992	1.76E-05	0.986	0.979
0.8	19496	3.32E+08	0.992	1.49E-05	0.984	0.977
0.85	19514	4.83E+08	0.993	1.03E-05	0.977	0.970
0.9	19517	9.16E+08	0.994	5.41E-06	0.956	0.950
0.95	19517	1.28E+09	0.994	3.87E-06	0.938	0.933
1	19517	1.34E+09	0.994	3.70E-06	0.936	0.930

Table B.1: BlFi Ratios for StBl for Subset 1

r	#blocks	total comparisons	PC	PQ	RR	α
0.05	342	171872	2.00E-04	5.82E-06	0.992	0.000
0.1	4985	943231	0.174	9.18E-04	0.999	0.174
0.15	9182	2139922	0.539	0.001	1.000	0.539
0.2	14770	3402320	0.918	0.001	1.000	0.917
0.25	17354	6664953	0.974	7.29E-04	1.000	0.974
0.3	17840	9622732	0.978	5.07E-04	0.999	0.977
0.35	17861	1.25E+07	0.978	3.89E-04	0.999	0.978
0.4	18486	2.22E+07	0.982	2.20E-04	0.999	0.981
0.45	18539	3.12E+07	0.982	1.57E-04	0.998	0.980
0.5	**19366**	**7.53E+07**	**0.988**	**6.54E-05**	**0.996**	**0.985**
0.55	19366	8.13E+07	0.988	6.07E-05	0.996	0.984
0.6	19380	1.01E+08	0.988	4.88E-05	0.995	0.983
0.65	19413	1.77E+08	0.992	2.80E-05	0.991	0.983
0.7	19425	2.22E+08	0.992	2.23E-05	0.989	0.982
0.75	19529	2.83E+08	0.992	1.75E-05	0.986	0.979
0.8	19529	3.34E+08	0.992	1.48E-05	0.984	0.976
0.85	19547	4.84E+08	0.993	1.02E-05	0.977	0.970
0.9	19550	9.17E+08	0.994	5.41E-06	0.956	0.950
0.95	19550	1.29E+09	0.994	3.86E-06	0.938	0.933
1	19550	1.35E+09	0.994	3.69E-06	0.935	0.930

Table B.2: BlFi Ratios for StBl for Subset 2

r	#blocks	total comparisons	PC	PQ	RR	α
0.05	5273	724720	0.030	2.08E-04	1.000	0.030
0.1	22361	2442134	0.929	0.002	1.000	0.929
0.15	23979	4624798	0.957	0.001	1.000	0.957
0.2	24882	8724877	0.970	5.54E-04	1.000	0.969
0.25	25449	1.56E+07	0.977	3.12E-04	0.999	0.976
0.3	25536	2.37E+07	0.985	2.07E-04	0.999	0.984
0.35	25572	3.74E+07	0.987	1.32E-04	0.998	0.985
0.4	**25620**	**5.89E+07**	**0.989**	**8.38E-05**	**0.997**	**0.986**
0.45	25625	9.93E+07	0.990	4.97E-05	0.996	0.985
0.5	25688	2.01E+08	0.993	2.46E-05	0.991	0.985
0.55	25688	2.62E+08	0.995	1.90E-05	0.989	0.984
0.6	25688	4.36E+08	0.999	1.14E-05	0.981	0.980
0.65	25689	1.08E+09	1.000	4.62E-06	0.954	0.954
0.7	25690	2.22E+09	1.000	2.25E-06	0.906	0.906
0.75	25693	8.27E+09	1.000	6.03E-07	0.648	0.648
0.8	25693	1.41E+10	1.000	3.53E-07	0.399	0.399
0.85	25693	2.42E+10	1.000	2.06E-07	-0.029	-0.029
0.9	25694	3.86E+10	1.000	1.29E-07	-0.644	-0.644
0.95	25694	5.08E+10	1.000	9.81E-08	-1.165	-1.165
1	25694	7.00E+10	1.000	7.13E-08	-1.980	-1.980

Table B.3: BlFi Ratios for StBl for Subset 3

r	#blocks	total comparisons	PC	PQ	RR	α
0.05	5708	729489	0.044	2.97E-04	1.000	0.043
0.1	22471	2469232	0.922	0.002	1.000	0.921
0.15	24140	4661142	0.950	0.001	1.000	0.950
0.2	25046	8681116	0.969	5.57E-04	1.000	0.969
0.25	25612	1.55E+07	0.976	3.14E-04	0.999	0.975
0.3	25700	2.36E+07	0.985	2.08E-04	0.999	0.984
0.35	25737	3.73E+07	0.988	1.32E-04	0.998	0.986
0.4	**25786**	**5.87E+07**	**0.990**	**8.41E-05**	**0.997**	**0.987**
0.45	25791	9.89E+07	0.990	4.99E-05	0.996	0.985
0.5	25851	2.00E+08	0.993	2.48E-05	0.991	0.985
0.55	25851	2.61E+08	0.995	1.91E-05	0.989	0.984
0.6	25851	4.35E+08	0.999	1.14E-05	0.981	0.980
0.65	25853	1.07E+09	1.000	4.65E-06	0.954	0.954
0.7	25854	2.21E+09	1.000	2.25E-06	0.906	0.906
0.75	25856	8.26E+09	1.000	6.04E-07	0.648	0.648
0.8	25856	1.41E+10	1.000	3.54E-07	0.399	0.399
0.85	25856	2.41E+10	1.000	2.07E-07	-0.028	-0.028
0.9	25857	3.86E+10	1.000	1.29E-07	-0.643	-0.643
0.95	25857	5.08E+10	1.000	9.81E-08	-1.164	-1.164
1	25857	7.00E+10	1.000	7.13E-08	-1.980	-1.980

Table B.4: BlFi Ratios for StBl for Subset 4

r	#blocks	total comparisons	PC	PQ	RR	α
0.05	18298	888641	0.729	0.004	1.000	0.729
0.1	28699	2662171	0.888	0.002	1.000	0.888
0.15	30875	5380877	0.935	8.67E-04	1.000	0.935
0.2	31797	9727768	0.955	4.90E-04	0.999	0.954
0.25	32426	1.77E+07	0.973	2.74E-04	0.999	0.972
0.3	32543	2.82E+07	0.984	1.74E-04	0.999	0.982
0.35	32600	4.56E+07	0.988	1.08E-04	0.998	0.986
0.4	32661	7.06E+07	0.992	7.00E-05	0.997	0.989
0.45	32675	1.17E+08	0.996	4.24E-05	0.995	0.990
0.5	**32775**	**2.10E+08**	**1.000**	**2.38E-05**	**0.991**	**0.991**
0.55	32780	2.85E+08	1.000	1.75E-05	0.988	0.988
0.6	32781	4.68E+08	1.000	1.07E-05	0.980	0.980
0.65	32784	9.24E+08	1.000	5.40E-06	0.961	0.961
0.7	32785	1.75E+09	1.000	2.85E-06	0.926	0.926
0.75	32789	5.47E+09	1.000	9.12E-07	0.768	0.768
0.8	32789	9.83E+09	1.000	5.07E-07	0.583	0.583
0.85	32789	1.88E+10	1.000	2.65E-07	0.203	0.203
0.9	32790	3.18E+10	1.000	1.57E-07	-0.350	-0.350
0.95	32790	4.82E+10	1.000	1.03E-07	-1.047	-1.047
1	32790	7.55E+10	1.000	6.60E-08	-2.205	-2.205

Table B.5: BlFi Ratios for StBl for Subset 5

r	#blocks	total comparisons	PC	PQ	RR	α
0.05	345	172405	2.00E-04	5.80E-06	0.992	1.99E-04
0.1	4954	928580	0.175	9.39E-04	0.999	0.175
0.15	9159	2126919	0.543	0.001	1.000	0.542
0.2	14742	3390766	0.923	0.001	1.000	0.923
0.25	17326	6660278	0.978	7.33E-04	1.000	0.978
0.3	17815	9609262	0.980	5.08E-04	0.999	0.979
0.35	17835	1.25E+07	0.980	3.91E-04	0.999	0.979
0.4	18457	2.22E+07	0.982	2.21E-04	0.999	0.981
0.45	18509	3.12E+07	0.982	1.57E-04	0.998	0.980
0.5	**19333**	**7.54E+07**	**0.988**	**6.54E-05**	**0.996**	**0.985**
0.55	19333	8.13E+07	0.988	6.06E-05	0.996	0.984
0.6	19346	1.01E+08	0.988	4.88E-05	0.995	0.983
0.65	19380	1.77E+08	0.992	2.80E-05	0.991	0.983
0.7	19392	2.21E+08	0.992	2.24E-05	0.989	0.982
0.75	19496	2.82E+08	0.992	1.76E-05	0.986	0.979
0.8	19496	3.32E+08	0.992	1.49E-05	0.984	0.977
0.85	19514	4.83E+08	0.993	1.03E-05	0.977	0.970
0.9	19517	9.16E+08	0.994	5.41E-06	0.956	0.950
0.95	19517	1.28E+09	0.994	3.87E-06	0.938	0.933
1	19517	1.34E+09	0.994	3.70E-06	0.936	0.930

Table B.6: BlFi Ratios for ACl for Subset 1

r	#blocks	total comparisons	PC	PQ	RR	α
0.05	342	171872	2.00E-04	5.82E-06	0.992	1.99E-04
0.1	4985	943231	0.174	9.18E-04	0.999	0.174
0.15	9182	2139922	0.539	0.001	1.000	0.539
0.2	14770	3402320	0.918	0.001	1.000	0.917
0.25	17354	6664953	0.974	7.29E-04	1.000	0.974
0.3	17840	9622732	0.978	5.07E-04	0.999	0.977
0.35	17861	1.25E+07	0.978	3.89E-04	0.999	0.978
0.4	18486	2.22E+07	0.982	2.20E-04	0.999	0.981
0.45	18539	3.12E+07	0.982	1.57E-04	0.998	0.980
0.5	**19366**	**7.53E+07**	**0.988**	**6.54E-05**	**0.996**	**0.985**
0.55	19366	8.13E+07	0.988	6.07E-05	0.996	0.984
0.6	19380	1.01E+08	0.988	4.88E-05	0.995	0.983
0.65	19413	1.77E+08	0.992	2.80E-05	0.991	0.983
0.7	19425	2.22E+08	0.992	2.23E-05	0.989	0.982
0.75	19529	2.83E+08	0.992	1.75E-05	0.986	0.979
0.8	19529	3.34E+08	0.992	1.48E-05	0.984	0.976
0.85	19547	4.84E+08	0.993	1.02E-05	0.977	0.970
0.9	19550	9.17E+08	0.994	5.41E-06	0.956	0.950
0.95	19550	1.29E+09	0.994	3.86E-06	0.938	0.933
1	19550	1.35E+09	0.994	3.69E-06	0.935	0.930

Table B.7: BlFi Ratios for ACl for Subset 2

r	#blocks	total comparisons	PC	PQ	RR	α
0.05	5273	724720	0.030	2.08E-04	1.000	0.030
0.1	22361	2442134	0.929	0.002	1.000	0.929
0.15	23979	4624798	0.957	0.001	1.000	0.957
0.2	24882	8724877	0.970	5.54E-04	1.000	0.969
0.25	25449	1.56E+07	0.977	3.12E-04	0.999	0.976
0.3	25536	2.37E+07	0.985	2.07E-04	0.999	0.984
0.35	25572	3.74E+07	0.987	1.32E-04	0.998	0.985
0.4	**25620**	**5.89E+07**	**0.989**	**8.38E-05**	**0.997**	**0.986**
0.45	25625	9.93E+07	0.990	4.97E-05	0.996	0.985
0.5	25688	2.01E+08	0.993	2.46E-05	0.991	0.985
0.55	25688	2.62E+08	0.995	1.90E-05	0.989	0.984
0.6	25688	4.36E+08	0.999	1.14E-05	0.981	0.980
0.65	25689	1.08E+09	1.000	4.62E-06	0.954	0.954
0.7	25690	2.22E+09	1.000	2.25E-06	0.906	0.906
0.75	25693	8.27E+09	1.000	6.03E-07	0.648	0.648
0.8	25693	1.41E+10	1.000	3.53E-07	0.399	0.399
0.85	25693	2.42E+10	1.000	2.06E-07	-0.029	-0.029
0.9	25694	3.86E+10	1.000	1.29E-07	-0.644	-0.644
0.95	25694	5.08E+10	1.000	9.81E-08	-1.165	-1.165
1	25694	7.00E+10	1.000	7.13E-08	-1.980	-1.980

Table B.8: BlFi Ratios for ACl for Subset 3

r	#blocks	total comparisons	PC	PQ	RR	α
0.05	5708	729489	0.044	2.97E-04	1.000	0.043
0.1	22471	2469232	0.922	0.002	1.000	0.921
0.15	24140	4661142	0.950	0.001	1.000	0.950
0.2	25046	8681116	0.969	5.57E-04	1.000	0.969
0.25	25612	1.55E+07	0.976	3.14E-04	0.999	0.975
0.3	25700	2.36E+07	0.985	2.08E-04	0.999	0.984
0.35	25737	3.73E+07	0.988	1.32E-04	0.998	0.986
0.4	**25786**	**5.87E+07**	**0.990**	**8.41E-05**	**0.997**	**0.987**
0.45	25791	9.89E+07	0.990	4.99E-05	0.996	0.985
0.5	25851	2.00E+08	0.993	2.48E-05	0.991	0.985
0.55	25851	2.61E+08	0.995	1.91E-05	0.989	0.984
0.6	25851	4.35E+08	0.999	1.14E-05	0.981	0.980
0.65	25853	1.07E+09	1.000	4.65E-06	0.954	0.954
0.7	25854	2.21E+09	1.000	2.25E-06	0.906	0.906
0.75	25856	8.26E+09	1.000	6.04E-07	0.648	0.648
0.8	25856	1.41E+10	1.000	3.54E-07	0.399	0.399
0.85	25856	2.41E+10	1.000	2.07E-07	-0.028	-0.028
0.9	25857	3.86E+10	1.000	1.29E-07	-0.643	-0.643
0.95	25857	5.08E+10	1.000	9.81E-08	-1.164	-1.164
1	25857	7.00E+10	1.000	7.13E-08	-1.980	-1.980

Table B.9: BlFi Ratios for ACl for Subset 4

r	#blocks	total comparisons	PC	PQ	RR	α
0.05	11877	622456	0.332	0.003	1.000	0.332
0.1	23050	4127867	0.855	0.001	1.000	0.855
0.15	27111	8890916	0.903	5.07E-04	0.999	0.903
0.2	30156	8608590	0.947	5.49E-04	0.999	0.947
0.25	31646	1.45E+07	0.962	3.32E-04	0.999	0.961
0.3	31909	2.31E+07	0.969	2.09E-04	0.999	0.967
0.35	32019	2.76E+07	0.971	1.76E-04	0.998	0.970
0.4	32127	4.04E+07	0.979	1.21E-04	0.998	0.977
0.45	32159	5.62E+07	0.981	8.71E-05	0.997	0.978
0.5	32423	1.08E+08	0.983	4.52E-05	0.995	0.978
0.55	**32426**	**1.26E+08**	**0.986**	**3.90E-05**	**0.994**	**0.980**
0.6	32427	2.21E+08	0.986	2.23E-05	0.990	0.976
0.65	32434	3.16E+08	0.986	1.55E-05	0.985	0.972
0.7	32435	5.55E+08	0.986	8.86E-06	0.974	0.961
0.75	32446	1.19E+09	0.988	4.15E-06	0.944	0.932
0.8	32446	2.26E+09	0.990	2.18E-06	0.894	0.884
0.85	32446	5.04E+09	0.995	9.84E-07	0.763	0.759
0.9	32447	9.44E+09	0.995	5.26E-07	0.556	0.553
0.95	32447	1.40E+10	0.995	3.55E-07	0.341	0.340
1	32447	1.87E+10	0.995	2.65E-07	0.119	0.118

Table B.10: BlFi Ratios for ACl for Subset 5

WS	PA	#blocks	total comparisons	PC	PQ	RR	α
ARCS	CEP	1	418257	0.857	0.0102	1.000	0.857
ARCS	CNP	56525	56525	0.888	0.0784	1.000	0.888
ARCS	ReCNP	33470	33470	0.876	0.1305	1.000	0.876
ARCS	ReWNP	56451	1.78E+07	0.973	2.72E-04	0.999	0.972
CBS	CEP	1	418257	0.072	0.0009	0.998	0.072
CBS	CNP	56525	56525	0.886	0.0782	1.000	0.886
CBS	ReCNP	31745	31745	0.874	0.1374	1.000	0.874
CBS	**ReWNP**	**56525**	**2.75E+07**	**0.975**	**1.77E-04**	**0.999**	**0.974**
ECBS	CEP	1	418257	0.173	0.0021	0.999	0.172
ECBS	CNP	56525	56525	0.910	0.0803	1.000	0.910
ECBS	ReCNP	35688	35688	0.904	0.1264	1.000	0.904
ECBS	ReWNP	56490	3.62E+07	0.975	1.34E-04	0.998	0.973
JS	CEP	1	418257	0.964	0.0115	1.000	0.964
JS	CNP	56525	56525	0.910	0.0803	1.000	0.910
JS	ReCNP	35631	35631	0.904	0.1266	1.000	0.904
JS	ReWNP	56525	3.38E+07	0.975	1.44E-04	0.998	0.973
EJS	CEP	1	418257	0.783	0.0093	1.000	0.783
EJS	CNP	56525	56525	0.910	0.0803	1.000	0.910
EJS	ReCNP	36492	36492	0.904	0.1236	1.000	0.904
EJS	ReWNP	56518	4.34E+07	0.975	1.12E-04	0.998	0.972

Table B.11: WS and PA for StBl and BlFi (r=0.5) for Subset 1

WS	PA	#blocks	total comparisons	PC	PQ	RR	α
ARCS	CEP	1	421277	0.856	0.0101	1.000	0.856
ARCS	CNP	56526	56526	0.884	0.0780	1.000	0.884
ARCS	ReCNP	33295	33295	0.866	0.1297	1.000	0.866
ARCS	ReWNP	56453	1.72E+07	0.973	2.83E-04	0.999	0.972
CBS	CEP	1	421277	0.072	0.0008	0.998	0.071
CBS	CNP	56526	56526	0.877	0.0774	1.000	0.877
CBS	ReCNP	31540	31540	0.861	0.1362	1.000	0.861
CBS	**ReWNP**	**56526**	**2.77E+07**	**0.975**	**1.76E-04**	**0.999**	**0.974**
ECBS	CEP	1	421277	0.167	0.0020	0.999	0.167
ECBS	CNP	56526	56526	0.900	0.0794	1.000	0.900
ECBS	ReCNP	35530	35530	0.892	0.1252	1.000	0.892
ECBS	ReWNP	56519	3.61E+07	0.973	1.35E-04	0.998	0.971
JS	CEP	1	421277	0.948	0.0112	1.000	0.948
JS	CNP	56526	56526	0.898	0.0793	1.000	0.898
JS	ReCNP	35455	35455	0.891	0.1253	1.000	0.891
JS	ReWNP	56526	4.31E+07	0.973	1.13E-04	0.998	0.971
EJS	CEP	1	421277	0.936	0.0111	1.000	0.936
EJS	CNP	56526	56526	0.897	0.0791	1.000	0.897
EJS	ReCNP	36331	36331	0.887	0.1218	1.000	0.887
EJS	ReWNP	56526	4.38E+07	0.974	1.11E-04	0.998	0.972

Table B.12: WS and PA for StBl and BlFi (r=0.5) for Subset 2

WS	PA	#blocks	total comparisons	PC	PQ	RR	α
ARCS	CEP	1	707104	0.867	0.0061	1.000	0.867
ARCS	CNP	56708	113400	0.745	0.0328	1.000	0.745
ARCS	ReCNP	32299	49808	0.740	0.0741	1.000	0.740
ARCS	**ReWNP**	**56707**	**1.71E+07**	**0.967**	**2.83E-04**	**0.999**	**0.966**
CBS	CEP	1	707104	0.296	0.0021	1.000	0.296
CBS	CNP	56708	113400	0.453	0.0199	1.000	0.453
CBS	ReCNP	25696	34557	0.424	0.0613	1.000	0.424
CBS	ReWNP	56708	2.94E+07	0.766	1.30E-04	0.999	0.765
ECBS	CEP	1	707104	0.314	0.0022	1.000	0.314
ECBS	CNP	56708	113400	0.409	0.0180	1.000	0.409
ECBS	ReCNP	28244	40974	0.373	0.0455	1.000	0.373
ECBS	ReWNP	56708	3.04E+07	0.583	9.55E-05	0.999	0.582
JS	CEP	1	707104	0.110	0.0008	1.000	0.110
JS	CNP	56708	113400	0.409	0.0180	1.000	0.409
JS	ReCNP	24528	35373	0.201	0.0284	1.000	0.201
JS	ReWNP	56708	3.58E+07	0.595	8.28E-05	0.998	0.594
EJS	CEP	1	707104	0.183	0.0013	1.000	0.183
EJS	CNP	56708	113400	0.490	0.0215	1.000	0.490
EJS	ReCNP	28015	42880	0.291	0.0339	1.000	0.291
EJS	ReWNP	56708	3.61E+07	0.837	1.16E-04	0.998	0.835

Table B.13: WS and PA for StBl and BlFi (r=0.4) for Subset 3

WS	PA	#blocks	total comparisons	PC	PQ	RR	α
ARCS	CEP	1	710315	0.867	0.0061	1.000	0.867
ARCS	CNP	56708	113401	0.744	0.0327	1.000	0.744
ARCS	ReCNP	32135	49629	0.737	0.0740	1.000	0.737
ARCS	**ReWNP**	**56707**	**1.67E+07**	**0.967**	**2.90E-04**	**0.999**	**0.966**
CBS	CEP	1	710315	0.300	0.0021	1.000	0.300
CBS	CNP	56708	113401	0.452	0.0199	1.000	0.452
CBS	ReCNP	25630	34497	0.423	0.0611	1.000	0.423
CBS	ReWNP	56708	2.93E+07	0.769	1.31E-04	0.999	0.768
ECBS	CEP	1	710315	0.315	0.0022	1.000	0.315
ECBS	CNP	56708	113401	0.408	0.0180	1.000	0.408
ECBS	ReCNP	28262	41079	0.373	0.0453	1.000	0.373
ECBS	ReWNP	56708	3.03E+07	0.585	9.61E-05	0.999	0.584
JS	CEP	1	710315	0.113	0.0008	1.000	0.113
JS	CNP	56708	113401	0.408	0.0180	1.000	0.408
JS	ReCNP	24578	35505	0.206	0.0290	1.000	0.206
JS	ReWNP	56708	3.58E+07	0.587	8.18E-05	0.998	0.586
EJS	CEP	1	710315	0.193	0.0014	1.000	0.193
EJS	CNP	56708	113401	0.488	0.0215	1.000	0.488
EJS	ReCNP	28076	43031	0.295	0.0341	1.000	0.294
EJS	ReWNP	56708	3.60E+07	0.836	1.16E-04	0.998	0.835

Table B.14: WS and PA for StBl and BlFi (r=0.4) for Subset 4

WS	PA	#blocks	total comparisons	PC	PQ	RR	α
ARCS	CEP	1	1327787	0.890	0.0033	1.000	0.890
ARCS	CNP	56717	283562	0.752	0.0132	1.000	0.752
ARCS	ReCNP	41443	135371	0.750	0.0276	1.000	0.750
ARCS	**ReWNP**	**56717**	**3.98E+07**	**0.966**	**1.21E-04**	**0.998**	**0.964**
CBS	CEP	1	1327787	0.692	0.0026	1.000	0.692
CBS	CNP	56717	283562	0.647	0.0114	1.000	0.646
CBS	ReCNP	37109	88888	0.625	0.0351	1.000	0.625
CBS	ReWNP	56717	3.55E+07	0.903	1.27E-04	0.998	0.902
ECBS	CEP	1	1327787	0.632	0.0024	1.000	0.632
ECBS	CNP	56717	283562	0.677	0.0119	1.000	0.677
ECBS	ReCNP	40678	113049	0.637	0.0281	1.000	0.637
ECBS	ReWNP	56717	1.00E+08	0.912	4.53E-05	0.996	0.908
JS	CEP	1	1327787	0.114	4.28E-04	1.000	0.114
JS	CNP	56717	283562	0.677	0.0119	1.000	0.677
JS	ReCNP	33061	86542	0.384	0.0222	1.000	0.384
JS	ReWNP	56717	1.32E+08	0.914	3.45E-05	0.994	0.909
EJS	CEP	1	1327787	0.126	4.74E-04	1.000	0.126
EJS	CNP	56717	283562	0.672	0.0118	1.000	0.672
EJS	ReCNP	35866	100865	0.405	0.0200	1.000	0.405
EJS	ReWNP	56717	1.31E+08	0.924	3.51E-05	0.994	0.919

Table B.15: WS and PA for StBl and BlFi (r=0.5) for Subset 5

WS	PA	#blocks	total comparisons	PC	PQ	RR	α
ARCS	CEP	1	418257	0.857	0.010	1.000	0.857
ARCS	CNP	56525	56525	0.888	0.078	1.000	0.888
ARCS	ReCNP	33470	33470	0.876	0.131	1.000	0.876
ARCS	ReWNP	56451	1.78E+07	0.973	2.72E-04	0.999	0.972
CBS	CEP	1	418257	0.072	0.001	0.998	0.072
CBS	CNP	56525	56525	0.886	0.078	1.000	0.886
CBS	ReCNP	31745	31745	0.874	0.137	1.000	0.874
CBS	**ReWNP**	**56525**	**2.75E+07**	**0.975**	**1.77E-04**	**0.999**	**0.974**
ECBS	CEP	1	418257	0.173	0.002	0.999	0.172
ECBS	CNP	56525	56525	0.910	0.080	1.000	0.910
ECBS	ReCNP	35688	35688	0.904	0.126	1.000	0.904
ECBS	ReWNP	56490	3.62E+07	0.975	1.34E-04	0.998	0.973
JS	CEP	1	418257	0.964	0.011	1.000	0.964
JS	CNP	56525	56525	0.910	0.080	1.000	0.910
JS	ReCNP	35631	35631	0.904	0.127	1.000	0.904
JS	ReWNP	56525	3.38E+07	0.975	1.44E-04	0.998	0.973
EJS	CEP	1	418257	0.783	0.009	1.000	0.783
EJS	CNP	56525	56525	0.910	0.080	1.000	0.910
EJS	ReCNP	36492	36492	0.904	0.124	1.000	0.904
EJS	ReWNP	56518	4.34E+07	0.975	1.12E-04	0.998	0.972

Table B.16: WS and PA for ACl and BlFi (r=0.5) for Subset 1

WS	PA	#blocks	total comparisons	PC	PQ	RR	α
ARCS	CEP	1	421277	0.856	0.010	1.000	0.856
ARCS	CNP	56526	56526	0.884	0.078	1.000	0.884
ARCS	ReCNP	33295	33295	0.866	0.130	1.000	0.866
ARCS	ReWNP	56453	1.72E+07	0.973	2.83E-04	0.999	0.972
CBS	CEP	1	421277	0.072	8.47E-04	0.998	0.071
CBS	CNP	56526	56526	0.877	0.077	1.000	0.877
CBS	ReCNP	31540	31540	0.861	0.136	1.000	0.861
CBS	**ReWNP**	**56526**	**2.77E+07**	**0.975**	**1.76E-04**	**0.999**	**0.974**
ECBS	CEP	1	421277	0.167	0.002	0.999	0.167
ECBS	CNP	56526	56526	0.900	0.079	1.000	0.900
ECBS	ReCNP	35530	35530	0.892	0.125	1.000	0.892
ECBS	ReWNP	56519	3.61E+07	0.973	1.35E-04	0.998	0.971
JS	CEP	1	421277	0.948	0.011	1.000	0.948
JS	CNP	56526	56526	0.898	0.079	1.000	0.898
JS	ReCNP	35455	35455	0.891	0.125	1.000	0.891
JS	ReWNP	56526	4.31E+07	0.973	1.13E-04	0.998	0.971
EJS	CEP	1	421277	0.936	0.011	1.000	0.936
EJS	CNP	56526	56526	0.897	0.079	1.000	0.897
EJS	ReCNP	36331	36331	0.887	0.122	1.000	0.887
EJS	ReWNP	56526	4.38E+07	0.974	1.11E-04	0.998	0.972

Table B.17: WS and PA for ACl and BlFi (r=0.5) for Subset 2

WS	PA	#blocks	total comparisons	PC	PQ	RR	α
ARCS	CEP	1	707104	0.867	0.006	1.000	0.867
ARCS	CNP	56708	113400	0.745	0.033	1.000	0.745
ARCS	ReCNP	32299	49808	0.740	0.074	1.000	0.740
ARCS	**ReWNP**	**56707**	**1.71E+07**	**0.967**	**2.83E-04**	**0.999**	**0.966**
CBS	CEP	1	707104	0.296	0.002	1.000	0.296
CBS	CNP	56708	113400	0.453	0.020	1.000	0.453
CBS	ReCNP	25696	34557	0.424	0.061	1.000	0.424
CBS	ReWNP	56708	2.94E+07	0.766	1.30E-04	0.999	0.765
ECBS	CEP	1	707104	0.314	0.002	1.000	0.314
ECBS	CNP	56708	113400	0.409	0.018	1.000	0.409
ECBS	ReCNP	28244	40974	0.373	0.045	1.000	0.373
ECBS	ReWNP	56708	3.04E+07	0.583	9.55E-05	0.999	0.582
JS	CEP	1	707104	0.110	7.75E-04	1.000	0.110
JS	CNP	56708	113400	0.409	0.018	1.000	0.409
JS	ReCNP	24528	35373	0.201	0.028	1.000	0.201
JS	ReWNP	56708	3.58E+07	0.595	8.28E-05	0.998	0.594
EJS	CEP	1	707104	0.183	0.001	1.000	0.183
EJS	CNP	56708	113400	0.490	0.022	1.000	0.490
EJS	ReCNP	28015	42880	0.291	0.034	1.000	0.291
EJS	ReWNP	56708	3.61E+07	0.837	1.16E-04	0.998	0.835

Table B.18: WS and PA for ACl and BlFi (r=0.4) for Subset 3

WS	PA	#blocks	total comparisons	PC	PQ	RR	α
ARCS	CEP	1	710315	0.867	0.006	1.000	0.867
ARCS	CNP	56708	113401	0.744	0.033	1.000	0.744
ARCS	ReCNP	32135	49629	0.737	0.074	1.000	0.737
ARCS	**ReWNP**	**56707**	**1.67E+07**	**0.967**	**2.90E-04**	**0.999**	**0.966**
CBS	CEP	1	710315	0.300	0.002	1.000	0.300
CBS	CNP	56708	113401	0.452	0.020	1.000	0.452
CBS	ReCNP	25630	34497	0.423	0.061	1.000	0.423
CBS	ReWNP	56708	2.93E+07	0.769	1.31E-04	0.999	0.768
ECBS	CEP	1	710315	0.315	0.002	1.000	0.315
ECBS	CNP	56708	113401	0.408	0.018	1.000	0.408
ECBS	ReCNP	28262	41079	0.373	0.045	1.000	0.373
ECBS	ReWNP	56708	3.03E+07	0.585	9.61E-05	0.999	0.584
JS	CEP	1	710315	0.113	0.001	1.000	0.113
JS	CNP	56708	113401	0.408	0.018	1.000	0.408
JS	ReCNP	24578	35505	0.206	0.029	1.000	0.206
JS	ReWNP	56708	3.58E+07	0.587	8.18E-05	0.998	0.586
EJS	CEP	1	710315	0.193	0.001	1.000	0.193
EJS	CNP	56708	113401	0.488	0.021	1.000	0.488
EJS	ReCNP	28076	43031	0.295	0.034	1.000	0.294
EJS	ReWNP	56708	3.60E+07	0.836	1.16E-04	0.998	0.835

Table B.19: WS and PA for ACl and BlFi (r=0.4) for Subset 4

WS	PA	#blocks	total comparisons	PC	PQ	RR	α
ARCS	CEP	1	966831	0.904	0.005	1.000	0.904
ARCS	CNP	56717	226854	0.813	0.018	1.000	0.812
ARCS	ReCNP	33827	89217	0.801	0.045	1.000	0.800
ARCS	**ReWNP**	**56717**	**2.29E+07**	**0.966**	**2.11E-04**	**0.999**	**0.965**
CBS	CEP	1	966831	0.658	0.003	0.999	0.657
CBS	CNP	56717	226854	0.589	0.013	1.000	0.589
CBS	ReCNP	31093	59647	0.568	0.048	1.000	0.568
CBS	ReWNP	56717	3.17E+07	0.957	1.51E-04	0.999	0.956
ECBS	CEP	1	966831	0.689	0.004	0.999	0.689
ECBS	CNP	56717	226854	0.683	0.015	1.000	0.683
ECBS	ReCNP	37395	85488	0.661	0.039	1.000	0.661
ECBS	ReWNP	56717	5.34E+07	0.955	8.92E-05	0.997	0.952
JS	CEP	1	966831	0.265	0.001	1.000	0.265
JS	CNP	56717	226854	0.715	0.016	1.000	0.715
JS	ReCNP	35339	79323	0.612	0.038	1.000	0.612
JS	ReWNP	56717	7.12E+07	0.959	6.71E-05	0.997	0.955
EJS	CEP	1	966831	0.338	0.002	1.000	0.338
EJS	CNP	56717	226854	0.728	0.016	1.000	0.728
EJS	ReCNP	36927	90850	0.628	0.034	1.000	0.627
EJS	ReWNP	56717	7.08E+07	0.961	6.77E-05	0.997	0.958

Table B.20: WS and PA for ACl and BlFi (r=0.55) for Subset 5

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Masterarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Mannheim, den 12.04.2017