Gottfried Wilhelm Leibniz Universität Hannover
Faculty of Electrical Engineering and Computer Science

Integration of Systematic Review Services with a Scholarly Knowledge Graph

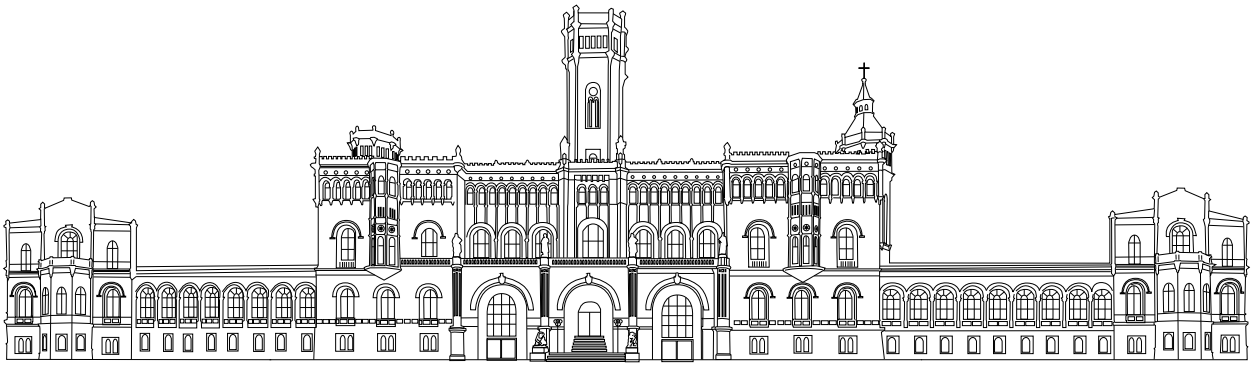
A thesis submitted in fulfillment of the requirements for the degree of
Master of Science in Computer Science

BY

Thassilo Martin Schiepanski
Matriculation number: 10004884
E-mail: t.schiepanski@stud.uni-hannover.de

First evaluator: Prof. Dr. Sören Auer
Second evaluator: Dr. Markus Stocker
Supervisor: MSc. Golsa Heidari

October 31st 2023



Declaration of Authorship

I, Thassilo Martin Schiepanski, declare that this thesis titled, 'Integration of Systematic Review Services with a Scholarly Knowledge Graph' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Thassilo Martin Schiepanski

Date: 01.11.2023

Realize that everything connects to everything else.

– Leonardo da Vinci

Creativity is just connecting things. When you ask creative people how they did something, they feel a little guilty because they didn't really do it, they just saw something. It seemed obvious to them after a while.

– Steve Jobs

Osyris is a genus of plants [that] can survive and grow by themselves, [but] they also opportunistically tap into the root systems of nearby plants and parasitize them.

– Wikipedia

Acknowledgements

First and foremost, I address major gratitude to my (surrogate) supervisor Dr. Markus Stocker for the strong support along the way. Not only for the continuous feedback on my thesis, but also for enjoyable conversations. Furthermore, I thank Golsa Heidari for introducing me to the topic, as well as boarding me on the *ORKG* project. Not least, great thanks to Prof. Dr. Sören Auer for making this thesis possible with his contributions to the field of knowledge engineering. Thanks are evenly due to all members of the *ORKG* team. Moreover, I greatly appreciate and share the passion which the *TIB* research exerts towards open science.

On a second note, I thank my fellow students for helping with annotation tasks throughout this thesis. For the included survey, thanks go to everyone who participated.

Deep gratitude belongs to my family. I am grateful for my parents' unconditional support. Likewise, I am for the bedrock bond between my siblings and me. Thanks for always sharing different perspectives on life with me.

To comply with examination regulations, the thesis title was stipulated beforehand. Retrospectively, however, a less generic, more inspiring title would be as follows:

Towards OSyRIS: ORKG Systematic Review Integration Service

Abstract

Scholarly Knowledge Graphs and Systematic Reviews share a primary purpose: Compiling a scientific state-of-the-art by comparing relevant scholarly contributions for a specific research question. Knowledge Graphs, however, do not represent a particularly accessible data structure for human editing. Instead, a landscape of Systematic Review Services supports researchers with the Systematic Review methodology. Nonetheless, integration of Systematic Review Services promises great potential for growth of scholarly Knowledge Graphs.

This thesis proposes OSyRIS, a comprehensive web service approach and implementation for integrating arbitrary Systematic Review Services with a scholarly Knowledge Graph, namely ORKG. The service bases on a threefold approach: First, a plausible infrastructure specification. Upon that, the service seamlessly integrates with the existing ORKG ecosystem. Second, *ScopedDBSCANScoring*, a novel scoring algorithm for predicate mappings based on pivotal Systematic Review data. Application of the algorithm fosters conceptual integrity of the underlying Knowledge Graph. And third, a User Interface represented by a widget of interactive modals that guide Systematic Review Service users through the data export process. At that, two task-specific UI elements are deliberately designed in order to support selection of quality mappings.

Based on the implementation, OSyRIS is shown in-depth quality: First of all, a representative integration into a popular Systematic Review Service provides an overall proof of correct functionality. Empirical evaluations furthermore support that the *ScopedDBSCANScoring* algorithm outperforms a predicate label only-based mapping baseline that is currently implemented in ORKG. As a preceding contribution, a gold standard data set for evaluation of label and literal-based Knowledge Graph disambiguation algorithms is presented. For the mapping task specific User Interface, an empirical evaluation retrieves most usable of a few alternative element representations.

Keywords: *Scholarly Knowledge Graphs, Systematic Reviews, Interoperability of Web-based Services, Knowledge Graphs Disambiguation, Digital Scholarship*

Zusammenfassung

Wissenschaftliche Knowledge Graphs und Systematic Reviews dienen einem gemeinsamen Zweck: Der Zusammenstellung eines aktuellen Stands der Wissenschaft (*state-of-the-art*), durch den Vergleich von relevanten Beiträgen zu einer bestimmten Forschungsfrage. Knowledge Graphs stellen jedoch keine besonders zugängliche Datenstruktur für die Bearbeitung durch Menschen dar. Stattdessen unterstützt eine Landschaft von Systematic Review-Diensten Forscher bei der Systematic Review-Methodik. Dennoch verspricht die Integration von Systematic Review-Diensten großes Wachstumspotenzial für wissenschaftliche Knowledge Graphs.

Diese Arbeit stellt OSyRIS vor, den umfangreichen Ansatz samt Umsetzung eines Web-Dienstes für die Integration von beliebigen Systematic Review-Diensten mit einem wissenschaftlichen Knowledge Graph, nämlich ORKG. Der Dienst basiert auf einem dreistufigen Ansatz: Erstens, einer verständlichen Infrastruktur-Spezifikation. Anhand dieser lässt sich der Dienst nahtlos mit dem existierenden ORKG-Ökosystem integrieren. Zweitens, *ScopedDBSCANScoring*, einem neuartigen Bewertungsalgorithmus für Predikat-Übersetzungen basierend auf ausschlaggebenden Systematic Review-Daten. Anwendung des Algorithmus pflegt die konzeptionelle Integrität des zugrundeliegenden Knowledge Graphs. Und drittens, einer Benutzerschnittstelle die durch ein Widget bestehend aus Interaktionsmodalen besteht, welche wiederum Nutzer:innen von Systematic Review-Diensten durch den Datenexport führen. Dabei werden zwei aufgabenspezifische Schnittstellenelemente sorgfältig entworfen, um Nutzer:innen bei der Wahl hochwertiger Übersetzungen zu untertützen.

Basierend auf der Umsetzung wird OSyRIS umfassende Qualität nachgewiesen: Zunächst zeigt eine repräsentative Integration in einen bekannten Systematic Review-Dienst die korrekte Funktionalität des Dienstes. Eine empirische Evaluierung unterstützt, dass die Leistung des $ScopedDBSCANScoring$-Algorithmus ein lediglich auf Predikat-Bezeichnern beruhendes Übersetzungsverfahren, wie es derzeit in ORKG implementiert ist, übertrifft. Als einhergehender Beitrag dieser Arbeit wird ein Gold-Standard-Datensatz für die Bewertung im Systematic Review-Kontext eingesetzter Algorithmen für die Knowledge Graph-Disambiguation präsentiert. Für die Benutzerschnittstelle, mit speziellem Bezug auf die Übersetzungsaufgabe, stellt eine empirische Evaluierung die benutzbarste einiger alternativen Darstellungen heraus.

Contents

List of Figures

List of Tables

List of Listings

Glossary

API Application Programming Interface

CSS Cascading Style Sheets

CSV Comma-Separated Values

DOI Digital Object Identifier

DOM Document Object Model

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

ID Identifier

IR Information Retrieval

JS JavaScript

JSON JavaScript Object Notation

KG Knowledge Graph

ORKG Open Research Knowledge Graph

OSyRIS ORKG Systematic Review Integration Service

RDF Resource Description Framework

REST Representational State Transfer

RML RDF Mapping Language

SPARQL SPARQL Protocol And RDF Query Language

SR Systematic Review

SR Service Systematic Review Service

UI User Interface

URL Uniform Resource Locator

W3C World Wide Web Consortium

web World Wide Web

Chapter 1

Introduction

Scholarly communication is a key aspect of scientific progress. The better a common state-of-the-art is known to researchers around the globe, the more it can be challenged. Desirably, a state-of-the-art should be accessible without much effort. The World Wide Web (web) was developed to advance scholarly communication. Ever since then, scientific literature has been provided by digital libraries. However, the web has specified a syntax for interlinking documents, but not abstract concepts described in these documents [1]. Therefore, researchers today still default to performing comprehensive literature research before approaching a subject. Yet, the web is right now in the transition from a document- to a concept-centric web, namely the semantic web [2]. At that, the Knowledge Graph (KG) is the ubiquitous formalisation for encoding abstract concepts. Scholarly scholarly KGs have thus been an auspicious approach towards improved accessibility of the scientific state-of-the-art [3], [4].

Except for granting logical merits, KG representations have disadvantages, nonetheless. Particularly, data structures like lists or tables feel more natural from a human perspective. In order to allow arbitrarily abstract concepts to be described, KGs require rather complex graph patterns. That being said, scholarly KGs represent but means of centralised scholarly communication infrastructures. The higher level, on the contrary, is served by interactive tools that engage with established scientific methodologies. Prevalent in many subjects, Systematic Reviews (SRs) represent a fundamental methodology and synonymous artifact that is supported by a landscape of applications on the web. Reviews compile a scoped state-of-the art for answering a predefined research question [5], [6]. In this regard, the interaction of web-based Systematic Review Services (SR Services) and scholarly KGs promises great benefits for once more progressing scholarly communication.

1.1 Motivation

At present, about thirty thousand SRs are published per year [7]. Moreover, the trend is increasing [8]. On the same page, hundreds of thousands of researchers use SR Services on a regular basis [9], [10]. SR Services aid the entire SR process. At the final stages, they accumulate the results in one place. Results of an SR are key contributions from original work. Those provide inherent value for growing scholarly KGs. Yet, the way from an SR Service to a scholarly KG is bridged by the entire publication lifecycle. A SR is published as usual and the official document must then explicitly be targeted for knowledge extraction and integration. In the end, the SR data is transferred to a higher level representation (e.g. paper) and later reduced again to the lower level KG representation. The lack of an interface in direct proximity to the results is an apparent problem. An effortless way to export the results would be in favour of scholarly KGs. Optimally, users of an SR Service would be provided with an option to export the available results with no more than a few clicks; right from the service application.

1.2 Problem

The problem is evident: Scholarly KGs would profit from SR data collected at SR Services. An integration interface is therefore desirable. However, interoperability is primarily valuable for the scholarly KG end. For that reason, low effort integration costs are a key requirement for an integration interface in order to engage with SR Service providers. Furthermore, according translation of SR data to proper KG representations is a mandatory intermediate step, viewing not only syntax, but also semantics. For it is known that KGs come with a well-known bottleneck: Ambiguous concept entities in the graph alleviate its purpose, as knowledge inference may no longer be complete. An according translation of concepts described in SR data would favour disambiguation and overall KG integrity. For scholarly KGs to scale with the pace of science, however, data should be able to be created by anyone at any time. Scholarly KGs would hence need to grow without (real time) supervision. Inherently, an integration interface SR Services would need to work fully-automated at the scholarly KG end.

1.3 Research Questions

To be concise, the above stated problem can be reflected by the following research questions:

RQ 1 *What interface for low-effort integration can a scholarly Knowledge Graph provide to arbitrary Systematic Review Services?*

Data collected in SR Services provides value for scholarly KGs. Therefore, the demand of interoperability clearly emanates from the scholarly KG side. At scale, an interface for interoperability should allow for simple integration from the SR Service side.

RQ 2 *What data processing measures can foster KG disambiguation in the underlying KG upon data exchange?*

Data read from SR Services require a syntactic translation in order to allow for being written to schoarly KGs. But also the concepts described within the respective SR should be mapped to those existing in the KG, i.e. disambiguated with the KGs at export time.

RQ 3 *What User Interface can support an intuitive application of the KG disambiguation measures upon data exchange?*

Data mappings between SR Services and scholarly KGs are merely an automated estimate. The human user should thus approve or disapprove individual mapping recommendations. The UI should accordingly support the user with making a reasonable decision.

1.4 Structure

The basic structure of this work is pursuant to common literature from the field. Initially, background knowledge required to understand this work is shared. Subsequently, related work is presented and commented with respect to this work. The main part starts with describing the approach for answering the research questions. Then, the implementation of the approach is touched upon. Based on the implementation, the following evaluation supports the value of the proposed service and its components. Due to the broad problem statement, the main part addresses topics from a broader spectrum of disciplines. The targeted service is ultimately a software engineering product. The KG disambiguation to foster KG integrity is moreover subject to knowledge engineering and machine learning in further view. The human

user interface evidently draws from human computer interaction principles. For that reason, the main chapters are structured all the same: First, they regard the overall widget interface. Second, they expand of the KG disambiguation measures. And third, they consider the UI. Themselves, the previously stated research questions resemble this structure. This work closes with a comprehensive discussion, including to a critical view on itself, as well as ideas for future work. A concise conclusion ends the work.

1.5 Contributions

The encompassing contribution of this work is a microservice that integrates with the Open Research Knowledge Graph (ORKG) ecosystem. Throughout, this service is referred to as OSyRIS (ORKG Systematic Review Integration Service). The service is contemplated by the following isolated contributions: First, a web service specification integrating with ORKG and arbitrary SR Services. Second, a high-level algorithm that fosters KG disambiguation in the targeted scholarly KG. And third, two User Interface (UI) components that support users with the disambiguation mapping task.

Data Availability The source code related with work is available online. Experiment related code resides in a University data repository under `https://data.uni-hannover.de` [11]. Service related code resides in an ORKG *GitLab* repository under `https://gitlab.com/TIBHannover/orkg` [12].

Chapter 2

Background

This chapter outlines knowledge required to understand the remainder of this work. The topical range spans the technical and contextual frame for the work. Computer science fundamentals are presupposed, as the background connects beyond.

2.1 Knowledge Graphs

By definition, a KG is merely a graph model of concepts in the real world. More precisely, nodes represent entities in the modeled domain, whereas edges depict relations among them. As a KG usually serves a specific domain, it maintains an accordingly specific scheme, also referred to as ontology. At that, it defines meta concepts like classes and properties. An ontology is technically a minimum part of the KG itself. At any point in time, a KG can be traversed to infer knowledge over the concepts [3].

There are different modelling specifications for KGs. Most common are directed edge-labelled graphs (*del graphs*), heterogeneous graphs and property graphs. They differ in how data is attached to nodes and edges. RDF (Resource Description Framework (RDF)) is the World Wide Web Consortium (W3C) declared standard model for KGs on the web. It bases on directed edge-labelled graphs, as facts are represented in a subject-predicate-object relation. Mathematically speaking, facts are triples. From a graph perspective, subjects and objects represent nodes, whereas predicates represent directed relationships. Subjects and objects can conceptually describe entities on the web. For this, they are depicted by Internationalised Resource Identifiers (IRIs) – a superset of Uniform Resource Locators (URLs). Objects, however, can also depict literals, i.e. serialisable information, such as numbers or strings. The

optional RDF schema presents a minimum ontology for KGs[1] [14] [3] [15].

RML (RDF Mapping Language (RML)) — superset of the W3C *R2RML* specification – is a data store mapping language. It can be used to describe constraint mappings from different types of relational data to anRDF representation. Unlike *R2RML*, it can not only source from relational databases, but also from files (e.g. *JSON* or *CSV*). At that, it is commonly implemented with data integration pipelines on KGs [16].

SQL (Structured Query Language) has been the longtime de-facto standard for relational database querying [17]. However, KGs represent graph-based databases. This is, querying KGs bases on different fundamentals. In order to draw from known concepts, RDF KGs were designated the *SQL*-syntax influenced SPARQL Protocol And RDF Query Language (SPARQL). SPARQL reuses popular *SQL* keywords, but uses RDF graph patterns for stating query conditions [18].

2.2 Systematic Reviews

A Systematic Review is both a methodology as well as the such raised artifact (a paper in general). It aims to answer a predefined research question in a systematic, unbiased way using existing publications [5], [6]. In general, reviews assess original work contributions with respect to the upfront defined question. As a result, extracted contributions are compared in order to compile a scoped state-of-the art [19]. From this, further procedures can connect, whether it is research, development or policy making. SRs are most popular in medical and healthcare domains. For instance, a systematic review in the healthcare domain could compare quality of different vaccines to aid clinicians [20]. Although there is no standard approach to SRs, literature sketches a few common steps [5], [6], [19]–[21]:

Preparation Definition of a constructive research question. Supplementary, definition of in- and exclusion criteria upon which studies will be respected.

Search Strategic search for studies supposedly answering the research question. Application of the in- and exclusion criteria.

Screening Critical appraisal of each included study's quality. Elimination of studies that do not meet specific quality criteria.

[1] *Turtle* is a language and file format for expressing RDF. It is often times used synonymous with RDF. Throughout this work, *Turtle* is implied with explicit KG listing [13].

Synthesis Extraction of relevant study contributions. Comparative encoding of contributions in a suitable format, e.g. a table.

Reporting Compilation of the results, including to the review context, synthesis and protocol. Transparent justification of the review process.

2.3 Clustering

Clustering is a fundamental technique used in the machine learning domain. Given a set of data points, clustering algorithms are ought to identify structural relationships. The different groups can be considered clusters of semantically related data points. Clustering helps to identify patterns in data for arbitrary purposes. As a visual example, clustering is a vital part of digital image processing (such as with segmentation). As a more abstract example, clustering is also heavily deployed with data mining algorithms. Given great magnitudes of data points, clustering algorithms can help to detect certain dynamics. Dynamics could describe categories, anomalies or distribution phenomena [22]. Clustering represents an unsupervised approach to learning. This is, clustering algorithms handle patterns in the data automatically. Clustering algorithms do moreover differentiate in whether the number of clusters is automatically derived or must be defined beforehand. [23]. Clustering algorithms classify as one of the following: Hierarchical clustering algorithms, which recursively assemble clusters based on global similarity metrics. The cluster size is therefore dependent on the depth of recursion. Partitional clustering algorithms, which iteratively assemble a fixed number of clusters with increasing specificity. Obtained clusters do not overlap. Density-based clustering algorithms, which assemble clusters based on spatial data point density. In this, clusters can have arbitrary shapes. In particular, outlier data points can be assigned a designated noise cluster [24].

2.4 Representational State Transfer

Representational State Transfer (REST) is a widely used architectural paradigm for HTTP-based Application Programming Interfaces (APIs). For it being stateless, REST-APIs can scale horizontally without further ado. REST alienates the anatomy of an HTTP request in favour of a uniform communication meta protocol [25]. Most basically, the HTTP verb defines the type of operation requested. The pathname of the request URL states the associated resources which are addressed. It can optionally include an Identifier (ID) to address a specific one. Also, the query string

can optionally provide modifier arguments (e.g. a resource limit). Resource specific data can be provided with the request data (body). The dominant messaging format among REST-compliant services is JavaScript Object Notation (JSON) [26]. Upon response, the HTTP status codes depict the result of a resource manipulation (e.g. `201` for resource created) [27]. Table 2.1 summarises the basic REST interface operations.

HTTP Verb	REST Operation
`GET`	Request resource
`POST`	Create resource
`PUT`	Modify existing resource (total)
`PATCH`	Modify existing resource (partial)
`DELETE`	Delete existing resource

Table 2.1: Basic REST methodology. The HTTP verbs are used to describe an interface for resource related operations.

2.5 Custom Web Components

Nowadays, most application UIs are web-based and build on web technologies. In fact, even desktop and mobile applications have increasingly integrated with the elaborate web technology stack [28]. The triad of the Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS) provides languages driving web applications. At that, HTML specifies a set of atomic-purpose UI structure elements. Rich UIs combine systematically deployed components of hierarchically nested HTML elements [29]. Yet, such components do violate basic software engineering principles. Particularly, reusability and information hiding are unachievable. As a result, common web views implement a proposed standard for custom web components that emulate ordinary HTML elements. A custom web component is represented by a JS class that inherits from the generic `HTMLElement` class. The class defines its behaviour. Each instance of an element injects an isolated Document Object Model (DOM) into the document, a so-called *Shadow DOM* [30].

2.6 Web Applications Widgets

On the web, a widget is a modular application with a UI that allows for embedding into arbitrary third-party applications. Widgets isolate development and mainte-

nance at a single provider [31]. Widget-based interoperability represents the ubiquitous approach for scaling interoperability across web service UIs. Most efficiently, widgets come with a minimum integration effort. Host applications that do not require any interaction simply embed widgets using the *iframe* HTML element [32]. For programmatic interaction, widgets exploit the possibility to reference cross site scripts in HTML documents [33]. For presenting a UI, in particular, widgets are subject to usability concerns upon design time. Usability describes the degree at which an interactive system can be used by specified users in a specified context. Evaluation of widget quality is thus mostly evaluation of its usability [34].

Chapter 3

Related Work

Knowledge engineering has been a popular subject in recent years. Evidently, a lot of work more or less relates with the approach shared in this work. Yet, this work does not only focus on knowledge engineering, but also other disciplines related with engineering a web-based interface. An extensive outlook on related work is hereinafter presented and connected with the thesis.

3.1 Open Research Knowledge Graph

The benefits of KGs for scholarly communication are obvious. For this reason, scholarly KGs have been a priority topic for the *TIB – Leibniz Information Centre for Science and Technology (TIB)*. In 2018, Auer, Kovtun, Prinz, *et al.* laid out first steps towards a KG for the scientific domain [4]. Successively, Jaradeh, Oelen, Farfar, *et al.* presented the Open Research Knowledge Graph [35]. From different perspectives, premises and feasibility were then assessed [35], [36]. As a result, the ORKG emerged, a scholarly KG developed and maintained by the *TIB*. ORKG is open source and serves open science [37], [38]. Thoroughly reviewed, ORKG reveals as a dominant representative of scholarly KGs. Instead of modelling scholarly communication extensively, ORKG is strictly focused on scientific artifacts to obtain simple state-of-the-art accessibility. For this, ORKG is constrained on the successive ontology. Figure 3.1 pictures a *del graphs* representation of an exemplary ORKG instance defined across the presented ontology.

Scientific contributions describe the actual value of research. They comprise information that is related to a research objective. Contributions thus help with practical problems or advance the state-of-the-art for a specific research subject. Accordingly, ORKG defines the class `http://orkg.org/orkg/class/Contribution`

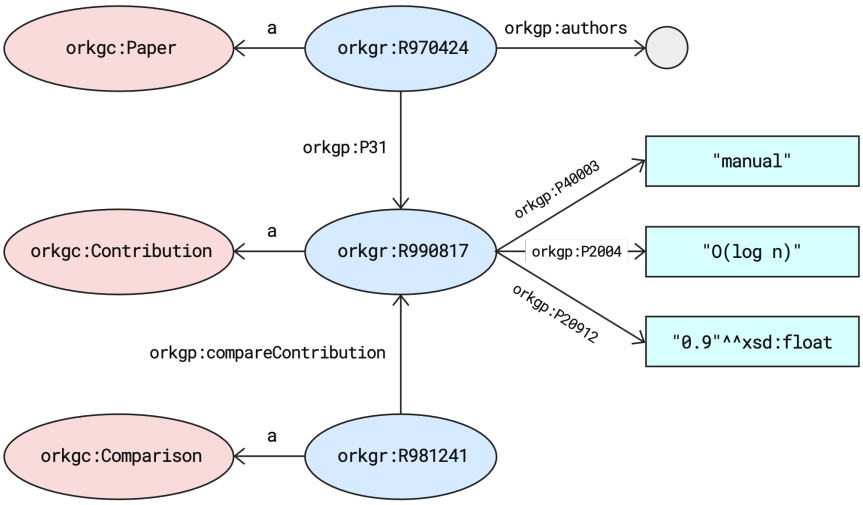


Figure 3.1: A minimum example of entity and relationship instances integrating the artifact-reduced ORKG ontology. The red ellipses describe schematic class entities, the blue ones individual resources. The cyan boxes describe literal objects. The arrows depict the relationships across resources.

(`orkgc:Contribution`) [39]. By frequency, papers describe the default artifact of scientific publications. Besides contextual information, papers comprise a set of contributions raised for a specific research question. ORKG defines the class `http://orkg.org/orkg/class/Paper` (`orkgc:Paper`) as a parent for multiple resources that are each classified a contribution [39], [40]. For the sake of highlighting a state-of-the-art, ORKG also describes a specific class `http://orkg.org/orkg/class/Comparison` (`orkgc:Comparison`). It aggregates topically related papers in order to describe a paper- and contribution-based comparison. In the ORKG-UI, comparisons are displayed as tables. Figure 3.2 shows a screenshot of a random comparison from ORKG [39], [41].

This work targets a service set to create data in a scholarly KG. ORKG marks the designated scholarly KG upon which the described implementation and evaluation are conducted.

3.2 Knowledge Graph Integrity

Whenever an entity or relationship is added to a KG, conceptual ambiguity could arise. This means, an already existing entity or relationship, respectively, could already refer to the same real world concept. As KGs usual scale unsupervised,

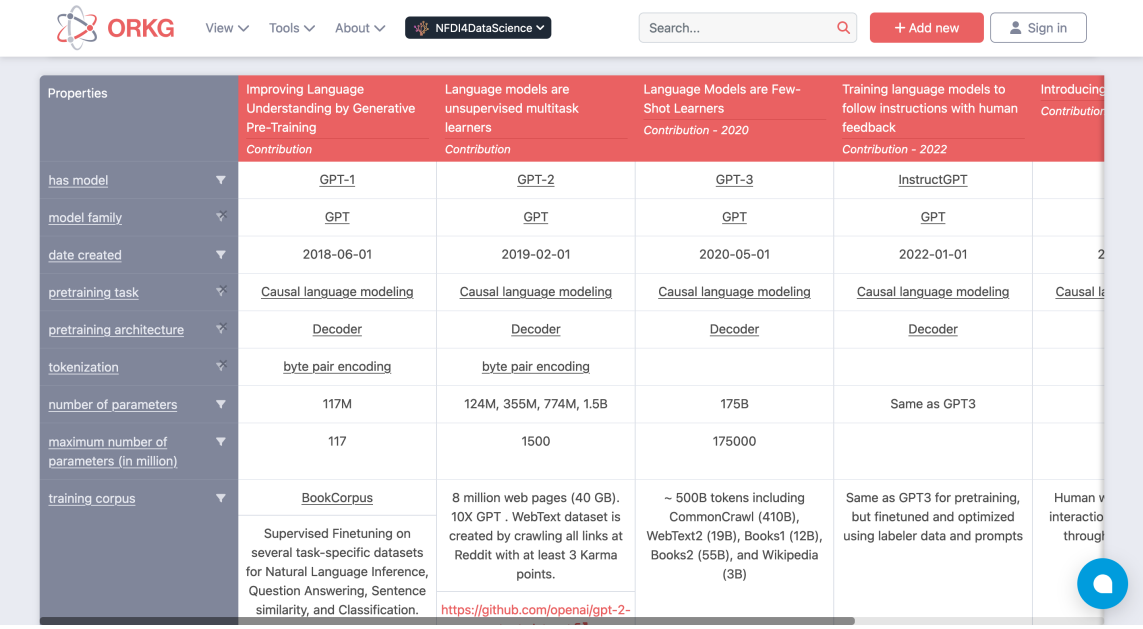


Figure 3.2: Screenshot of a random comparison from ORKG. The compared paper resources head columns, whereas the contribution criteria head the rows. This is, the cells represent individual contributions.

this is a frequent phenomenon. A major research topic on KGs concerns measures to mitigate conceptual ambiguity. Or, with other words, to optimise the overall conceptual integrity of KGs. According research has diverged in two directions.

Alignment Most research regarding integrity of KGs is based on available KGs. This is, given a single or multiple such graphs, measures seek to align already integrated concepts for uniqueness. For this, semantic similarity of entities and relationships is computed to highlight structures that likely describe the same concept. Proposed measures can be used in production in order to periodically prune highlighted duplicates, or fuse entire knowledge graphs. As first steps towards alignment techniques, Traverso, Vidal, Kämpgen, *et al.* define a similarity metric for KGs. At that, similarity is computed over scoped congruence in the graph structure [42]. Similarly, Zhu and Iglesias define graph based similarity measures with a focus on path finding and ontological hierarchy [43]. More progressively, Li, Zhang, and Zhang applied an active learning strategy to detect similarity based on entity related word vector embeddings [44]. Additionally, Zhu, Bao, Han, *et al.* proposed an alignment model based on a graph convolutional network that embeds entities directly into a

vector space [45].

Disambiguation The other perspective on KG integrity regards the data integration process. Herein referred to as disambiguation, possible ambiguities are detected online, i.e. at time of expected data creation. However, disambiguation evidently relies on a specific reference context. Similarity computation requires some truth about the integration data. While alignment techniques share specific KGs as a truth, disambiguation techniques depend on different contexts. Collarana, Galkin, Traverso-Ribón, *et al.* describe an approach that connects with the alignment approach: First, RDF triples are constructed from a heterogeneous data source in order to obtain an intermediate KG. Subsequently, the intermediate and the target KG are merged using alignment techniques [46]. In his thesis, Oghli described a comprehensive ORKG pipeline for recommending predicates based on paper full-text. Initially, similar papers are clustered as a reference model. Incoming papers are embedded into a *TF-IDF* vector space in order to be compared with likewise represented paper clusters. As a result, batches of predicates give a hint for what predicates could apply for the incoming data [47]. The ORKG SimComp API provides an interface to compute similarity of scientific documents. It is implemented in the ORKG frontend application [48], [49]. The SimComp API bases on Elasticsearch. Elasticsearch is also used in the ORKG frontend with the manual contribution creation UI. Elasticsearch is a document-based search and analytics engine. It uses the *BM25 (Best Matching 25)* scoring algorithm. As with the document-based nature, the *BM25* algorithm is mostly progressive measures [50].

The targeted service will not have periodic maintenance capabilities in scope. Immediate integrity measures will be required online. Implemented integrity measures should therefore be bound to disambiguation, rather than alignment techniques. The data context is evidently the one given by related SRs.

3.3 Systematic Review Services

In general, each major step in an SRs (see 2.2) could be supported by a different application. Designated SR Services, however, support users at a sequence of steps. A comprehensive review has revealed a landscape of SR Services. An integration interface between an SR Service and a scholarly KG would connect as late as at the extracted or synthesised data. For that reason, five services – representative for synthesis-inclusive SR Services – are examined and compared. The comparison is deliberately addressing data export formats. Other properties are

not relevant, as (partial) services that do not collect export-ready data do particularly not provide KG mappable data. The compared services include to **CADIMA (CAD)** [51], [52], **Covidence (Cov)** [9], [53], **EPPI-Reviewer (E-R)** [54], [55], **PICO Portal (PIC)** [56], [57], and **SysRev (Sys)** [58], [59]. With one exception, all services provide export functionality for tabular formats of both extracted and compared (synthesised) data. Table 3.1 compares the export capabilities that the review revealed for the stated SR Services.

Export	**CAD**	**COV**	**E-R**	**PIC**	**Sys**
References	yes	yes	yes	yes	yes
Formats[1]	*RIS (TXT)*	*CSV, RIS*	*RIS (TXT)*	*CSV, RIS*	*CSV, JSON*
Data Extraction	yes	yes	yes	**no**	yes
Formats[1]	*XLS*	*CSV*	*CSV*	-	*XLS*

Table 3.1: Comparison of popular Systematic Review Services with regard to offered file export functionality. Except for one case, all reviewed services offer tabular file exports for both data extraction and synthesis sheets.

The targeted service should be agnostic towards SR Services. Knowledge about minimum data guarantees across the majority of SR Services is thus required. A common tabular representation promises useful.

3.4 Density-based Clustering

In 1996, Ester, Kriegel, Sander, *et al.* proposed the *DBSCAN (Density-Based Spatial Clustering of Applications with Noise)* clustering algorithm. Inherently, this introduced a novel, density-based approach to the clustering problem. *DBSCAN* recursively clusters data points that are related by a spatial proximity-based neighbourhood. This is represented by a predefined neighbourhood parameter ϵ. Hence, basic knowledge about data characteristics is required. Results might particularly be poor on unseen data [60]. To address these limitations, *DBSCAN* has been advanced in various ways. Ankerst, Breunig, Kriegel, *et al.* proposed *OPTICS (Ordering Points To Identify the Clustering Structure)*. *OPTICS* applies linear ordering to the data points in order to monitor reachability distances. Reachability gaps can be exploited to obsolete a predefined ϵ parameter. As a result, density levels among different

[1] Multiple file formats that serve a common general purpose were combined into a single representative. For instance, *RIS* includes *BIBTEX*, and *CSV* includes *XLS*.

clusters can vary [61]. Moreover, McInnes, Healy, and Astels proposed *hdbscan (Hierarchical Density-Based Spatial Clustering of Applications with Noise)*. Literally, *hdbascan* adds hierarchical levels to *BDSCAN*. It iterates over different ϵ to detect clusters of various density [62].

The family of *DBSCAN* clustering algorithms dominates the density-based clustering literature. Worth mentioning, Comaniciu and Meer presented a likewise popular algorithm that build on a different approach: The *MeanShift* algorithm iteratively shifts data points towards the mode of data point density regions. Clusters correspond to the regions of the same gravity upon shift convergence. In contrast to the *HDBSCAN*-family, however, *MeanShift* is not robust against noise, as it detects (hyper)ball-shaped clusters like partitional clustering algorithms [63].

The targeted service should detect coherency in SR and KG data to improve disambiguation. In particular, this should outperform a label only-based mapping approach.

3.5 Web Applications Widgets

Widget development is mostly an engineering discipline. They are commonly deployed to supplement the organic content of a website. Popular examples are embedded *Google Maps* or *Instagram Posts* [64, 65]. In fact, ORKG already provides a widget for linking ORKG paper entries for given DOIs [66]. Complementary, widgets that require a shift of attention are usually represented by modal UIs. Modals represent UI windows that overlay the primary content of a website. At that, they block it to focus the widget-respective secondary task [67]. Modal widgets are common with outsourced communication or data submission interfaces [68].

Widgets ultimately represent an exploit of web technologies. Hence, the scientific perspective is sparse. Paulson, however, laid out fundamentals for interactive widget development. The `XMLHttpRequest` methodology was therefore described for website on-page communication [69, 70]. Han and Park presented usability experiments of widgets on the web. Their key results highlight the importance of making the purpose of widgets unduly explicit. Contexts of integration are unknown and might not describe a clear purpose [71].

The targeted service should uphold ease of integration for SR Services providers. The widget solution is uncontested in this regard. Also, it allows for presenting the SR Services users with a UI from the scholarly KG's end.

3.6 Information Retrieval User Interfaces

Looking at UIs, information retrieval applications mostly embody browser views. This means, users are provided with a technically indefinitely long list of results (as many as exist). Importantly, results are ordered by their degree of fulfilling a declared information need [72]. Aside, computation of such a degree – commonly referred to as score – is not trivial. It is especially application specific. The ordering holds implicit and exclusive affordability about the presented objects. Jen-Hwa Hu, Ma, and Chau designed UIs for information retrieval tasks with regard to cognitive psychology theories. Their evaluations showed that UI elements that specifically tie with the information retrieval aspects outperform bare list-based elements. This is, results should explicitly describe their informative degree. Also, they supported higher retrieval quality and satisfaction for novice users [73]. Rather unrelated, Schneider, Weinmann, and Brocke defined a design cycle for nudging users towards a certain behaviour on UIs. It yields, options that are desired by a provider should be selected by default (highlighted) and showed with primacy (first) [74]. In that regard, their findings can be transferred to information retrieval UI design.

The targeted service should provide human users with scored mapping options for their SR criteria. A problem-specific UI that supports users – of which the majority can be expected to have no experience with the task – with the mapping task.

Chapter 4

Approach

Needless to say, breaking down the rather complex problem helps with finding and describing an appropriate solution. Each section is met with a topically related approach. First, this means a software and web engineering approach. Second, a machine learning and information retrieval approach. And third, it is for a human computer interaction approach.

4.1 Service Interface

A reliable first step for refining an interface between SR Services and ORKG is to identify the key requirements. Least to say,the dominant functional requirement regard the data exchange. Two dominant non-functional requirements, on the other hand, pertain to the SR Service agnosticism, and the KG integrity. Incrementally, minimum consistencies identified across SR Services are hereinafter adopted a comprehensive service infrastructure.

4.1.1 Data Guarantees

Most important for architectural decisions are the data guarantees that SR Services provide all alike. This means, for ORKG to read review data, a generic model over the collected data needs to be abstracted. Contrary to initial expectations, however, SR Services maintain a simple and ubiquitous model. This is due to the fact that all services replicate the common SR methodology. On the other hand, the ORKG ontology is narrow on papers (`orkgc:Paper`) and contained contributions (`orkgc:Contribution`). The majority of reviewed SR Services provide tabular data export functionality. Exports span paper metadata and the contribution extractions

or synthesis (see 3.3). Exported tables are accordingly equivalent structurally with ORKG comparisons (`orkgc:Comparison`).

4.1.2 Scope Guarantees

SR Services usually offer data exports in separate units (e.g. only literature references, data extraction or synthesis). Regardless, it is evidently available entirely at SR Services. This is, once extracted or synthesised data has been collected, it could be merged into a comprehensive relational representation. Such a representation would be contributions mapped over respective papers and unifying criteria labels. A merge could happen in an arbitrary web document [75]–[77]. The SR data must, however, be complete. The scope is therefore tied with documents at the final stages of SRs (data synthesis or reporting).

4.1.3 Key Requirements

Two non-functional requirements are imperative with the problem statement. Between a motivated SR data export and the data being written to ORKG, the disambiguation measures should take effect. Specific measures are expanded below (see 4.2). Yet to be approached first, interoperability with an arbitrary SR Service is sought at low integration costs. Interoperability of web services is bound to take one of two fundamental approaches. For one, different providers develop their services independently, but with respect to an agreed communication interface. The overall purposeful logic is more or less evenly distributed. Such symmetric interoperability is suited for situations that require bilateral code ownership (cooperative expertise). However, it does not scale organically with additional services. The alternative is to accumulate the entire logic at one end. This is especially reasonable if the interoperability is more important from the respective end. Asymmetric interoperability is widely adopted with shared UIs, called widgets.

Web applications at scale contemplate distributed systems. Although presenting a single UI, they maintain on a pool of individual instances in the background. The distribution approach originates from the need to scale on hardware. However, increasingly product-driven development methodologies have also connected with distributed system from an architectural perspective. Strongly cohesive functionality is therefore encapsulated to modular sub applications, also referred to as (micro-)services. ORKG started with a monolithic backend application, but has recently transitioned towards a microservice architecture [48], [78]–[81].

Advantages of asymmetric interoperability, as well as the ORKG macro archi-

tecture, lead to solve the SR Service integration problem with an individual microservice. For simplicity, that very service will hereinafter be referred to as OSyRIS (ORKG Systematic Review Integration Service). From a software engineering perspective, declaration of a reusable service interface is a constructive software quality measure. And so is the resulting horizontal scalability. Also, hiding the bulk integration logic behind the interface advances the quality. The modular approach simplifies development and maintenance. It is henceforth solely reliant on ORKG internal decisions.

4.1.4 Architecture

The previous findings allow for a sufficient architectural model of the envisaged service. The service communicates with SR Services and the ORKG backend on demand. The ORKG backend application maintains the actual underlying KG. By technical specification, it is represented by a *Neo4j* graph database. *Neo4j* uses a *property graph* model [82], [83]. Yet, the backend makes data accessible in a semantic web-compliant RDF representation (*del graph*) by a SPARQL endpoint [84]. OSyRIS can use the SPARQL endpoint API to query the underlying KG, rather than working directly on the *Neo4j* instance. Other services could be used on demand, but are not required. To be consistent with a distributed system, OSyRIS should be deployed through the encompassing virtual ORKG frontend interface. Figure 4.1 schematically describes the macroscopic component and implicit interface view of ORKG. In contrast, figure 4.2 describes the microscopic component and interface view of OSyRIS. Interfaces are thereby the remote communication relations. Relevant required interfaces of the ORKG backend are also shown.

4.2 Knowledge Graph Disambiguation

The major non-functional requirement is to provide online KG disambiguation. Incoming SR data exported from an SR Service to ORKG must first serve the ontology (see 3.1). An according syntactical classification is the evident first measure to be applied. Subsequently, the disambiguation measures can take place, based on the virtual resources.

4.2.1 Entity Mapping

Papers and contributions in ORKG are represented by entities. In order to relate papers and contributions, ORKG defines a designated predicate. On that account,

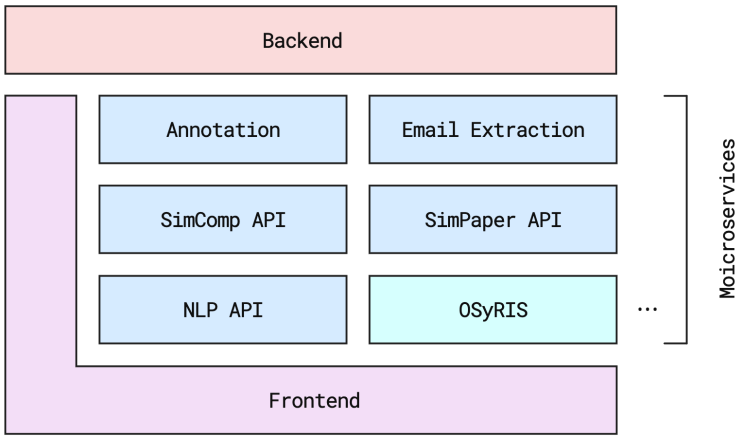


Figure 4.1: Schematic macro-architectural component and implicit interface view of ORKG. The frontend component virtualises the backend and microservices for a single User Interface. Technically, each component can communicate with another or clients individually, but the human UI is the frontend component itself, which hosts the frontend application.

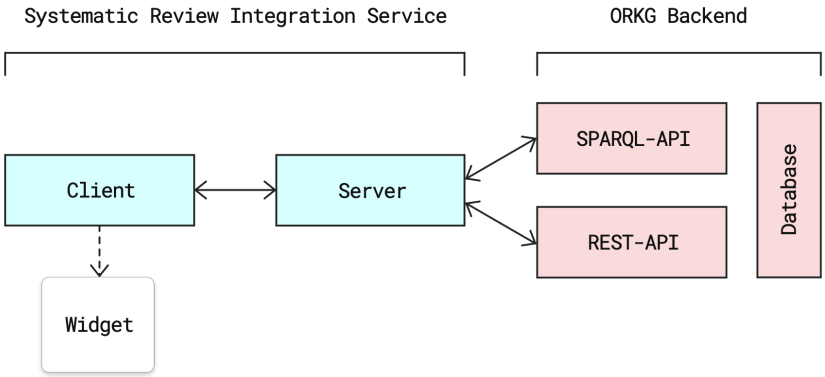


Figure 4.2: Schematic micro-architectural component and interface view of OSyRIS. The arrows depict remote communication relations. Notice that the application is divided: A client component runs in an SR Service, i.e. a web client. The server application is the actual microservice application. It communicates on demand with the ORKG backend API. The database, including the actual KG, is maintained by the backend application in isolation. Notice that the client application injects the widget UI.

papers are just associated with meta data and a number of contribution entities.

Single quantitative and qualitative data is in point of fact encapsulated in individual contributions. The tabular data retrieved from an SR Service contains a set of papers and associated contributions. For the matter of fact, these contributions describe rather atomic parts of a contribution in total. Having said this, for each paper in an SR, a single `orkgc:Paper` and `orkgc:Contribution` are defined and interrelated. As a pleasant fact, disambiguation must merely discover whether a paper has already been represented in ORKG. Having a DOI as a global ID, the disambiguation is a simple DOI-based query. Contributions are inherently comparable on the ambiguous entity. To get an idea, listing 4.2.1 shares an exemplary RDF encoding of parts of an incoming SR.

Listing 4.1: Example RDF triple representation for a small SR. Three literal contributions are given and comprised within a broader contribution. Only one predicate is defined anew, while two were already existing in the graph and are thus reused. All created resource IDs are illustrative and would require an automatic assignment according to an internal strategy.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX orkgr: <http://orkg.org/orkg/resource/>
PREFIX orkgc: <http://orkg.org/orkg/class/>
PREFIX orkgp: <http://orkg.org/orkg/predicate/>

orkgp:P20912 a rdfs:Property ;
            rdfs:domain orkgr:Contribution ;
            rdfs:range xsd:float ;
            rdfs:label "f1 measure" ;

orkgr:R990817 a orkgc:Contribution ;
             rdfs:label "Contribution 1" ;
             orkgp:P40003 "manual" ;
             orkgp:P2004 "O(log n)" ;
             orkgp:P20912 "0.9"^^xsd:float .

orkgr:R970424 a orkgc:Paper ;
             rdfs:label "Example Paper" ;
             orkgp:authors [
```

```
21                a rdf:Bag ;
22                rdf:_1 "Example Author 1" ;
23                rdf:_2 "Example Author 2" .
24            ]
25            orkgp:P31 orkgr:R990817 .
```

4.2.2 Predicate Mapping

Relationships in ORKG only describe individual contributions. These are literals associated with papers. Except for ORKG internal meta data, contributions itself are free from ontological constraints. In general, contributions can associate with literals, or nested entities that describe more complex contributions. However, as SRs share manually encoded contributions, they are true to a flat literal representation. This is, each contribution in the SR data can be described by a literal and related with the beforehand created or reused contribution entity. The relationship is thereby a predicate that embodies the respective SR criteria label. In contrast to papers, it is likely that such criteria already exist with predicates in ORKG (think of *F1 score* as a common metric to evaluate information retrieval systems). Unlike papers, however, contribution criteria do not hold uniquely identifiable features (such as a DOI). That being said, predicate mapping represents the bottleneck for the disambiguation measure. The remainder of this section is concerned with approaching the disambiguation with a predicate mapping algorithm. Deliberate respect is due to the context sketched by SR data.

Terminology First it is important to state and describe a regular terminology for the disambiguation approach. The following terms will be used throughout this work:

Strong Truth The strong truth is the criteria (to-be predicate) label and the related set of literals provided with incoming SR data.

Weak Truth The weak truth is the set of predicates, and furthermore their respective set of literals, that already exists in the KG.

Candidate A candidates is a weak truth instance that represents a possible disambiguation mapping, including to predicate label, predicate id and all associated literals in the KG.

Key Assumptions A few assumptions with incoming SRs and KGs in general are evident. They favour a successive construction of a predicate disambiguation algorithm:

A1.i Different concepts across the truth could be labeled with homonyms (e.g. *kappa* for *Cohen's* κ and *Feiss'* κ).

A1.ii A single concepts across the truth could be labeled with synonyms (e.g. *predictor variable* and *exposure variable* for *Independent Variable*).

A2.i The weak truth is not always, but at least for the most part, conceptually correct; i.e. could contain a fraction of outliers (noise).

A2.ii The strong truth is conceptually correct.

A3.i The weak truth could contain different valid representations (e.g.).

A3.ii The strong truth is in a uniform representation (e.g. only values from the interval $[0, 1]$ for *F1 Score*, but no percentage value from $(1, 100]$).

A naive approach would compute mapping scores based on label similarity. This is, it would apply a suitable string similarity metric. In fact, a label only-based approach is currently implemented in the ORKG frontend application at different occasions (e.g. for predicate creation within the manual contribution editor). Yet, label only-based scoring is vague. First of all, because strong truth labels for a single concept likely differ with different human labelling preferences. Second, because linguistic ambiguities represent an insuperable obstacle (**A1.i**, **A1.ii**).

If not for a unique ID, mapping of predicates can rely on the context given by SRs. In fact, the strong truth gives a condensed certainty: A value type and range that literals associated with a similar predicate from the weak truth also associate with (**A2.i**, **A2.ii**). An evident first step to master the situation is thus to reduce the weak truth to only the predicates that are dominated by the same datatype literals as the strong truth. For the next step, a nearby approach imposing on the problem would try to simply view truths as a collection of documents. Thereby, each of the documents could be embedded to a vector space, e.g. using the *TF-IDF* model. Using a distance metric, such as cosine similarity, the closest weak truth to the strong truth vector could be computed and traced back to the respective predicate. However, the big picture on the weak truth is not necessarily reflecting the small strong truth sample (**A3.i**). Prospective experiments supported supposedly poor performance. Technically, a literal only-based scoring needs to spot whether

the strong truth is represented by the weak truth. However, the weak truth can picture heterogeneous groups of literals. A fundamental step is thus to detect a cluster in the weak truth that shares the characteristics of the entire strong truth (**A3.ii**). Finding an arbitrarily shaped cluster in noisy data dictates the use of a density-based clustering algorithm. Matching the requirement by name, the *DBSCAN* family of algorithms represent according clustering behaviour. Transductive clustering algorithms like *DBSCAN* are dependent on basic knowledge about the data they work on as they rely on spatial reference parameters. There are a plethora of variations of the original *DBSCAN* algorithm that mitigate that limitation. For the case, however, the strong truth favourably helps with this issue. The parameters for *DBSCAN* can be estimated well using simple averaging metrics on the strong truth literals. More precisely, the ϵ parameter for *DBSCAN* could for instance be the standard deviation of data points in the strong truth. To induce some tolerance, a buffer coefficient could be applied. The neighbourhood size can moreover be set upfront to enforce some degree of cluster cohesiveness. Also, the data point to start the first cluster expansion from can be picked closest to the strong truth mean. The best matching cluster is inherently expanded first. In case the expansion reveals noise, the weak truth data point next closest to the strong truth mean is used, instead. Once a cluster is retrieved, the algorithm can terminate, as the remaining data is regardless. This novel variation (reduction) of *DBSCAN* is hereinafter called *ScopedDBSCAN*. The literal only-based score is then the affiliation of the cluster inherently represented by the strong truth data and the matched weak truth cluster. Empty clusters must provoke the worst possible score. Figure 4.3 presents how different density-based algorithms cluster a series of exemplary data distributions.

Left to describe is a cluster affiliation metric. Cluster affiliation, certainly providing great space for optimisation, is for a first approach the following: The weak truth cluster should usually mark the higher volume cluster. It might come that the strong truth compiles a data range narrower than those in the weak truth cluster. A truth cluster being a proper subset of the other cluster would yield a perfect score. The less the overlap, the worse the score. Obviously, however, a strong truth can contain correct values that are beyond reflection by the weak truth. An inverted parabolic weighting can discount lower differences. This is herein abbreviated with δ. Simply viewing the overlap might nonetheless not be enough to retrieve a substantial score. For this, the degree of orthogonality between the linear regressions of both clusters can help to detect similarity of data distribution orientations. This is herein abbreviated with θ. Figure 4.4 depicts how the described behaviour applies to some ellipsoid clusters in 2-dimensional space. Cluster affiliation, tantamount to the literal only-based score, is a weighted aggregate of δ and θ.

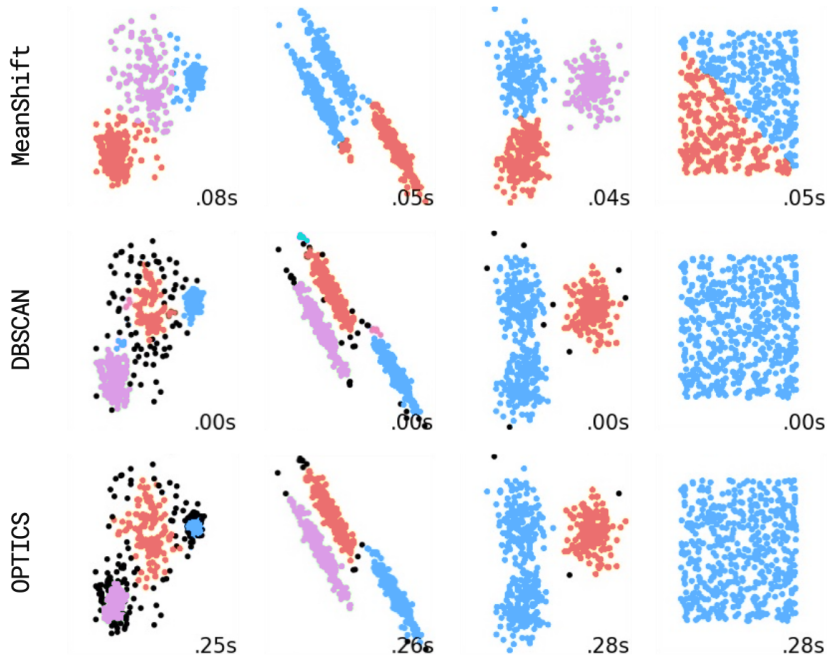


Figure 4.3: Series of data distributions and according classification results as delivered by different clustering algorithms. The black dots are designated outliers. Different colors denote different cluster labels. *MeanShift* represents a widely popular clustering algorithm for unsupervised clustering, but does not consider outliers. is the other popular choice for outlier sensible clustering, but was previously benchmarked significantly lower runtime performance than. The benchmark was performed in one sequential run, resembling the depicted order. The measured time was system time delta before and after execution of each algorithm *DBSCAN* [85]. Order and sideload effects must be considered, but the large margin gives a viable idea of the runtime order between *DBSCAN* and *OPTICS*.

It is important to highlight that phenomenons like homonyms or synonyms from linguistics can also apply to arbitrary data. Proof is given by looking at the following example: Data that describes the predicate *Recall* could equally well describe the predicate *Precision*. This resembles the homonymous label phenomenon. For this reason, a unified label and literal-based approach can greatly enhance the overall scoring performance, mutually supporting isolated mapping score accuracy. Mathematically speaking, the score is ultimately a weighted sum of label only-based scoring and literal only-based scoring. For a perfect string similarity, however, the

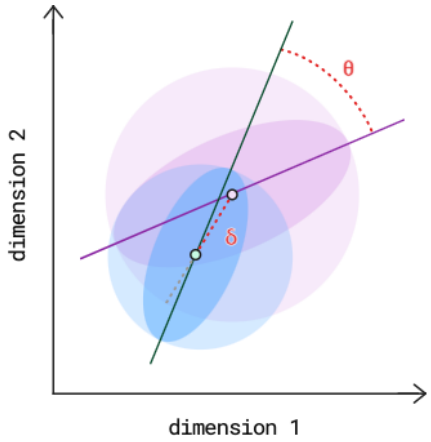


Figure 4.4: Depiction of two arbitrary clusters in a 2-dimensional space. The darker ellipsoids represent the cluster data. The lighter circles on top represent the radius which is spanned by the cluster data, respectively. The lines depict the linear orientation of the cluster data distributions, i.e. a linear regression. The affiliation between the clusters is the balanced sum of the following two coefficients: One, the unit angle between the linear regressions, reaching its maximum at full orthogonality, i.e. manifolds of $\frac{\pi}{2}$ (θ). Two, the inverted parabolic weighted cluster overlap (δ).

perfect score should be used as a perfect label only-based match is most likely a perfect match, nonetheless. There, the strengths of literal only-based mapping can be reused. To be complete, the result of executing the hereby implicitly described algorithm is a ranked and sliced list of scores. The scores are each computed on the strong truth iterated over the structurally filtered weak truth from the KG.

4.2.3 *ScopedDBSCANScoring* Algorithm

The previous paragraph highlights all procedures required to formalise a scoring algorithm for predicate mappings. Listing 4.2.3 describes the algorithm with abstract pseudo code.

$$
\begin{aligned}
ScopedDBSCANScoring &\mapsto [0,1] \\
ScopedDBSCAN &\mapsto \mathbb{R}^* \\
ClusterAffiliation &\mapsto [0,1] \\
SliceScores \mapsto [C_1, .., C_k], k \in \mathbb{N}, C &:= CandidatePredicate
\end{aligned}
$$

Listing 4.2: *ScopedDBSCANScoring Algorithm*

```
1  PROCEDURE ScopedDBSCANScoring(truth_strong, Index, k, weight_label):
2      type_dominant ⟵ argmax_type typeof(truth_strong)
3      Truth_weak ⟵ Index(type_dominant)
4      S ⟵ ∅
5
6      truth_strong ⟵ Embed(truth_strong)
7
8      FOR truth_weak IN Truth_weak:
9          score_label ⟵ UnitStringSimilarity(Label(truth_strong, Label(truth_strong)
10
11         IF score_label =1 OR type_dominant NOT IN { string, number, date }:
12             S ⟵ AppendScores(k, S, score_label)
13             CONTINUE
14
15         C_weak ⟵ ScopedDBSCAN(Literals(truth_strong), Literals(truth_weak))
16         C_strong ⟵ Literals(truth_strong)
17         score_literal ⟵ ClusterAffiliation(C_strong, C_weak)
18         score_label,literal ⟵ [ weight_label * score_label ] + [ (1 − weight_label) * score_literal ]
19
20         S ⟵ AppendScores(k, S, score_label,literal)
21
22     RETURN S
```

```
1  PROCEDURE ScopedDBSCAN(L_strong, L_weak):
2      ϵ ⟵ σ(L_strong) * (1 + TOLERANCE)
3
4      F ⟵ { argmin_l(weak) |L_weak − μ(L_strong)| }
5      C ⟵ ∅
6      WHILE |F| > 0 AND |L_weak| > 0:
7          core ⟵ min F
8          L_weak ⟵ L_weak \ core
9          F ⟵ F \ { core }
10         C ⟵ C ∪ { core }
```

```
        N ⟵ { l | l ∈ L_weak, GeometricDistance(l, core) < ϵ }
        IF |N| ≥ NEIGHBOURHOOD_SIZE:
            F ⟵ F ∪ N
        ELSE IF |C| = 0:
            RETURN ScopedDBSCAN(L_strong, L_weak)

    RETURN C
```

```
PROCEDURE ClusterAffiliation(C_strong, C_weak):
    RETURN \frac{1}{2} * [ \delta(C_strong ∩ C_weak) + \theta(LinReg(C_strong), LinReg(C_weak)) ]
```

```
PROCEDURE AppendScores(k, S, candidate)
    min_scored ⟵ min_Score S
    IF |S| < k OR min_scored < Score(candidate) THEN
        S ⟵ S \ curmin
        S ⟵ S \cup
```

A lingering open issue is that data has so far been assumed in a spatial representation. Obviously, KGs can store non-numerical literals, too (such as strings). An intermediate step requires vector space embedding of arbitrary data. Semantics-driven embedding pipelines have been thoroughly researched and can be reused at this point. Embeddings required for *ScopedDBSCANScoring* are robust against scaling effects, but solely require distances to reflect semantic relations. Embeddings could for instance apply multidimensional scaling on the complete minimum semantic distance matrix of truth literals. Besides the affiliation procedure, embedding quality can be expected great impact on the overall scoring performance.

On a final note, the weak truth data can be retrieved by different means. The most nearby approach would simply query the KG each time the algorithm is executed. Subsequently, the retrieved data would have to be embedded. For sufficiently large weak truth data, however, minor changes can be assumed to not affect mapping scores significantly. A more sophisticated approach would thus maintain a weak truth index constructed in parallel. That index can deliver embedded weak truth per datatype without further ado. Reconstruction of the index can happen periodically with a period that reflects the overall data change behaviour (for instance, every 24 hours).

4.3 User Interface

In a best case scenario, export of SR data to ORKG would expect just a single interaction, such as a button click. In this scenario, the KG disambiguation would work with perfect quality. A human-in-the-loop would not have to decide it. However, perfect quality can not be assumed. That being said, a human-in-the-loop is useful to support quality mappings. Fortunately, SR Service users are right at hand upon the export, as well as the experts about the exported data. An immediate mapping interface to utilise SR Service users as the human-in-the-loop is immediate. This leads to a widget approach that integrates a task specific view of the ORKG frontend application right to SR Services.

Within an SR Service client, OSyRIS can be integrated with a static script that is requested. Successively, the script can be interpreted by the requesting web client. Upon that, it injects the widget into the document, i.e. creates HTML elements and appends it to the global DOM. Any other communication can take place upon the ORKG default RESTful methodology. This means, once the widget is injected, the client application can perform `XMLHttpRequest`-based requests to the OSyRIS server application. The widget modals are hidden by default, but only a button is presented. The export process favourably divides the different required stages across different modals. A button click can open the predicate mapping recommendations modal. The stages least to cover are as follows. Figure 4.5 represents these stages with a sequential finite-state machine graph.

1. **Predicate Mapping and Selection** The chosen SR criteria will end up predicates in the scholarly KG. Accordingly computed mapping candidates are recommended to the SR Service user, expected to be selected or neglected (for requesting a new predicate).

2. **Authentication and Submission** The predicate strategy was approved. In order to associate the SR resources in ORKG with a specific user, the SR Service user is asked to authenticate (and possibly redirected to the registration form, if needed).

3. **Success and Links** The export was successful. The SR Service user is informed about the results and presented with links to the created resources in ORKG.

   Error An irreversible error interrupted the process at an arbitrary stage. A generic error modal provides the SR Service user with a helpful error message.

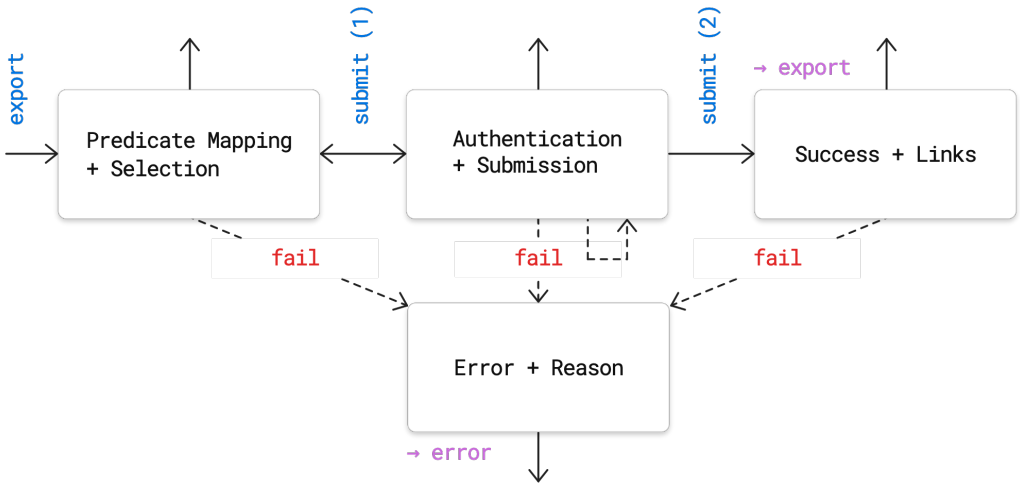


Figure 4.5: Interactive sequence of the envisaged widget modals given as a *Moore Machine*. The blue labels (vertical) describe desired interactive transitions, whereas the red labels (horizontal) describe error transitions. The reflexive error transition edge is explicitly describing failed authentication. In that case the user shall be able to retry the authentication with correct credentials. The purple labels (horizontal, indicating arrow) describe events emitted upon incoming state transition.

Technically speaking, the space of possible UIs design solutions is endless. Yet, it is a supreme heuristic to pick up the user at their prior knowledge. In general it holds, the less application-specific a UI element is, the more common it is across applications. Simple elements do thus require no more than a quick review of existing UIs – if at all. Having said this, modals of the OSyRIS widget can be anticipated to be mostly compositions of basic text and and input elements. However, for the predicate mapping task, a specific UI to support the user can be expected improved mapping quality – especially for a majority of novice users. Two elements are determined to support the mapping task.

4.3.1 Candidate Selection Component

The mandatory component is for providing selection options for the recommended mapping candidates. It is mandatory with the justified purpose of the widget. The user can actively select the mapping option they think conceptualises their selected review criteria best. This includes to their own label for insisting on the creation of a new predicate. Classified, the UI component that solves the problem resembles a checkbox. The dominant design rule for nudging users towards a desired option is

to leave it the default selection, first to appear. The default option can accordingly be the highest scoring candidate. In order not to disregard the option for creating a new predicate, however, a threshold can be applied (e.g. only preselect if a candidate score is above 0.5). For an empirical evaluation to reveal, the amount of options to present to the user is open.

4.3.2 Candidate Score Component

Showing mapping candidate options only would leave the user with estimating mapping scores. The estimate would only be an implication from the perceived option labels and ordering. For this reason, the approached widget is supplemented with an expressive score meter. The score meter can represent the score in different ways. Also a matter of empirical evaluation, whether a discrete percentage value, an analogue bar, or both in combination facilitate the mapping quality is to be revealed.

Chapter 5

Implementation

The approach has outlined non-trivial aspects towards OSyRIS, a web service that supposedly solves the SR Service and scholarly KGs integration problem. The architectural, algorithmic and UI specification have been laid out. It is now to be addressed with an implementation that allows for a proper evaluation of the approach. Moreover, the implementation gives the encompassing contribution to deploy with ORKG.

5.1 Service Interface

For the server-side service architecture, the implementation is but textbook software development. Specifically, OSyRIS builds on the same technology stack as other ORKG web services [48], [78]–[81]. For a *Python* runtime environment, HTTP(S) capabilities are provided by *uvicorn*/*gunicorn* server gateway implementations [86], [87]. Static files are served as text content from a designated directory alongside a suitable `Content-Type` header. It is important to notice that static files are served with the `Access-Control-Allow-Origin` header `*` (any) to allow access from any host machine (arbitrary SR Services). JSON represents the messaging format for the REST-API. The REST methodology is moreover realised with the *FastAPI* framework [88]. The service client communicates with the server using the *XMLHttpRequest* technology. That being said, requests are performed on demand in the background. In particular, this keeps the current web document alive (no reload) [89].

Messaging models presented in the remainder of this section are described with *TypeScript* interfaces. *TypeScript* interfaces provide a language for typed JSON schemes. Favourably, they are equivalent to both *Python* dictionaries, and JS objects. JSON keys are named according to the *JSON:API* recommendations. These

recommend alphanumeric camel case, starting with a lowercase letter [90]. The naming conventions allow JS code to use the dot notation for accessing key values. Regardless, *Python* code is limited to the square bracket access notation on dictionaries.

5.1.1 Client Model

The central data maintained in the OSyRIS client application represents the reference for the export. Upon this, recommended mappings can first be requested. The final submission moreover relates selected mappings with the reference. The reference data object is (re)written upon each motivated export. The data model describes a list of paper objects. Each paper object contains relevant meta data and a nested object that maps each SR criteria label with the respective contribution literal.

```
interface ITruth {
    [index: number]: {
        authorNames: string[];
        paperTitle: string;
        strongTruth: {
            [label: string]
            : string|number|boolean;
        };

        paperDOI?: string;
    }
}
```

5.1.2 Recommendation

The initial and essential request to the OSyRIS-API regards the mapping recommendations. Uncommon for REST resources, a recommendation is not a database object. This is, recommendations represent transient, rather than persistent resources. However, recommendation requests accordingly describe resource creation requests.

```
POST /recommend
Headers {
```

```
    Accept: 'application/json'
    Content-Type: 'application/json'
}
```

Request Model A recommendation request message contains the reference data. It is the string truth for online KG disambiguation measures. Section 5.2.1 regards the disambiguation measures. Alongside the reference data, the number of mapping candidates to be retrieved can be specified[1].

```
interface IRecommendationRequest {
    k: number;
    truth: ITruth;
}
```

Response Model A recommendation response message contains scoring information for each of the provided strong truth labels. Scoring information includes the actual score, as well as relevant details about the candidate predicate. As the service is stateless, the details are required in order to recover the mapping candidates upon submission.

```
interface IRecommendationResponse {
    [label: string]: {
        details: {
            description?: string;
            link?: string;
            resourceID?: string;
        };
        label: string;
        score: number;
    }[]
}
```

[1]In order to mitigate denial of service attacks due to huge k, the server enforces a feasible maximum. More precisely, the *ScopedDBSCANScoring* implementation preliminary assigns k the minimum of k and 10. It is nonetheless useful to enable a variable k with responses to anticipate reusability of the API.

5.1.3 Authentication

Resources in ORKG are associated with authoring users. For privacy reasons, user accounts are protected by password authentication (as usual). In order to write SR data to ORKG, the user is interrupted for providing their ORKG credentials. ORKG uses the *OAuth* authentication framework that provides stateless authentication capabilities in line with the REST methodology [84], [91]. The credentials are send from the OSyRIS client to the OSyRIS server application. The server application mediates the authentication request with the ORKG backend. An obtained access token is then passed back to the client application. The client application stores the access token for future (authenticated) requests[2].

```
GET /recommend
Headers {
    Content-Type: 'application/x-www-form-urlencoded'
    Authorization: 'Basic orkg-client:secret'
}
```

5.1.4 Submission

The submission request completes the export. As mentioned, authentication is implicit to a submission request. To the achieve this, the `Authorization` header carries the previously obtained access token.

```
POST /submit
Headers {
    Accept: 'application/vnd.orkg.paper.v2+json'
    Authorization: 'Bearer <ACCESS-TOKEN>'
    Content-Type: 'application/vnd.orkg.paper.v2+json'
}
```

Request Model A submission request message must transport the reference data, as well as mappings between strong truth labels and candidate information (if selected).

[2]An authentication request could also be performed right from the client application in order to half the traffic and leverage system response time. However, possibly desirable measures like prioritised authentication could no longer be applied.

```
interface ISubmissionRequest {
    truth: ITruth;
    mappings: {
        [label: string]: {
            resourceID?: string;
            label: string;
        }
    }
}
```

Response Model A submission response message primarily serves the purpose to inform am SR Service user with the export results. A result is primarily a success state, i.e. whether it was successful. A successful response, however, also shares a link to the the papers created in ORKG. These can later be used to create a comparison.

```
interface ISubmissionResponse {
    resourceIds: string[];
}
```

5.1.5 Integration

With the web application widget approach, integrating an SR Service with OSyRIS is as simple as declaring a script reference. More precisely, SR Services can state a script tag within the markup of documents that are designated to aggregate SR results. The script source needs to correspond to the URL of the serviced OSyRIS client application script. The script itself is is hosted by the OSyRIS server application. The reference instructs the web client of an SR Service user to request and interpret the script upon document load.

Widget Element The OSyRIS client application provides SR Services the API to adopt their local development methodology with. This means, to display the OSyRIS widget, SR Service providers are required to declare its position in the DOM. Statically, this concludes to adding the defined `orkg-export` tag to the markup. Alternatively, the widget element could be created dynamically and append to the DOM. The UI invasive export button appears at the designated DOM position.

Debugging By default, the integration is linked with the productive ORKG. In order to comfort integration debugging, the client script can be queried with the unary URL parameter `/?dev`. This instructs OSyRIS to work on the ORKG sandbox environment (`https://sandbox.orkg.org`), instead.

Data Provision Most importantly, the individually structured SR data must be accessible for the OSyRIS client application. JS implements the concept of higher order functions. At that, functions are treated like any other data object. They can thus also represent arguments for other functions [92]. To tie with the JS methodology, OSyRIS expects to be declared a callback function which returns the Systematic Review (SR) data in the declared format (`ITruth`; see 5.1.1). Upon the export is motivated – i.e. the export button is clicked – the callback is invoked by the client application. The returned data is validated upon the truth model shape. If valid, it is stored in client memory for the further process. The declaration of the function happens with adding a orkg-data-callback attribute to the widget element. The attribute value must a global (window scope) identifier for the callback function.

5.2 Knowledge Graph Disambiguation

5.2.1 Entity Mapping

When a submission request arrives, the OSyRIS server application is given already sorted out predicate mappings. In contrast, entity disambiguation can reliably be achieved unsupervised. For papers to be identified, the submitted `doi` field is used to retrieve a possibly existing resource. The REST-API does not offer a paper resource lookup service by DOI [93]. For that reason, the SPARQL endpoint is queried for the resource ID of the paper with the given DOI [94]. In case no resource exists in the KG, a new one is created (`POST`). In case a paper resource exists, a new contribution is created, as well as a statement relating it with the existing paper. Listing 5.2.1 states the DOI respective paper resource (ID) lookup SPARQL query.

Listing 5.1: SPARQL query for looking up a paper resource by DOI. The token `<DOI>` (line 6) simply denotes the variable DOI string.

```
1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 PREFIX orkgp: <http://orkg.org/orkg/predicate/>
3
4 SELECT ?paper_id WHERE {
```

```
  ?paper_id orkgp:P26 <DOI>^^xsd:string .
}
LIMIT 1
```

For incoming paper information that is not yet represented in ORKG, or does not specify a `doi` field, a new paper is created. The ORKG backend does not require a DOI field, but an arbitrary identifier that is unique with the paper in the scientific domain (usually this is the Digital Object Identifier (DOI)). For no DOI being given, a *Universally Unique ID (UUID)* is generated [95]. The syntactic data integration procedure is hidden by the ORKG backend. Appendix A displays how an RML script would be constituted to syntactically map SR data from a relational to an RDF representation. For completion, this connects with the previous RDF example.

Comparisons in ORKG can not be updated. As known from code dependency managers [96], this is a deliberate measure. Comparisons are viewed as timely snapshots of a specific state-of-the-art that could be referred to from outside resources. Updates can however be published through a new version of a comparison. Versions therefore relate comparisons with a time semantic. After all, manipulating comparisons always results in a new comparison resource.This behaviour is due to the current comparison creation paradigm implemented in ORKG.

The comparison creation paradigm, as implemented in the ORKG backend, is not REST-compliant [93]. For a proper REST methodology, a comparison would be created using a `POST` request. The comparison linked to the SR Service users is a virtual comparison in ORKG. A virtual comparison can be created with only a `GET` request given resource IDs of the papers to be compared. Those IDs are retrieved with paper creation responses in the previous step. In order to persist in ORKG, a virtual comparison must manually be saved by the user [97]. The URL for a virtual comparison of an exported SR is constructed from the paper IDs in the related submission response message.

5.2.2 Predicate Mapping

Upon an incoming recommendation request, the OSyRIS server application invokes the *ScopedDBSCANScoring* algorithm for predicate disambiguation. In fact, it iterates the execution of the *ScopedDBSCANScoring* algorithm over the provided strong truth instances. Each strong truth is therefore computed the top k mappings (k as requested). The results are finally packaged according to the recommendation response model and send back to the client application.

Embeddings

The implementation of the *ScopedDBSCANScoring* algorithm is close to the presented pseudo code (see 4.2.3). However, the algorithm is specified for an arbitrary spatial encoding of literal KG data. Quick response time is a pivotal aspect of a human directed system is to be considered. An acceptable recommendation response should not exceed a few seconds. To work as quick, the presented implementation uses only one dimensional vector space embeddings to reduce mathematical complexity throughout the further score computation process. An embedding procedure must technically be given for each group of occurring datatype that share a common similarity semantic. Data that – although typed differently – can be applied the same distance measure can be viewed as one type. For instance, the *Euclidean Distance* can used with integer and a floating point number alike. ORKG uses the *XML* schema datatypes. Those are commonly prefixed with `xsd`. The *XML* schema specifies the following considerable datatype abstractions for scientific contribution literals: Strings (`xsd:string`), decimal numbers (including integers; `xsd:decimal`), floating point numbers (`xsd:float`, `xsd:double`), booleans (`xsd:boolean`), and dates (`xsd:date`, `xsd:time`, `xsd:duration`, etc.) [98]. The different embedding procedures are as follows.

Numbers Numbers are already in a numerical, one dimensional representation by default. The embedding is thus identical with the raw data. Eventually, any embedding will be in the numerical representation. The only reasonable distance metric in 1D-space for *ScopedDBSCAN* is the absolute difference of two points.

Strings Strings represent the biggest challenge to embedding. In the end, KG string literals can have arbitrary length, but are usually rather short. That being said, they represent sentences in a natural language. To find out the semantic distances, a sentence transformer model fine-tuned on semantic embeddings is used to compute high-dimensional semantic embeddings. The chosen model is *all-mpnet-base-v2* from the *SBERT* library, which builds on *PyTorch*. *all-mpnet-base-v2* excels at general purpose semantic embeddings [99]. Subsequently, the pairwise distances are computed using cosine similarity and stored in a distance matrix. The distance matrix. The matrix is then downscaled to a 1D-space preserving the relative distances. The downscaling procedure calculates the squared distances, centers the distances and performs eigendecomposition. The first eigenvector represents the proper 1D representation. This procedure is usual for distance relation preserving multidimensional scaling [100].

Dates Dates naturally embody a universal semantic, regardless of the specific formal representation. This is, they represent a specific point in time. That being said, dates are all translated to the corresponding *Unix Timestamp* for a common numeric representation [101]. They can further be viewed as numbers and applied the numerical embedding procedure. Some dates are not specific with a point in time, but rather time frames (such as years or months). The *Timestamp* is then seen to correspond with the centric point in time within the respective frame (e.g. mid-year).

Booleans Boolean typed data spreads across just two punctual clusters. An embedding is inherent, for instance 0 and 1 as usual. The cluster affiliation would always be perfect, and so would be the literal only-based score. This is acceptable, as the datatype was matched and the score is moreover dominated by label similarity. A label only-based score is mostly equivalent. It is used directly to skip the more complex literal only-based procedure.

Default Any other abstract data type is disregarded and thus denied a literal context scoring. The label only-based scoring is used as the fallback procedure for unknown or deliberately disregarded data types.

String Similarity

The algorithm is weighted with a label only score to balance out shortcomings of isolated truth scores. The label only-based score corresponds to a unit normalised string similarity. For that reason, the string similarity implementation likely represents an important performance factor. For the successive evaluation, two string similarities are implemented. The first one is an inverse unit normalised *Levenshtein* distance (1) [102]. The second one is a decreasing frequency weighted *Jaccard* index. The *Jaccard*-based similarity is expected to emphasise a bag-of-words, rather than strict n-grams nature of semantically related labels (2) [103].

$$similarity_{levensthein}(s_1,\ s_2) = 1 - \frac{dist_{levenshtein}(s_1,\ s_2)}{max\{|s_1|,\ |s_2|\}} \tag{1}$$

$$similarity_{jaccard}(W_1,\ W_2)\ =\ \frac{|W_1\ \cap\ W_2|^{\frac{1}{2}}}{|W_1\ \cup\ W_2|} \tag{2}$$

Cluster Affiliation In 1D-space, there is but one linear orientation a data cluster can follow. For that reason, different cluster orientations are always fully coherent. The cluster affiliation procedure as presented is now merely the discounted cluster overlap δ.

Weak Truth Index

It is useful not to query the KG with every execution of the *ScopedDBSCANScoring* algorithm. Instead, it is useful to maintain an independently constructed weak truth index in memory of the OSyRIS server application. The algorithm in execution can query from the weak truth index for leveraged throughput. The index is set to be reconstructed at low traffic rates, e.g. at night, about every 24 hours. Structurally, the index is a cache dictionary with an entry for each abstracted datatype. The entries are arrays of spatially embedded literals.

5.3 User Interface

Once the OSyRIS client script is interpreted by a web client, a custom web component is defined. It is subsequently declared to the global custom element registry to resemble a normal HTML entity (`orkg-export`). The component comprises both an export button, as well as the modal overlay. The overlay is initially hidden. Upon using the export button, the overlay visibility toggles. The first modal to appear is the mapping recommendations modal. Until the recommendations are loaded, however, visual feedback that communicates the loading process is shown in-place (gray bars as usual for the ORKG frontend application). Since the service client is interpreted on the window scope, a hard requirement is not to declare any global identifiers (except for the registered element class). That being said, the client script is invoked through an anonymous function that acts like a private scope wrapper. For the same reason, the client application must not integrate third-party libraries or frameworks. Also, keyboard shortcuts are not registered for alternative button interaction. Listing 5.3 describes the abstract class interface for the custom web component. Figure 5.1 shows a screenshot of the widget predicate recommendation modal implementation deployed in a sandbox environment. Appendix B contains a screenshot for each modal.

Listing 5.2: Abstract class interface for the custom World Wide Web component of the ORKG Systematic Review Integration Service widget. Notice that the implementation provides a static helper function for translating Comma-Separated Values encoded Systematic Review data to a valid reference model representation. As most Systematic Review Services provide Comma-Separated Values exports, utilisation of the helper can reduce integration effort, once more.

```
abstract class ORKGExportElement extends HTMLElement {
  static parseCSVString(csvStr, mappings = {
    authorNames: "author_names",
    paperName: "paper_name",
  }, delim: string = ",", nestedDelim: string = ";"): IDataModel;

  private _shadowRoot: ShadowRoot;
  private dataReadCallback: () => IDataModel;
  private dataModel: IDataModel;
  private mappingsModel: IDataModel["mappings"];
  private overflowBackupStyle: string;
  private wrapperHistory: string[];

  connectedCallback: () => void;

  private readData(): void;

  private openModal(): void;
  private closeModal(): void;

  private openModalWrapper(modalId: string): void;
  private closeModalWrapper(): void;
  private navigateBackModalWrapper(): void;

  private openModalError(err: Error|string|{
    [lang: string]: string;
  }): void;

  async private toExport(): void;
```

```
30    private toAuth(): void;
31    async private toSubmit(): void;
32    private toComplete(): void;
33  }
```

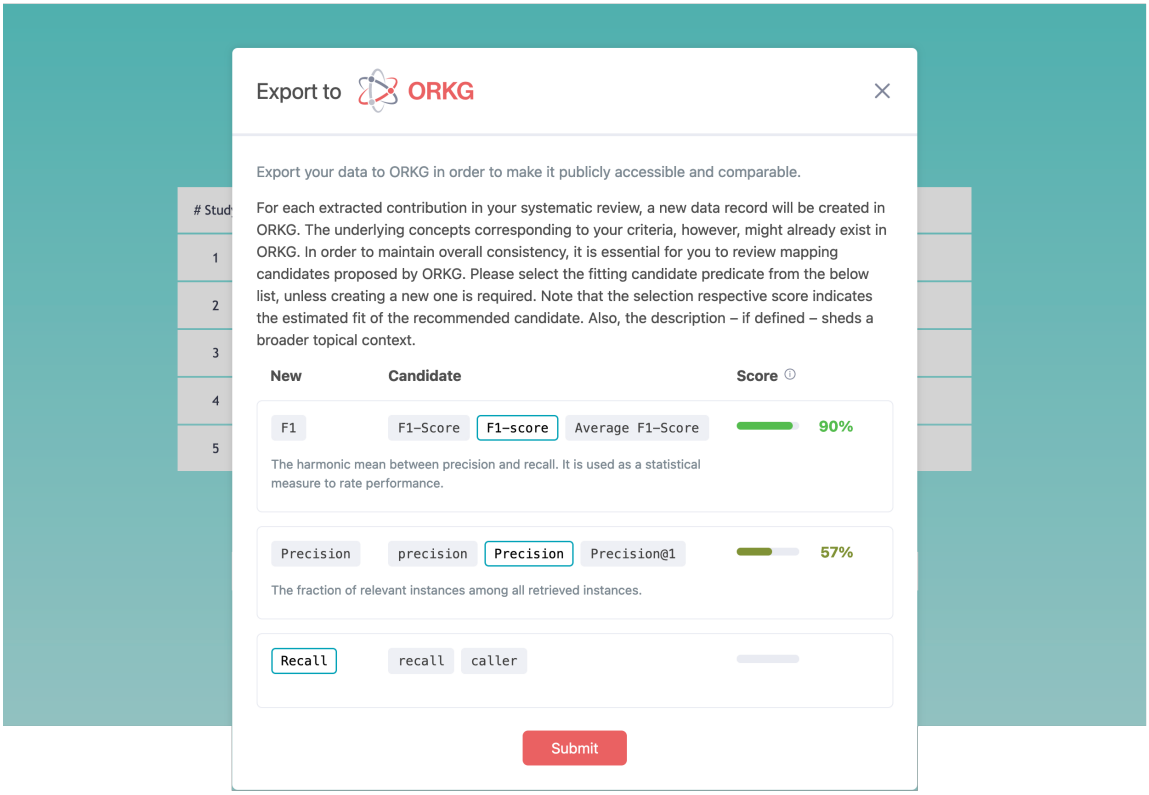


Figure 5.1: Screenshot of the widget predicate recommendation modal implementation deployed in a sandbox environment. The predicate recommendation modal mediates the Knowledge Graph disambiguation results to the user of the Systematic Review Service.

5.3.1 Internationalisation

Internationalisation is the practice of adjusting a UI to different international target populations. Most of all, this regards language translations. SR Services could endeavour to provide their UI in multiple languages. Although ORKG provides

a single English language UI, the preference of external service providers should thoroughly be met. For that reason, the widget includes a build routine that renders different language widget assets from JSON translation files. Unless for a explicit preference, web services can serve languages based on the Accept-Language request header. However, the widget is served from the ORKG host system. The SR Services mediate the user session. User request headers are thus hidden from the OSyRIS server application. Therefore, the OSyRIS client script can be queried with the URL parameter `/?lang` with an alpha-2 *ISO* language code [104]. For undefined languages, the English widget is served as a default.

5.3.2 Events

Web browser JS environments build on event-driven communication. This is, nodes in the DOM can cause emission of events that possibly carry information. Events bubble upwards the DOMs-tree as to allow hierarchically related elements to listen to the events with a callback. A typical type of event, for instance, is the *click* event (inherent to a directed mouse click by the user). The widget is attached meaningful events for the integrator to listen for. Although the widget is a blackbox application, an integrating SR Service can optionally handle the events in a desired way. For instance, the export button could be hidden after the SR was successfully exported. Once the review data has changed, the button visibility could toggle once again. Table 5.1 states two events that are part of the implementation. Listing 5.3.2 shows an example for how an event can be handled.

Event Name	Event Detail	Event Cause
`error`	Error message	The SR export process aborted with error.
`export`	URL of *Comparison* created in ORKG	The SR export process was successful.

Table 5.1: Events associated with the OSyRIS widget underlying custom web component. Each event is emitted upon a related cause. Events can optionally carry cause-specific detail information. The detail information can be read by attached listeners.

Listing 5.3: Example listener on export widget element. The listener intercepts errors throughout the export process. The handler assigns an implied data synthesis table element an error class in order to give visual feedback (e.g. as red border around the table).

```
document.querySelector("orkg-export")
.addEventListener("error", e => {
    document.querySelector("table.synthesis-table")
    .classList.add("error");
});
```

5.3.3 Errors

Whenever an error occurs throughout the export process, the widget transitions to the error modal. Known error sources are deliberately caught in order to express an according error message. Unknown errors are intercepted globally. This results in a generic error message. In case the error originated from the communication of OSyRIS (including to the ORKG backend), the generic error message is appended with the respective server error response message.

5.3.4 Styles

To blend into the ORKG fronted, the widget applies a coherent stylesheet. To date, ORKG has not published design guidelines. However, as far as possible, the widget reuses layout and styles from the ORKG frontend application. Mappable layouts were filtered from the actual ORKG frontend markup. Mappable styles were filtered from actual ORKG frontend stylesheets [3].

5.3.5 Mapping Task

The specified *Candidate Selection Component* presents the user with predicate candidates. The highlighted options are thereby the individual predicate labels. As usual for a checkbox, the different options are aligned in a horizontal bid.

The *Candidate Score Component* is a meter. Meters are usual for representing an analogous state degree. The degree here is evidently the mapping score. Research on UI elements supported that redundant depictions of a single state representation element reduce the perceived cognitive load [105]. For this, the score meter is applied also color coding. The coloring reflects the score on a slightly desaturated spectrum

[3] A new stylesheet was compiled to bear overall reproducibility in mind. However, a more *DevOps*-friendly integration with the ORKG Frontend would be achieved with merging the new stylesheet with the existing ones. The existing stylesheet(s) could then simply be referenced from the widget markup in-place of the new stylesheet.

between red (poor score) and green (strong score). The RGB[4] value is computed with the following function:

$$Color : [0,1] \subset \mathbb{R} \mapsto \{\{c_0, c_1, c_2)\} \mid i \in \mathbb{N} : c_i \in [0,1],\ 1 \leq i \leq 3\} =: \mathbb{RGB}$$
$$Color(score) := \{\tfrac{1-score}{2} + \tfrac{1}{3},\ \tfrac{score}{2} + \tfrac{1}{3},\ \tfrac{score}{3}\}$$

Additionally, a third degree of cognitive redundancy is introduced. The score meter show a gradual transition between values (bar width, text, color) upon changing the selected option. Animated transitions were in fact supported helpful with value-differential decision tasks [106]. The animation approach uses a recursive function with an ease out, logarithmic call deference.

$$Animate : [0,100]^2 \subset \mathbb{N} \mapsto \{(c,s) | c \in \mathbb{RGB}, s \subset [0,100]\} \quad \text{with} \quad C_{delay} \in \mathbb{N}$$

Listing 5.4: ...for small constants (< 1000), the full recursion timeout (sum of iterated timeouts) is roughly 10 times the constant in milliseconds.

1 **PROCEDURE** $Animate(score_{cur},\ score_{target})$
2 **IF** $score_{cur} = score_{target}$ **THEN RETURN**
3 $score_{cur} \longleftarrow score_{cur} + \frac{score_{target} - score_{cur}}{|score_{target} - score_{cur}|}$
4 $color_{cur} \longleftarrow Color(score_{cur} * \frac{1}{100})$
5 $timeout \longleftarrow \max\{\log^2 |score_{target} - score_{cur}|, 1\}^{-1} * C_{delay}$
6 **DEFER** $Animate(score_{cur},\ score_{target})$ **FOR** $timeout$ **ms**
7 **RETURN** $(color_{cur},\ score_{cur})$

[4]The acronym *RGB* stands for the *Red*, *Green* and *Blue* color space model. *RGB* colors are triples of respective color channel intensities. A described color is additive with its components, i.e. the higher the cumulative value is, the brighter the color. Components are from a system specific unit interval.

Chapter 6

Evaluation

So far, OSyRIS has proposed approaches that supposedly enable an integration of SR Services with ORKG. Yet, whether the implementation actually solves the problem is to be evaluated. The service is therefore exemplary integrated into a real SR Service client. This is achieved with on a formally representative injection. Also, the quality of the proposed disambiguation algorithm *ScopedDBSCANScoring* is empirically evaluated on a problem-specific dataset. Furthermore a qualitative evaluation highlights the most suited UI alternatives for the predicate mapping task.

6.1 Service Interface

Integrating OSyRIS requires but a minimum of two lines of code (statements, to be precise). This comprises the client application script and the widget tag. To serve the client model interface, a data read callback is likely required, nonetheless. Functionality of the service can be proved without requiring a SR Service to actually integrate it productively. Neither does the ORKG ecosystem. In fact, the service can be deployed and made accessible from an experimental host. Technically, this is equivalent to a productive deployment from the ORKG end. Productively, SR Services are required to integrate OSyRIS though the script reference and successive widget setup (see 5.1.5). What is interesting here is that the integration does not necessarily require a static integration, i.e. encoding into the web asset files. A dynamic integration on SR Service client runtime works quite as well. That being said, an arbitrary SR Service client can be injected the integration script (for debugging) and successive setup code on runtime. Technically, this is in turn equivalent to a productive integration from an SR Service end. The outlined procedure is performed in **CADIMA** [9]. The data read callback (`orkg-data-callback`) reads

data from a data extraction table presented within the data extraction document (`/area/dataExtractionView`). Listing 6.1 contains the code that is injected into **CADIMA** to achieve a virtual productive deployment.

Listing 6.1: Code snippet that can be injected into the *CADIMA* data extraction document upon runtime in order to test OSyRIS with with *CADIMA*. Furthermore, the code snippet can simply be adjusted for another Systematic Review Service (within the `orkgReadCallback` callback).

```
const script = document.createElement("script");
script.setAttribute("src", "http://<EXPERIMENTAL-HOST>/widget.js");
document.querySelector("head").appendChild(script);

window.orkgReadCallback = () => {
    const criteria = Array.from(
        document.querySelector("#dataExtractionGridView").querySelectorAll("th")
    ).map(th => th.textContent.trim());

    const csvData = [ [ "author_names", "paper_name" ]
        .concat(criteria).join(",") ];

    const paper = [];
    Array.from(document.querySelector("#dataExtractionGridView")
    .querySelectorAll("td")).forEach((td, i) => {
        const j = i % criteria.length;
        if(i !== 0 && j == 0) {
            csvData.push([ "Example Author", `Example Paper ${j}` ]
                .concat(paper).join(","));
            paper.length = 0;
        }

        paper.push(td.textContent.trim());
    });

    return ORKGExportElement.parseCSVString(csvData.join("\n"));
};
```

```
28
29 const widget = document.createElement("orkg-export");
30 widget.setAttribute("orkg-data-callback", orkgReadCallback);
31 document.querySelector("body").appendChild(widget);
```

OSyRIS can be shown functional completeness through the described injection. The widget is presented as expected. Using it to export an example SR to the ORKG sandbox also works as expected. The dominant non-functional requirements can be accepted to be fulfilled. McConnell states that per day professional programmers write about 70 Lines Of Code (LOC) for smaller projects (ca. 10,000 (LOC)), and about 10 LOC for large projects (ca. 10,000,000 LOC). Jones, Bonsignour, and Subramanyam estimates about 25 LOC across different domains. To be compared, the injection required 25 LOC (without whitespace), i.e. at most one wo:man day of integration effort.

6.2 Knowledge Graph Disambiguation

The disambiguation of entities was described as a formal procedure. The DOI can distinguish papers globally, as it is a unique identifier by design. Whether the predicate-based disambiguation requires an evaluation of the *ScopedDBSCANScoring* algorithm.

6.2.1 Data Preparation

Evaluation of *ScopedDBSCANScoring* requires a dataset upon which the algorithm can be executed and results reasonably be interpreted. Related work on KG alignment is commonly evaluated on corpora like *CESSMcite* or *DBP15Kcite*. Besides, a gold standard dataset for contextualised KG disambiguation as presented in this work does not exist. That being said, an according dataset must be constructed beforehand [11]. The dataset should contain a comprehensive set of real world representative weak truth instances, i.e. predicates and associated literals. Also, these weak truth instances should reflect the data distribution dynamics typical for scholarly KGs. Having said this, the construction comprises a query step to retrieve real data from a scholarly KG, as well as a step that adds different variations of weak truth instances.

Dataset Query The dataset is compiled directly from ORKG. For the evaluation, two (most) usual types are focused. This is strings and decimals. Decimals are

structurally representative for numbers in general. Per datatype, data was queried for 10 high- and low-volume literal contexts each through the SPARQL-endpoint. The low-volume literal contexts should represent situations where a disambiguation algorithm could not rely on a comprehensive weak truth context. Notice that only predicates with a description were extracted in order to allow for interpreting the data later on. As a result, a fundamental dataset with a total of 40 records was constructed. Listing **??** states the SPARQL query used to retrieve the different volumes of datatype specific evaluation dataset instances from ORKG.

Listing 6.2: SPARQL query for retrieving parts of the basic evaluation dataset. The tokens shaped <A|B> (lines 11 and 18) simply denote different alternatives `A` and `B` that were used with different queries. All four combinations were used.

```
1  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3  PREFIX orkgc: <http://orkg.org/orkg/class/>
4  PREFIX orkgp: <http://orkg.org/orkg/predicate/>
5
6  SELECT DISTINCT ?predicate ?label ?description
7    (GROUP_CONCAT(?literal; separator="; ") AS ?literalsList)
8  WHERE {
9    ?contribution a orkgc:Contribution ;
10                 ?predicate ?object .
11   FILTER(datatype(?object) = xsd:<string|decimal>)
12   ?predicate rdfs:label ?label ;
13              orkgp:description ?description .
14   BIND(str(?object) AS ?literal)
15 }
16 GROUP BY ?predicate ?label ?description
17 HAVING(COUNT(?literal) >= 1)
18 ORDER BY <ASC|DESC>(COUNT(?literal))
19 LIMIT 10
```

Dataset Annotation In order to get an idea about how data in ORKG is constituted, the dataset was accordingly annotated. The annotators were two graduate computer science students. The annotations happened independent from each other. The annotation process comprised a few steps: At first, the annotators were given

only the predicate descriptions. With that, they were asked to annotate the criteria that are supposedly referred to by the descriptions. Afterwards, the annotation results were exchanged in order to classify the criteria label chosen by the other annotator. This step should reveal information on how concepts in ORKG would meet with criteria labels used in SRs. The classes were as follows.

$\mathbf{C}_{phrasing}\mathbf{1}$ *The annotated label exactly matches the actual dataset label, i.e. is a perfect phrasing.*

$\mathbf{C}_{phrasing}\mathbf{2}$ *The annotated label partially matches the actual dataset label; i.e. is partial phrasing. At least one of the space separated tokens in the annotated label is a proper subtoken of the dataset label.*

$\mathbf{C}_{phrasing}\mathbf{3}$ *The annotated label does not at all match the actual dataset label, i.e. no one of the space separated words in the term is a substring of the label*

The distribution across classes reflects how data in a is constituted. Partial phrasings are most frequent. This is probably due to the fact that criteria share a common concept name, but are expressed differently (order, supplementary words, etc.). At around half of the phrasings, the dataset label is partially matched. Phrasings that do not at all relate with the dataset label occur about 10% more frequently than perfect phrasings. Figure 6.1 shows the phrasing class distribution as a bar plot. For the definite analysis, the results should be weighted according to the phrasing distribution to better reflect the real world model.

Second on, the annotators were given the entire preliminary dataset. They were now to annotate the representation of the literal data. The classes are thus regarding whether literals are represented as clusters of coherent representations, clusters of coherent representations with outliers (noise), or as arbitrarily heterogeneous representations. The term representation coherency was explained to the annotators as whether literals are perceived to belong to a common group, e.g. numbers from a certain interval or strings that are all cities.

$\mathbf{C}_{cluster}\mathbf{1}$ *Clusters of representations, i.e. homogeneous structures of literals (e.g. only [0,1], or [0,1] and [0,100] for the concept Recall).*

$\mathbf{C}_{cluster}\mathbf{2}$ *Clusters of representations, but including outliers.*

$\mathbf{C}_{cluster}\mathbf{3}$ *No clusters of representations, but only heterogeneous structures of literals.*

The class distribution differs significantly across the datatypes. Clusters of literal representations are recognised at any decimal strong truth. Around 80% of the decimal classifications do not even reveal outliers. On the contrary, strings show only a few clean clusters. It comes with a similar fraction of clustered representations with outliers and heterogeneous representations at around 40% on the dataset, each. Figure 6.2 shows the representation class distribution as a bar plot. For the definite analysis, the results should (also) be weighted according to the phrasing distribution. This should better reflect the real world data dynamics.

It seems that numerical data is framed more frequently by common units and typical value ranges. String data, on the other hand, seems to be used less within generic frames. Particularly, unit measures regularly apply to numeric data only. Obviously, the expected classification is a subjective matter. The classification of decimals, however, achieved perfect agreement. A κ-measure (*Cohen*) of 0.66 for the string classification provided a moderate inter-rater reliability. For constructing the dataset, disagreements were resolved in a joint discussion.

Dataset Refinement The final dataset was ought to compile a reference between possible strong truth labels and different representations of the weak truth. The weak truth is given by the queried data (a scoped snapshot of a KG). The to-be-evaluated strong truth is additionally raised with a perfect label, a partial label, and a not matching label. The different labels can be constructed right from the weak truth in the dataset, respectively. In fact, the perfect phrasing equals just the weak truth label. The partial label is just the longest space-separated word in the weak truth label. In case no space exists in the weak truth label, the first three quarters of the characters in the label are used as the partial label (outer whitespace is stripped). The not matching label is just a randomly generated word with an equal length as the weak truth label. However, latters in the generated word do not contain any character that is also contained in the weak truth label. This should obviate accidental string overlap score advantages. Furthermore, for the high-volume fraction of the data, the weak truth literals are given in three constitutions (8 items each): As a perfect subset of a single representation cluster in the weak truth (intersection). As a set of literals that are each not contained in the weak truth, but represent roughly valid values for the respective concept (disjunction). This is useful to check if a contextual predicate disambiguation algorithm also works on data that is not in the weak truth. And, as a half overlap of a subset of a single representation cluster and not contained literals. This is, it can be drawn right from the intersection and the disjunction. Constructing an intersection can also be performed with little effort, eyesight and optional review capabilities. Constructing a disjunction, however, is not

a trivial task (especially for the string data). For that reason, a pre-trained general purpose language model greatly lends itself to solve this task. It helps with providing less biased, more far-sighted generation of contextual text (compared to humans). *GPT-3* models have recently supported great quality in recalling and paraphrasing literature publications. Particularly, they have been trained on a plethora of scientific literature [109]. The model is prompted with the following template, given a comma-separated list of the weak truth labels in place of the pattern expression (up to the token limit):

Please generate exactly 8 additional concepts related to the scientific concepts described by the following values: "`<labels>`*". Do not reuse any of the given concepts.*

To wrap it up, the refined dataset enables evaluations of predicate disambiguation algorithms. Therefore, the algorithms can be iterated over the real world representative weak truth. Therefore the full cartesian product of three strong truth label phrasing types (perfect, partial, none) is given to the evaluated algorithm. If the respective weak truth has a high volume of literals, different strong truth literals constitutions (instersection, disjunction or their overlap) can also be tested.

6.2.2 Experiment Subjects

OSyRIS is a deliberate work towards progressing ORKG. At that, the baseline system for the *ScopedDBSCANScoring* algorithm proposed in this work should be the ORKG system closest with solving the predicate disambiguation problem. In fact, the *ORKG SimComp API* provides a similarity interface for comparing scientific artifacts. However, it is build for comparing papers and contributions, rather than (comparably small) predicate-literal contexts. For predicates, ORKG implements a label only-based predicate mapping recommendation interface. It is implemented with the manual contribution creation form in the ORKG frontend application. The service bases on *Elasticsearch*. The evaluation herein uses the outlined system as a baseline. For simplicity, it is referred to as *ORKG SimPredicate*. Furthermore, different parametric variations of the *ScopedDBSCANScoring* algorithms are evaluated against the baseline:

ORKG SimPredicate (experimental name) *The ORKG frontend application provides a predicate label-based recommendation retrieval interface. The retrieval returns a ranked list of mapping candidates.*

ScopedDBSCANScoring$_{0.50}$ + Jaccard *ScopedDBSCANScoring with a literal only-based scoring weight of 0.5 and unit normalised Jaccard for the literal only-based*

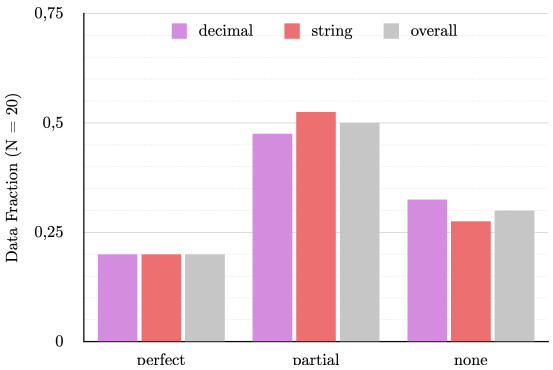


Figure 6.1: Data fraction per class of criteria phrasings that were provided by independent annotators for concepts described by only a short explanatory text from ORKG. Based on the true label, the phrasing classes differentiate perfect matches, partial matches and no matches (i.e. empty intersection of space separated words). The data comprises 20 instances per datatype (decimals and strings). The average over both classes is presented with the gray bar plot. Partial matches dominate the distribution. No match seems more likely than a perfect match, which foreshadows problems for label only-based disambiguation measures.

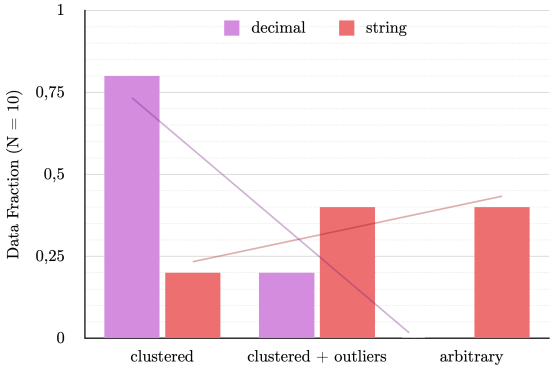


Figure 6.2: Data fraction per class of representation clusters that were provided by independent annotators for predicate related literals from ORKG. Literal representations can be classified either as given only in clusters of coherent representations, clusters of coherent representations with outliers, or as an arbitrary heterogeneous set. The data comprises 10 instances per datatype (decimals and strings; only high-volume literal contexts allow for an according classification). The trendlines show a coarse divergence between the datatype classification dynamics.

scoring

ScopedDBSCANScoring$_{0.50}$ + Levenshtein *ScopedDBSCANScoring with a literal only-based scoring weight of 0.5 and frequency-weighted Levenshtein for the literal only-based scoring*

ScopedDBSCANScoring$_{0.75}$ + Levenshtein *ScopedDBSCANScoring with a literal only-based scoring weight of 0.75 and frequency-weighted Levenshtein for the literal only-based scoring*

6.2.3 Experiment Setup

The baseline system, as well as the algorithms, are each applied to the previously raised gold standard dataset instances. The ORKG baseline does not provide a transparent scoring, but yields relevance implicit with the ordering. Moreover, the predicate disambiguation does not ask for an appropriate ranking, but for a specific predicate. For this, the subjects are not compared based on typical information retrieval quality metrics like the F1 score. Instead, the experiment subjects are compared on a problem specific one-precise precision@k metric (*1-precision@k*). This is, the *1-precision@k* corresponds to one divided by the position of the correct candidate in the results list (e.g. 1/4 for the correct candidate at fourth position). To enable repeatability, the experiment is run on the gold standard dataset instead of an ad-hoc created weak truth index. Individual scores of *ScopedDBSCANScoring* algorithm are monitored in parallel to allow for an isolated quality statement.

6.2.4 Experiment Analysis

The data phrasing distribution foreshadows that label only-based scoring is weak on the dataset. The fraction of perfect phrasings are lower than the other two types of phrasings. Evidently, **ORKG SimPredicate** works perfectly on the perfect phrasing labels. However, it does also not work at all on the none matching labels. The proposed *ScopedDBSCANScoring* algorithm mitigates the label only context insensitivity. The best performing subject is **ScopedDBSCANScoring$_{0.5}$ + Levenshtein**. In total, the subject algorithm outperforms the baseline by a margin: The overall *1-precision@k* average presents an advantage of about 17% over the **ORKG SimPredicate**. In order to obtain a more representative score for ORKG, the *1-precision@k* values are additionally balanced according to the previously obtained phrasing class distribution. The overall advantage furthermore grows to even about 60%. Against prior expectations, the letter-wise *Levenshtein* similarity performs better than the word-wise *Jaccard* string similarity metric within the scoring algorithm. Equal weight between label and literal scoring, however, seems traceable with the two-context approach. Table 6.1 compiles the experiment results for all subjects upon both the balanced and the phrasing classification-based weighting. Figure 6.3 shows the same data as a bar plot, including an overall phrasing dynamic balanced average.

	Overall *1-precision@k*
ORKG SimPredicate	0.430, 0.337 (balanced)
ScopedDBSCANScoring$_{0.50}$ + Jaccard	0.478, 0.485 (balanced)
ScopedDBSCANScoring$_{0.50}$ + Levenshtein	**0.504, 0.540** (balanced)
ScopedDBSCANScoring$_{0.75}$ + Levenshtein	0.501, 0.539 (balanced)

Table 6.1: Experimental results after execution on each stated subject on the gold standard dataset. The results are drawn from the balanced dataset, as well as weighted according to the strong truth label phrasing behaviour revealed earlier.

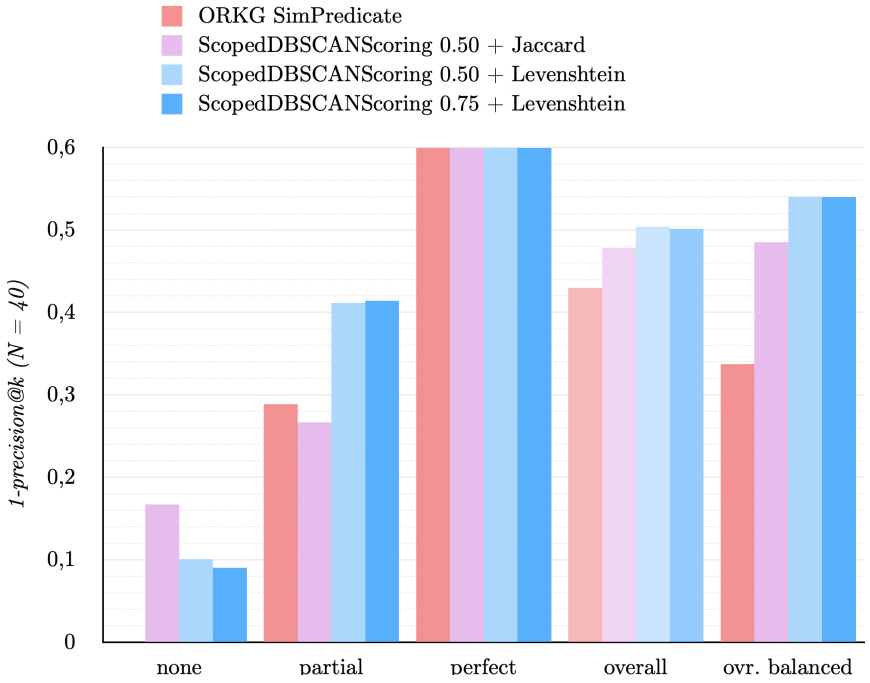


Figure 6.3: Experiment specific *1-precision@k* measures (vertical axis) of the different subjects (horizontal axis) after execution on each stated subject on the gold standard dataset. The results are shown for each label phrasing class in isolation, as well as the bare and class distribution balanced (weighted average) overall. The perfect phrasing results in a perfect score for all subject and is thus cut in favour of zooming into the differences. Notably, all parametric variations of the *ScopedDBSCANScoring* algorithm outperform the *ORKG SimPredicate* titled baseline. Especially after balancing the results according to the criteria label phrasing distribution revealed beforehand, the performance advantage is about 60%.

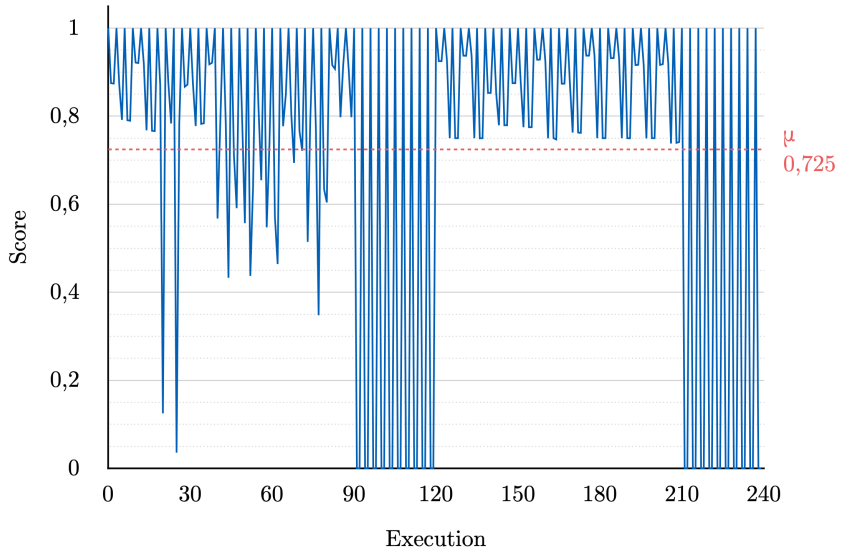


Figure 6.4: Plot of each mapping score computed over all of the 240 executions of the *ScoepdDBSCANScoring* algorithm with *Levenshtein* string similarity and a label based-only score weight of 0.5. The two spots in the data with visual score drops occur at data that neither provides a useful label, nor a literal data context. The substantial score average is a promising insight to accept supportive quality for humans performing the mapping task.

Besides just ranking precision, the average correct mapping score over all 240 algorithm executions shows substantial quality. This is, the algorithm provides an expressive value for humans using the score to decide upon suitable mappings. 6.4 is a plot over all of the 240 scores. Two areas, however, reveal poor scores. These occur for predicates that do not have a sized weak truth literals context, as well as strong semantic label differences. Given that no context at all is given at that point, the scoring can be assumed better for any halfway representative context given.

6.3 User Interface

Usability experiments are concerned with evaluating different atomic purpose user interface elements upon overall acceptance. The two described mapping task specific elements are thus investigated. Different alternatives were given to a group of users in order to have them interact with each alternative in mutual isolation. Subsequently, each participant was asked to assess each alternative. The assessment bases on

different usability perception aspects defined upfront as a uniform survey.

6.3.1 Experiment Conditions

Evidently, the experiment comprises two conditions, namely the proposed elements. The levels for each such independent variable are the three different alternative specifications.

Condition 1: Predicate Selection Component

Alternative A First result

Alternative B First 3 results

Condition 2: Predicate Score Component

Alternative A Text (only) percentage

Alternative B Bar meter percentage

Alternative C Text and bar meter percentage

6.3.2 Experiment Task

Each participant is instructed to export the results of a minimal SR synthesis to a fictional . In order to sketch a roughly familiar review situation, the participants are asked to extract relevant contributions from a set of three fictional paper abstracts, beforehand. Therefore, the contributions including to respective criteria labels must be written to a provided synthesis table. Regardless of the extracted data, the export randomly generates scores and candidate labels (different case or generic prefixes). Half of the generated candidate scores are (at least) substantial, whereas the other half is poor. This should nudge the participants towards interacting with the widget.

The abstracts for the tasks are generated using again the *GPT-3*-API. The prompts are based on a generic prompt template. Table 6.2 states the tokens substituted in the following template to generate 3 abstracts for 5 topics:

Please generate 3, about 50 words long fictional abstracts about the topic `<t>`*. Within each abstract, state specific values for the properties* `<p1>`*,* `p2>`*, and* `<p3>` *with specific labels.*

Topic `t`	Property `p1`	Property `p2`	Property `p3`
information retrieval systems	*proposed system name*	*evaluated precision*	*evaluated recall*
language models	*proposed model name*	*evaluated vocabulary size*	*inter-annotator agreement*
code complexity metrics	*proposed metric name*	*time complexity*	*degree of approximation*
graph walk algorithms	*proposed algorithm name*	*time complexity*	*whether it is complete or not*
graphic processing units	*proposed device name*	*number of cores*	*average task time*

Table 6.2: Token substitute sentences for different topics from the computer science domain. The fundamental prompt towards *GPT-3* asks for the generation of 3 fictional abstracts for the respectively substituted topic.

6.3.3 Experiment Setup

The usability experiment is performed with an unsupervised, within group design. Test subjects are able to load the experimental application remote and perform it in their own pace. Since the widget is set to be used from within an arbitrary SR Service, a mocked-up SR Service application is provided. The application only provides a data synthesis view. Also, a generic target scholarly KG is mocked-up. The such set up environment is minimal in respect to perform the task. Particularly, however, this setup eliminates side effects which could occur with real world applications, including precautionary bias and distracting features. Figure 6.5 shows a screenshot of the mocked-up SR Service. Accordingly, figure 6.6 shows a screenshot of the mocked-up scholarly KG.

Each participant is assigned a single experimental condition, for which all levels. This is, the UI element alternatives are sequentially presented upon a new task instance. The number of participants depends on the condition level cardinality (2 and 3, respectively). **Condition 1** is tested on 4 participants[1]. The levels are assigned according to two full rotations of *Latin Square* ordering to balance out learning effects. **Condition 2** is tested on 6 participants[1], given one rotation of the *Latin Square* ordering strategy.

[1] Usually, the amount of test subjects used for qualitative studies is extended until saturation in the analysed data is reached. However, an adequate minimum of subjects is applied. This is usually not less than 15 to 20 people. Due to the strict time frame of this work, the amount of participants was considered sufficient, as data saturation was reached early.

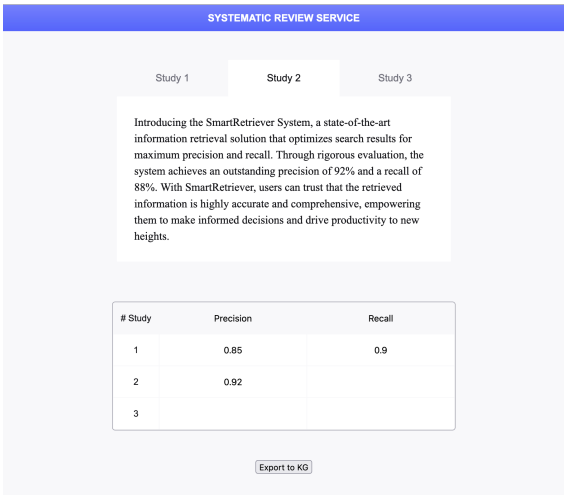


Figure 6.5: Screenshot of the mocked-up fictional Systematic Review Service application deployed for the usability experiments.

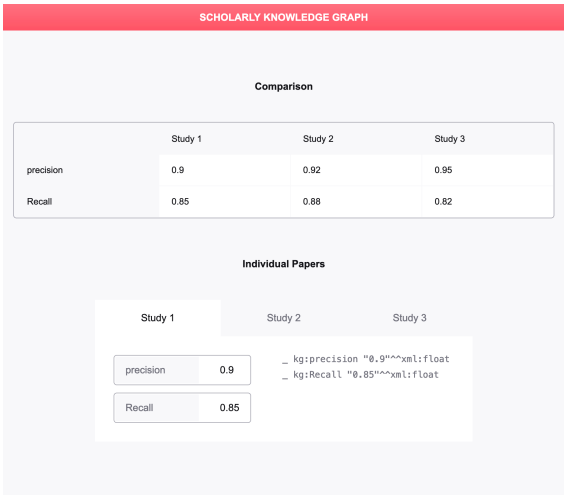


Figure 6.6: Screenshot of the mocked-up fictional scholarly Knowledge Graph application deployed for the usability experiments.

After all conditions were tested by each participants, they are asked to complete a short survey. The survey repeats over all alternatives in the assigned order. The answers are asked through a 5-point *Likert* scale, spanning across the options *"Strongly agree"* (5), *"Neutral"* (3) and *"Strongly disagree"* (1). The survey comprised below stated questions:

Q1 *The interface element was required to complete the integration task.*
The respective element provides requisite functionality for solving the mapping task. This statement should reveal how useful the element was actually perceived. This is either that a mapping should be selected (**Condition 1**), or that a suitability estimate (score) should be known (**Condition 2**).

Q2 *The interface element's underlying state was precisely depicted.*
The respective element carries variable information, i.e. displays a complex state. This statement should reveal how readable the depicted element state was perceived. This is either mapping options (**Condition 1**), or a mapping score (**Condition 2**).

Q3 *The interface element's purpose was clear to understand.*
The respective element subliminally communicates how it should be used by

their appearance. This statement should reveal how affordable the element was perceived. This is either actively selecting a mapping (**Condition 1**), or reading a score (**Condition 2**).

Q4 *The interface element was expected to exist as presented.*
The respective element requires a certain cognitive load from the user. This statement should reveal how familiar the element was perceived. This is either a checkbox nature (**Condition 1**), or a meter nature (**Condition 2**).

Q5 *The interface element's appearance fit into the overall widget.*
The respective element is part of a composed interface for solving the mapping task. This statement should reveal how aesthetic the element was perceived.

Q6 *Usage of the interface element did not require additional information.*
The respective element is one of a many factors in solving the mapping task. This statement should reveal how sufficient information for using the elements was perceived.

6.3.4 Experiment Analysis

The overall experiment results support **Alternative B** more usable for **Condition 1**, and **Alternative C** for **Condition 2**. Table 6.3 lists different *Likert* scores for different score perspective bins. Figure 6.7 shows a bar plot of the overall *Likert* scores.

	Bin	Overall	Q2..Q5
Condition 1	Alternative A	3.46	
n = 4	Alternative B	**4.29**	
Condition 2 (I)	Alternative A	3.56	4.04
n = 6	Alternative B	3.11	3.71
	Alternative C	**3.61**	**4.17**
	Question 1	**2.75**	
	Question 2	4.63	
	Question 3	4.81	
	Question 4	3.87	
	Question 5	4.56	
	Question 6	**2.50**	
Condition 2 (II)	Alternative A	3.92	
n = 4	Alternative C	**4.29**	

Table 6.3: Computed *Likert* scores for the different conditions, alternatives and partially also questions. Alternative A for condition 1 and alternative C for condition 2 reveal most usable. Q1 and Q6 show a poor score compared to the remaining scores for the second condition. After implementing preliminary insights, additional experiments show improved *Likert* scores.

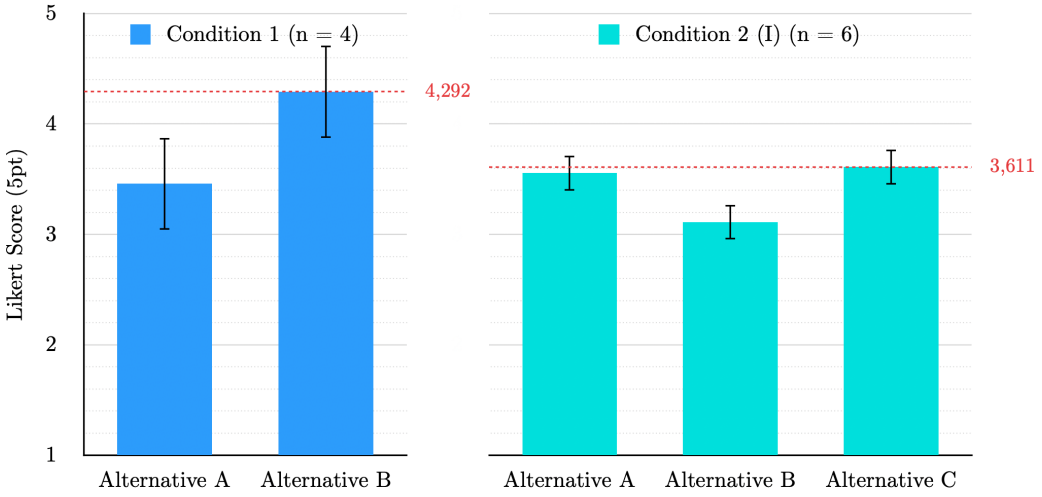


Figure 6.7: Survey results given the 5-point *Likert* score (vertical axis) over all condition alternatives (horizontal axis). The error bars depict the standard error. n refers to the number of participants per condition.

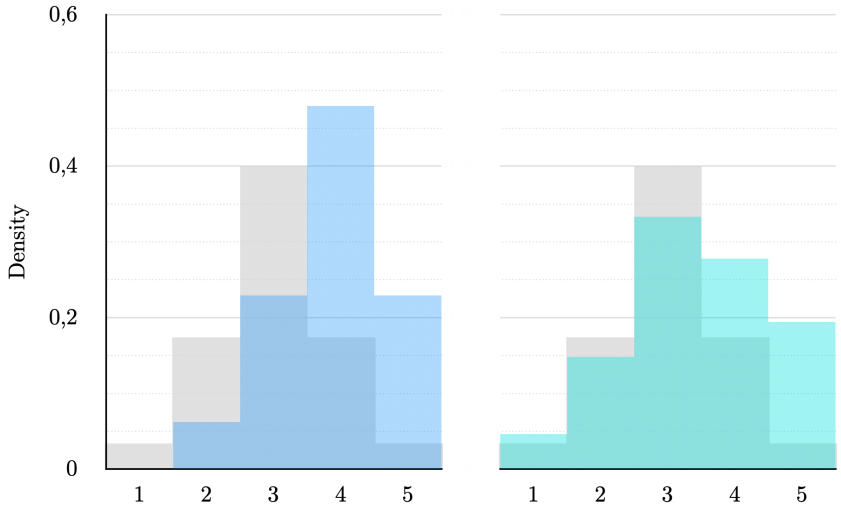


Figure 6.8: Distribution of the *Likert* scores from the usability survey. The gray distribution in the background depicts the expected normal distribution around the neutral score. Notice that a normal distribution is roughly matched, but shifted for the first condition (blue).

Figure 6.8 shows that the *Likert* score data is slightly skewed from the mean, i.e. rather not normally distributed. To support statistical significance, a non-parametrical test was required. Given are two paired samples of ordinal outcomes (*Likert* scores) based on nominal exposure (conditions). Among referable non-parametric statistical tests, the *Wilcoxon Signed-rank* test is suited for the present properties. The test investigates the central tendencies between two samples. To support statistical significance for the given survey results, the null hypothesis is tested (two alternatives for a condition have the same usability effect). As non-parametrical tests are considered less robust than parametrical test, the chosen significance level was 0.01. At a p-value below the significance level, the null hypothesis would be rejected. Then, the two-tailed alternative hypothesis would be accepted (the higher *Likert* score also implies better usability).

Wilcoxon Signed-rank results provide that differences in the **Condition 1** outcome are significant at a p-value of 0.00044 (z-value -3.5162). Hence, **Alternative B** is the more usable one. It can be assumed, that the options element is better to understand with a few options rather than just one supplementary option. In contrast, *Wilcoxon Signed-rank* results provide that differences in the **Condition 2** outcome are not significant at a p-value of 0.68916 (z-value -0.4024).

For the second conditioned element, participant answers show a margin between

the score trend of **Q1** and **Q6**, and the remaining ones. Figure 6.9 shows the *Likert* scores for **Condition 2** for each question. Looking at **Q1** in isolation, the elements appear superfluous. However, the score is technically mandatory information to solve the task. Therefore, the scores rather show that additional information is required to correctly interpret the score element. Therefore, the implementation is added candidate predicate descriptions from ORKG and a more detailed mapping task description. Four additional experiment participants are given the adjusted score meter condition. The second survey round reveals that the scores significantly improve compared to those from the first round. More precisely, the details improved the element purpose perception. Now, **Alternative C** highlights as the best usable alternative with a substantial score. Figure 6.10 shows the *Likert* scores for **Condition 2** after each survey round.

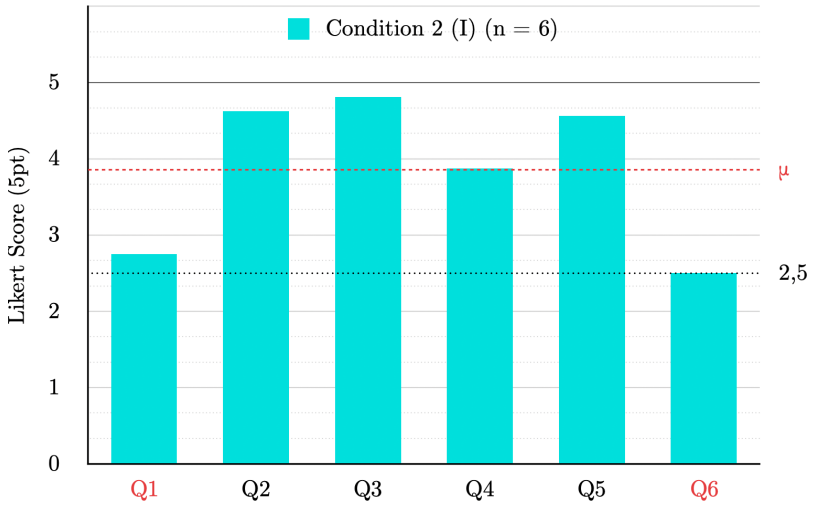


Figure 6.9: Survey results given the 5-point *Likert* score (vertical axis) over the individual questions (horizontal axis) for the second condition. Answered score averages for question 1 and question 6 compare poor to the remaining answers. With a certain technical purpose, the answers indicate a lack of detail information.

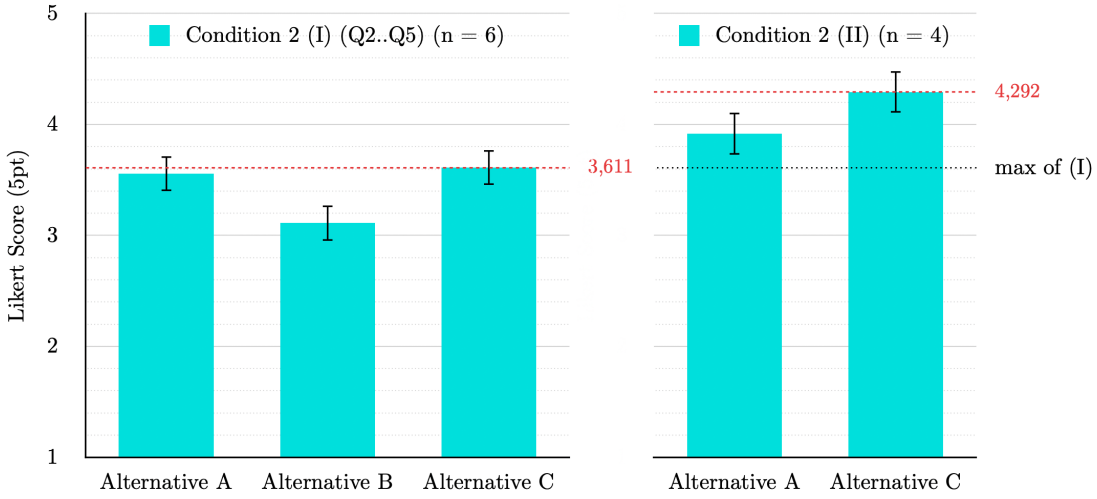


Figure 6.10: Survey results given the 5-point *Likert* score (vertical axis) over focused condition alternatives (horizontal axis) for the second condition. The first diagram (I) depicts the scores after the first survey round. The second diagram (II) depicts the scores after the second survey round. Notably, the added detail information advanced usability of the text-based score meter alternatives from a moderate to a substantial level.

Chapter 7

Discussion

OSyRIS comprises the technical depth modern web services stipulate. Many aspects to the service, however, resemble best practices and remain unmentioned to self-evidence. Yet, the significant and novel approaches were explained in detail. Also, the respective implementations were succinctly described, as well as empirically evaluated. On a closing note, the presented solution is critically discussed. At that, the guiding research questions are revisited. Furthermore, known limitations with the work are stated and commented. Not least, ideas for future work are presented.

7.1 Solution

Widget Service The widget approach was shown to serve the infrastructural requirements laid out early. In fact, according interoperability within the technical bounds was already supported from the background. The lean and generic interface towards arbitrary SR Services favours low effort integration. It also sustains software engineering quality, hiding the entire logic in the scholarly KG backend application. The ultimate window scope injection into *CADIMA* acted as a proof of functionality. At both to show ease of integration on the SR Service end, and data export capabilities. Without further ado, **RQ 1** was satisfied to great extent.

Knowledge Graph Disambiguation The approached online disambiguation measures upon the scholarly KG were also thoroughly supported. As a collateral contribution, a gold standard dataset was first compiled to allow for an unbiased evaluation of predicate mapping (and possibly scoring) algorithms/systems. The proposed *ScopedDBSCANScoring* algorithm was supported to outperform the ORKG baseline

by a margin (about 60% performance advantage). Given that *ScopedDBSCANScoring* was specifically designed for contextual SR data, it holds notable advantages over the label only-based solutions. In particular, it connects at the strength of the label only-based scoring, but can supplementary rely on a literal data context to recover on poor labels. However, it can still not help with mappings for weakly contextualised data on the full spectrum, which is reasonable. Since the implementation was given as a first attempt using only a one dimensional space, further optimisations can be assumed to advance performance once again. Overall, **RQ 2** was quantitatively answered to a substantial degree.

User Interface Not least, the information retrieval specific UI elements were compared for most effective alternative representations. In line with interactive experiments, surveys delivered data to justify an implementation decision. Three mapping candidates for the selection component supported as the best alternative. Statistic significance over the alternative with one candidate was provided. For the mapping score meter, results did not deliver statistically significant differences. However, the analogue score meter without an explicit value lagged behind the other alternatives. A question-isolated view of the results revealed a yet misunderstood imperative for the element to solve the task sensibly. For this, a successive experiment iteration was providing adjusted alternatives with more detail information. The usability improved from a moderate to a substantial level. The choice of the better alternative was now a matter of the best scoring alternative. With standard or template element-based UIs as a baseline, the elaborated UI provided a usability surplus for the mapping task. Thereby, **RQ 3** was also, yet qualitatively answered.

7.2 Scalability

At this point, it is worth assessing the scalability of OSyRIS. At that, scalability is primarily depending on two variables. From the system architecture point of view, increasing request loads can be met out-of-the-box. Additional service instances can be added horizontally on demand. This is due to the the service implementing a stateless paradigm, entirely. The second variable is the size of the KG, which the proposed *ScopedDBScanDScoring* algorithm works on. An upper asymptotic, i.e. worst case time complexity analysis helps to estimate performance on a huge KG. Encompassing, the algorithm iterates the weak truth predicates p. Within, the literals only-based scoring is considerably constant in insignificant label lengths. The literals only-based scoring is a succession of ScopedDBSCAN clustering and cluster

affiliation computation. The original *DBSCAN* algorithm is said to work in $O(n^2)$. However, the scoped modification uses a fringe, complementary to open data points for iteration. Also, it terminates after a single cluster detection. Hence, *ScopedDBSCAN* is linear in the amount of d-respective literals l. The cluster affiliation is respectively linear. Moreover, the definite sorting algorithm is limited in a supposedly small k, i.e. in $O(k)$. Having said this, it holds that $ScopedDBSCANScoring \in O(|p| * |l|)$. Linear time for data-complete algorithms marks the lower bound time complexity.

For a reason, the weak truth index construction is outsourced to a more infrequent routine. Querying a KG is linear in its size n, given that SPARQL query patterns stating only graph patterns and `FILTER` statements [110]. However, for each of the queried predicate-literals record, the literals are embedded into a vector space. This is, the index construction runs in quadratic time. With a realistic assumption that the amount of predicates grows sublinear with the amount of literals, the index construction is more expensive than .

7.3 Limitations

Although each of the research questions was answered to a high degree, the work bares several limitations. Most of them occur with connecting problems from the broader view. The following issues are identified.

Complex Contributions Tabular data structures commonly edited through SR Services imply atomic literal and criteria data encoding. This favours mapping to ORKG. Despite being constraint to using literal encoding techniques, however, SR Service users could still use terms referring to complex real world concepts. Those would actually be modelled by an entity in ORKG. A real case from ORKG models locations as entities, sourced from but literally written down location names in the respective review [111], [112]. A similar phenomenon happens with composed relationships. Videlicet, *del graphs* can model complex relationships through *n-ary* relationships. This exploits entities to represent a relation acting as a junction for nested relationships [113]. Thereof independent, comparisons on *n-ary* relationships is also a broader visualisation problem. Overall, how to map flat literal SR data to accordingly complex models represents credible future work.

Predicate Description Sparsity Unfortunately, a large fraction of the predicates in ORKG does not hold a description. Candidate predicate descriptions, however,

supported highly important for users of SR Services to solve the mapping tasks using OSyRIS. Mappings of non-described predicates can be expected significant quality drops.

Score Meter Color The mapping score meter as implemented in the OSyRIS widget applies colour coding to communicate the underlying state. Precisely, using a spectrum between red and green. As is well known, use of colour in graphical UIs is inaccessible for a color blind population. Red-green colour vision deficiency applies to about 5% of the world population [114]. However, color is just one of three visual redundancy degrees the score meter depicts. For the case, color blindness is thus not an excluding issue.

7.4 Future Work

OSyRIS favours a plethora of adjustments and extensions. A few impulses for few future work are shared below.

***ScopedDBSCANScoring* Optimisation** As laid out above, the complexity of the ScopedDBSCANScoring algorithm is mostly due to the literal embedding, clustering and cluster affiliation. That being said, future work could drive these individual procedures in order to obtain improved results, overall. Also, whether the entirety of the weak truth is required to compute useful mappings, or rather a representative subset, could be assessed.

Full-text Consideration Some of the SR Services offer full-text reviews. This is, users are able to upload papers and work directly on the text, e.g. to highlight relevant information. Extending the OSyRIS integration APIs, the full-texts could also be part of the recommendation request data. On this, a methodology presented by Oelen, Stocker, and Auer for importing comparison tables from full-text review articles could be reused [115]. Just as well, the full-text-based predicate recommendation pipeline presented by Oghli could be applied [47].

ORKG Frontend Integration The OSyRIS widget has so far been viewed integrated into a SR Service. The application itself, however, would also work directly inside the ORKG frontend. The interface for manual comparison creation provides *CSV* import functionality. An uploaded *CSV* file is parsed and new KG entities and relationships are accordingly created. The widget could be deployed intermediately,

providing the online disambiguation capabilities to the import. In fact, OSyRIS is able to handle the ORKG mapping process right away, working on tabular data by design.

Naming Conventions Unlike other KGs, ORKG does not uphold a specific naming convention for entities and relationships. In case, however, naming conventions should be introduced in the future, they could be enforced right through the OSyRIS data integration pipeline.

Export Constraints SR Services provide researchers with a platform to organise their SR data. Not all of the data might end up in a published SR, however. The same desire might hold for comparisons in ORKG. This is, the OSyRIS UI could offer the user an additional option to in- or exclude a criteria from the export.

Chapter 8

Conclusions

Scholarly communication aims at sharing a common state-of-the-art across the global research community. Evidently, KGs promise infrastructural benefits for scholarly communication. In contrast to knowledge inference, however, KGs do not excel at human editing friendliness. Instead, popular methodologies like SRs rely on familiar data encoding procedures. Rather than that, a landscape of designated SR Services has been growing. Data stored at SR Services holds inherent potential for growth of scholarly KGs.

This thesis concerned the problem of integrating SR Services with ORKG. From different perspectives, it presented a comprehensive solution in form of a web service titled OSyRIS. The service was presented through a formal specification and respective implementation. Besides integrating seamlessly with the ORKG ecosystem, OSyRIS was shown to provide a low-effort integration interface to arbitrary SR Service. Moreover, an online predicate disambiguation measure to foster conceptual integrity of ORKG was proposed with the *ScopedDBSCANScoring* algorithm. Given pivotal predicate labels and contextual literals, the algorithm can compute the likelihood that already existing predicates in the graph refer to the same real world concept. Empirically, the implemented algorithm outperformed a label only-based mapping baseline by a margin. The evaluation was performed on a supplementary constructed gold standard dataset. The compared baseline is currently implemented in ORKG (a.o. with manual predicate creation). As a third contribution, a mapping task specific UI was examined. At that, better representations among different UI element alternatives were highlighted through a usability survey.

Noteworthy, the solution showed solid scaling capabilities upon a worst case complexity analysis. Future work could be highlighted many aspects to improve OSyRIS components; primarily these were algorithmic and modelling optimisations.

Bibliography

[1] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann, "World-wide web: The information universe," en, *Internet Res.*, vol. 2, no. 1, pp. 52–58, Jan. 1992.

[2] T. Berners-Lee *et al.*, *Semantic web road map*, 1998.

[3] A. Hogan, E. Blomqvist, M. Cochez, *et al.*, "Knowledge graphs," *ACM Comput. Surv.*, vol. 54, no. 4, Jul. 2021. DOI: 10.1145/3447772.

[4] S. Auer, V. Kovtun, M. Prinz, A. Kasprzik, M. Stocker, and M. E. Vidal, "Towards a knowledge graph for science," in *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, ser. WIMS '18, Novi Sad, Serbia: Association for Computing Machinery, 2018, ISBN: 9781450354899. DOI: 10.1145/3227609.3227689.

[5] M. Bearman, C. D. Smith, A. Carbone, *et al.*, "Systematic review methodology in higher education," *Higher Education Research & Development*, vol. 31, no. 5, pp. 625–640, 2012. DOI: 10.1080/07294360.2012.702735.

[6] M. Crowther, W. Lim, and M. A. Crowther, "Systematic riew and meta-analysis methodology," en, *Blood*, vol. 116, no. 17, pp. 3140–3146, Oct. 2010.

[7] G. Halevi and R. Pinotti, "Systematic reviews: Characteristics and impact," *Publ. Res. Q.*, vol. 36, no. 4, pp. 523–537, Dec. 2020.

[8] F. Hoffmann, K. Allers, T. Rombey, *et al.*, "Nearly 80 systematic reviews were published each day: Observational study on trends in epidemiology and reporting over the years 2000-2019," *Journal of Clinical Epidemiology*, vol. 138, pp. 1–11, 2021. DOI: 10.1016/j.jclinepi.2021.05.022.

[9] "Covidence," Covidence. (2023), [Online]. Available: https://www.covidence.org (visited on 06/18/2023).

[10] "Rayyan," Rayyan. (2023), [Online]. Available: https://www.rayyan.ai (visited on 06/18/2023).

[11] T. M. Schiepanski, *Dataset: Integration of systematic review services with a scholarly knowledge graph (data)*, Repository, 2023. DOI: 10.25835/wexlp2ye.

[12] T. M. Schiepanski. "Orkg osyris." Repository, Open Research Knowledge Graph. (2023), [Online]. Available: https://gitlab.com/TIBHannover/orkg/orkg-systematic-review-tools-integration (visited on 10/30/2023).

[13] "Rdf 1.1 turtle," World Wide Web Consortium, Standard, 2014. [Online]. Available: https://www.w3.org/TR/turtle/.

[14] "Rdf schema 1.1," World Wide Web Consortium, Standard, 2014. [Online]. Available: https://www.w3.org/TR/rdf11-schema/.

[15] "Rdf 1.1 concepts and abstract syntax," World Wide Web Consortium, Standard, 2014. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/.

[16] rml.io, "Rdf mapping language (rml)," World Wide Web Consortium, Standard, 2022. [Online]. Available: https://rml.io/specs/rml/.

[17] "Information technology database languages sql," International Organization for Standardization/International Electrotechnical Commission, Standard, 2023.

[18] "Sparql 1.1 overview," World Wide Web Consortium, Standard, 2013. [Online]. Available: https://www.w3.org/TR/sparql11-overview/.

[19] E. Aromataris and A. Pearson, "The systematic review: An overview," *American Journal of Nursing*, vol. 114, no. 3, Mar. 2014. DOI: 10.1097/01.NAJ.0000444496.24228.2c.

[20] S. Gopalakrishnan and P. Ganeshkumar, "Systematic reviews and meta-analysis: Understanding the best evidence in primary healthcare," *Journal of Family Medicine and Primary Care*, vol. 2, no. 1, p. 9, 2013.

[21] L. S. Uman, "Systematic reviews and meta-analyses," *Journal of the Canadian Academy of Child and Adolescent Psychiatry*, vol. 20, no. 1, p. 57, 2011.

[22] M. Gaertler, *Clustering*, U. Brandes and T. Erlebach, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 178–215, ISBN: 978-3-540-31955-9. DOI: 10.1007/978-3-540-31955-9_8.

[23] A. Saxena, M. Prasad, A. Gupta, *et al.*, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, 2017. DOI: https://doi.org/10.1016/j.neucom.2017.06.053.

[24] P. Bhattacharjee and P. Mitra, "A survey of density based ccustering algorithms," *Frontiers of Computer Science*, vol. 15, p. 151 308, Sep. 2020. DOI: 10.1007/s11704-019-9059-3.

[25] L. Richardson, M. Amundsen, and S. Ruby, *RESTful Web APIs: Services for a Changing World*. O'Reilly Media, 2013, ISBN: 9781449359737.

[26] B. Lin, Y. Chen, X. Chen, and Y. Yu, *Comparison between json and xml in applications based on ajax*, 2012. DOI: 10.1109/CSSS.2012.297.

[27] M. Masse, *REST API Design Rulebook* (Oreilly and Associate Series). O'Reilly Media, 2011, ISBN: 9781449310509. [Online]. Available: https://books.google.nl/books?id=4lZcsRwXo6MC.

[28] Z. Mijailovic and D. Milicev, "A retrospective on user interface development technology," *IEEE Software*, vol. 30, no. 6, pp. 76–83, 2013. DOI: 10.1109/MS.2013.45.

[29] E. Bidelman. "Custom elements v1 - reusable web components," web.dev. (2017), [Online]. Available: https://web.dev/articles/custom-elements-v1 (visited on 09/11/2023).

[30] J. Oh, W. H. Ahn, and T. Kim, "Web app restructuring based on shadow doms to improve maintainability," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2017, pp. 118–122. DOI: 10.1109/ICSESS.2017.8342877.

[31] A. Taivalsaari and T. Mikkonen, "The web as an application platform: The saga continues," in *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2011, pp. 170–174. DOI: 10.1109/SEAA.2011.35.

[32] "The iframe element," World Wide Web Consortium, Standard. [Online]. Available: https://www.w3.org/TR/2011/WD-html5-20110525/the-iframe-element.html%5C#the-iframe-element.

[33] M. van Doorn and A. Eliëns, "Integrating applications and the world-wide web," *Computer Networks and ISDN Systems*, vol. 27, no. 6, pp. 1105–1110, 1995. DOI: https://doi.org/10.1016/0169-7552(95)00015-Y.

[34] "Ergonomics of human-system interaction – part 11: Usability: Definitions and concepts," International Organization for Standardization, Geneva, CH, Standard, 2018.

[35] M. Y. Jaradeh, A. Oelen, K. E. Farfar, *et al.*, "Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge," in *Proceedings of the 10th International Conference on Knowledge Capture*, ser. K-CAP '19, Marina Del Rey, CA, USA: Association for Computing Machinery, 2019, pp. 243–246, ISBN: 9781450370080. DOI: 10.1145/3360901.3364435.

[36] S. Auer, A. Oelen, M. Haris, *et al.*, "Improving access to scientific literature with knowledge graphs," *Bibliothek Forschung und Praxis*, vol. 44, no. 3, pp. 516–529, 2020. DOI: doi:10.1515/bfp-2020-2042. [Online]. Available: https://doi.org/10.1515/bfp-2020-2042.

[37] M. Stocker, A. Oelen, M. Y. Jaradeh, *et al.*, "Fair scientific information with the open research knowledge graph," *FAIR Connect*, vol. 1, pp. 19–21, 2023. DOI: 10.3233/FC-221513.

[38] TIB – Leibniz Information Centre for Science and Technology. "Open research knowledge graph," Open Research Knowledge Graph. (2023), [Online]. Available: https://orkg.org (visited on 01/07/2023).

[39] TIB – Leibniz Information Centre for Science and Technology. "Orkg ontology." Repository, Open Research Knowledge Graph. (2023), [Online]. Available: https://gitlab.com/TIBHannover/orkg/orkg-ontology (visited on 08/15/2023).

[40] TIB – Leibniz Information Centre for Science and Technology. "Papers," Open Research Knowledge Graph. (2023), [Online]. Available: https://orkg.org/about/20/Papers (visited on 09/27/2023).

[41] TIB – Leibniz Information Centre for Science and Technology. "Comparisons," Open Research Knowledge Graph. (2023), [Online]. Available: https://orkg.org/about/15/Comparisons (visited on 09/27/2023).

[42] I. Traverso, M.-E. Vidal, B. Kämpgen, and Y. Sure-Vetter, "Gades: A graph-based semantic similarity measure," ser. SEMANTiCS 2016, Leipzig, Germany: Association for Computing Machinery, 2016, pp. 101–104, ISBN: 9781450347525. DOI: 10.1145/2993318.2993343.

[43] G. Zhu and C. A. Iglesias, "Computing semantic similarity of concepts in knowledge graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 72–85, 2017. DOI: 10.1109/TKDE.2016.2610428.

[44] L. Li, Z. Zhang, and S. Zhang, "Knowledge graph entity similarity calculation under active learning," *Complexity*, vol. 2021, pp. 1–11, 2021.

[45] B. Zhu, T. Bao, R. Han, *et al.*, "An effective knowledge graph entity alignment model based on multiple information," *Neural Networks*, vol. 162, pp. 83–98, 2023, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2023.02.029.

[46] D. Collarana, M. Galkin, I. Traverso-Ribón, C. Lange, M.-E. Vidal, and S. Auer, "Semantic data integration for knowledge graph construction at query time," in *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, 2017, pp. 109–116. DOI: 10.1109/ICSC.2017.85.

[47] O. A. Oghli, "Information retrieval service aspects of the open research knowledge graph," M.S. thesis, Leibniz Universität Hannover, 2022. DOI: 10.15488/11834.

[48] "Orkg simcomp api." Repository, TIB – Leibniz Information Centre for Science and Technology. (2023), [Online]. Available: https://gitlab.com/TIBHannover/orkg/orkg-simcomp/orkg-simcomp-api (visited on 09/03/2023).

[49] TIB – Leibniz Information Centre for Science and Technology. "Orkg frontend." Repository, Open Research Knowledge Graph. (2023), [Online]. Available: https://gitlab.com/TIBHannover/orkg/orkg-frontend (visited on 10/07/2023).

[50] "Elasticsearch," Elastic NV. (2023), [Online]. Available: https://www.elastic.co/de/elasticsearch (visited on 09/14/2023).

[51] "Cadima," JKI - Julius Kühn-Institut - Federal Research Centre for Cultivated Plants. (2023), [Online]. Available: https://www.cadima.info (visited on 06/15/2023).

[52] C. Kohl, E. J. McIntosh, S. Unger, *et al.*, "Online tools supporting the conduct and reporting of systematic reviews and systematic maps: A case study on cadima and review of existing tools," en, *Environ. Evid.*, vol. 7, no. 1, Dec. 2018.

[53] J. Babineau, "Product review: Covidence (systematic review software)," *Journal of the Canadian Health Libraries Association/Journal de l'Association des bibliothèques de la santé du Canada*, vol. 35, no. 2, pp. 68–71, 2014.

[54] "Eppi-reviewer," EPPI-Centre. (2023), [Online]. Available: https://eppi.ioe.ac.uk/cms/Default.aspx?tabid=2914 (visited on 10/25/2023).

[55] J. Thomas and J. Brunton, "Eppi-reviewer 4: Software for research synthesis," Jan. 2010.

[56] "Pico portal," PICO Portal. (2023), [Online]. Available: https://picoportal.org (visited on 09/12/2023).

[57] J. Minion, O. Egunsola, L. Mastikhina, *et al.*, "Pico portal (product review)," *Journal of the Canadian Health Libraries Association / Journal de l'Association des bibliothèques de la santé du Canada*, vol. 42, Dec. 2021. DOI: 10.29173/jchla29590.

[58] "Sysrev," Insilica LLC. (2023), [Online]. Available: https://sysrev.com (visited on 09/12/2023).

[59] T. Bozada, J. Borden, J. Workman, M. Del Cid, J. Malinowski, and T. Luechtefeld, "Sysrev: A fair platform for data curation and systematic evidence review," *Frontiers in Artificial Intelligence*, vol. 4, 2021. DOI: 10.3389/frai.2021.685298.

[60] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96, Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[61] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," SIGMOD '99, vol. 28, no. 2, pp. 49–60, 1999, ISSN: 0163-5808. DOI: 10.1145/304181.304187.

[62] L. McInnes, J. Healy, and S. Astels, "Hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017. DOI: 10.21105/joss.00205.

[63] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002. DOI: 10.1109/34.1000236.

[64] "Add a google map with a marker to your website," Google LLC. (2023), [Online]. Available: https://developers.google.com/maps/documentation/javascript/adding-a-google-map (visited on 09/28/2023).

[65] "Embed an instagram post or profile," Meta Platforms, Inc. (2023), [Online]. Available: https://help.instagram.com/620154495870484 (visited on 09/28/2023).

[66] "Orkg widget documentation," TIB – Leibniz Information Centre for Science and Technology. (2023), [Online]. Available: https://orkg.org/help-center/article/14/ORKG%5C%5FWidget%5C%5FDocumentation (visited on 10/24/2023).

[67] "Overlays," AUDI AG. (2023), [Online]. Available: https://www.audi.com/ci/en/guides/user-interface/components/overlays.html (visited on 09/28/2023).

[68] J. Rowe. "Web widget," Zendesk, Inc. (2023), [Online]. Available: https://support.zendesk.com/hc/articles/4408836216218 (visited on 09/28/2023).

[69] L. Paulson, "Building rich web applications with ajax," *Computer*, vol. 38, no. 10, pp. 14–17, 2005. DOI: 10.1109/MC.2005.330.

[70] "Interface xmlhttprequest," Web Hypertext Application Technology Working Group (Apple, Google, Mozilla, Microsoft). (2023), [Online]. Available: https://xhr.spec.whatwg.org/%5C#interface-xmlhttprequest (visited on 09/28/2023).

[71] M. Han and P. Park, "A study of interface design for widgets in web services through usability evaluation," in *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, ser. ICIS '09, Seoul, Korea: Association for Computing Machinery, 2009, pp. 1013–1018, ISBN: 9781605587103. DOI: 10.1145/1655925.1656109.

[72] S. M. Zabed Ahmed, C. McKnight, and C. Oppenheim, "A review of research on human-computer interfaces for online information retrieval systems," *The Electronic Library*, vol. 27, no. 1, pp. 96–116, Jan. 2009. DOI: 10.1108/02640470910934623.

[73] P. Jen-Hwa Hu, P.-C. Ma, and P. Y. Chau, "Evaluation of user interface designs for information retrieval systems: A computer-based experiment," *Decision Support Systems*, vol. 27, no. 1, pp. 125–143, 1999, ISSN: 0167-9236. DOI: https://doi.org/10.1016/S0167-9236(99)00040-8.

[74] C. Schneider, M. Weinmann, and J. vom Brocke, "Digital nudging," en, *Commun. ACM*, vol. 61, no. 7, pp. 67–73, Jun. 2018.

[75] K. U. Priesnitz, A. Vaasen, and A. Gathmann, "Baseline susceptibility of different european lepidopteran and coleopteran pests to bt proteins expressed in bt maize: A systematic review," *Environmental Evidence*, vol. 5, no. 1, p. 27, Dec. 2016, ISSN: 2047-2382. DOI: 10.1186/s13750-016-0077-4.

[76] "Study quality sheet," Springer Nature. (2023), [Online]. Available: https://static-content.springer.com/esm/art%5C%3A10.1186%5C%2Fs13750-016-0077-4/MediaObjects/13750%5C%5F2016%5C%5F77%5C%5FMOESM7%5C%5FESM.xlsx (visited on 10/18/2023).

[77] "Evidencesynthesisdatabase," JKI - Julius Kühn-Institut - Federal Research Centre for Cultivated Plants. (2023), [Online]. Available: https://www.cadima.info/index.php/area/evidenceSynthesisDatabase (visited on 10/08/2023).

[78] "Orkg annotation." Repository, TIB – Leibniz Information Centre for Science and Technology. (2023), [Online]. Available: https://gitlab.com/TIBHannover/orkg/annotation (visited on 09/05/2023).

[79] "Orkg email address extraction." Repository, TIB – Leibniz Information Centre for Science and Technology. (2023), [Online]. Available: https://gitlab.com/TIBHannover/orkg/orkg-email-address-extraction (visited on 09/05/2023).

[80] "Orkg nlp api." Repository, TIB – Leibniz Information Centre for Science and Technology. (2023), [Online]. Available: https://gitlab.com/TIBHannover/orkg/nlp/orkg-nlp-api (visited on 09/05/2023).

[81] "Orkg simpaper api." Repository, TIB – Leibniz Information Centre for Science and Technology. (2023), [Online]. Available: https://gitlab.com/TIBHannover/orkg/orkg-simpaper-api (visited on 09/05/2023).

[82] I. Technology, "Neo4j, the world's leading graph database," *Neo4j Graph Database*, Jan. 2015.

[83] "Neo4j," Neo4j, Inc. (2023), [Online]. Available: https://neo4j.com (visited on 09/14/2023).

[84] TIB – Leibniz Information Centre for Science and Technology. "Orkg backend." Repository, Open Research Knowledge Graph. (2023), [Online]. Available: https://gitlab.com/TIBHannover/orkg/orkg-backend (visited on 10/07/2023).

[85] "Scikit-learn clustering," Scikit-learn Developers. (2023), [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html (visited on 10/02/2023).

[86] "Uvicorn." Repository, Encode. (2023), [Online]. Available: https://github.com/encode/uvicorn (visited on 09/02/2023).

[87] B. Chesneau. "Gunicorn." Repository. (2023), [Online]. Available: https://github.com/benoitc/gunicorn (visited on 09/02/2023).

[88] S. Ramırez. "Fastapi." Repository. (2023), [Online]. Available: https://github.com/tiangolo/fastapi (visited on 09/02/2023).

[89] Microsoft. "Typescript interfaces." (2023), [Online]. Available: https://www.typescriptlang.org/docs/handbook/interfaces.html (visited on 10/09/2023).

[90] "Recommendations," JSON:API, Tech. Rep., 2024. [Online]. Available: https://jsonapi.org/recommendations.

[91] "The oauth 2.0 authorization framework," Internet Engineering Task Force (IETF), Tech. Rep., 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6749.

[92] A. Aravinth and S. Machiraju, *Higher Order Functions*. Berkeley, CA: Apress, 2018, pp. 41–62, ISBN: 978-1-4842-4087-8. DOI: 10.1007/978-1-4842-4087-8_3.

[93] TIB – Leibniz Information Centre for Science and Technology. "Orkg rest api documentation." v0.40.0, Open Research Knowledge Graph. (2023), [Online]. Available: https://tibhannover.gitlab.io/orkg/orkg-backend/api-doc (visited on 09/02/2023).

[94] TIB – Leibniz Information Centre for Science and Technology. "Orkg (sparql)," Open Research Knowledge Graph. (2023), [Online]. Available: https://orkg.org/sparql (visited on 08/19/2023).

[95] P. Leach, M. Mealling, and R. Salz, "A universally unique identifier (uuid) urn namespace," The Internet Society, RFC, 2005.

[96] I. Pashchenko, D.-L. Vu, and F. Massacci, "A qualitative study of dependency management and its security implications," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20, Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1513–1531, ISBN: 9781450370899. DOI: 10.1145/3372297.3417232.

[97] A. Oelen and M. Y. Jaradeh, private communication, Oct. 2023.

[98] "Xml schema part 2: Datatypes second edition," World Wide Web Consortium, Standard, 2004. [Online]. Available: https://www.w3.org/TR/xmlschema-2/.

[99] "Sentence-transformers." Repository, Ubiquitous Knowledge Processing Lab, Technische Universität Darmstadt. (2023), [Online]. Available: https://github.com/UKPLab/sentence-transformers (visited on 08/30/2023).

[100] J. Zhang, *Multidimensional Scaling*, J. Zhang, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 143–163, ISBN: 978-3-540-75148-9. DOI: 10.1007/978-3-540-75148-9_7.

[101] P. Leach, M. Mealling, and R. Salz, "The open group base specifications issue 7," Institute of Electrical and Electronics Engineers/The Open Group, Standard, 2018.

[102] H. Maarif, R. Akmeliawati, Z. Htike, and T. S. Gunawan, "Complexity algorithm analysis for edit distance," in *2014 International Conference on Computer and Communication Engineering*, 2014, pp. 135–137. DOI: 10.1109/ICCCE.2014.48.

[103] A. Hallez, A. Bronselaer, and G. de Tré, "Comparison of sets and multisets," *Internat. J. Uncertain. Fuzziness Knowledge-Based Systems*, vol. 17, pp. 153–172, Aug. 2009.

[104] "The oauth 2.0 authorization framework," International Organization for Standardization, Geneva, CH, Standard, 2017.

[105] J. Zhang, C. Xue, J. Wang, *et al.*, "Effects of cognitive redundancy on interface design and information visualization," in *Advances in Usability and User Experience*, T. Ahram and C. Falcão, Eds., Cham: Springer International Publishing, 2018, pp. 483–491, ISBN: 978-3-319-60492-3.

[106] C. González and G. M. Kasper, "Animation in user interfaces designed for decision support systems: The effects of image abstraction, transition, and interactivity on decision quality*," *Decision Sciences*, vol. 28, no. 4, pp. 793–823, 1997.

[107] S. McConnell, *Software Estimation*. Redmond, WA: Microsoft Press, Feb. 2006.

[108] C. Jones, O. Bonsignour, and J. Subramanyam, *The Economics of Software Quality*. Boston, MA: Addison-Wesley Educational, Jul. 2011.

[109] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences," *Minds and Machines*, vol. 30, no. 4, pp. 681–694, Dec. 2020. DOI: 10.1007/s11023-020-09548-1.

[110] J. Pérez, M. Arenas, and C. Gutierrez, "Semantics and complexity of sparql," *ACM Trans. Database Syst.*, vol. 34, no. 3, Sep. 2009, ISSN: 0362-5915. DOI: 10.1145/1567274.1567278.

[111] A. Oelen, J. D'Souza, M. Stocker, *et al.* "Covid-19 reproductive number estimates." (2020), [Online]. Available: https://orkg.org/comparison/R44930 (visited on 10/04/2023).

[112] Y. Liu, A. A. Gayle, A. Wilder-Smith, and J. Rocklöv, "The reproductive number of covid-19 is higher compared to sars coronavirus," *Journal of Travel Medicine*, vol. 27, no. 2, Feb. 2020. DOI: 10.1093/jtm/taaa021.

[113] "Defining n-ary relations on the semantic web," World Wide Web Consortium, Standard, 2006. [Online]. Available: https://www.w3.org/TR/swbp-n-aryRelations/.

[114] M. P. Simunovic, "Colour vision deficiency," *Eye*, vol. 24, no. 5, pp. 747–755, May 2010. DOI: 10.1038/eye.2009.251.

[115] A. Oelen, M. Stocker, and S. Auer, "Creating a scholarly knowledge graph from survey article tables," in *Digital Libraries at Times of Massive Societal Transition*, Springer International Publishing, 2020, pp. 373–389. DOI: 10.1007/978-3-030-64452-9_35.

Appendix A

RML Example

Listing A.1: In order to share a comprehensive view on how Systematic Review is transferred to a Knowledge Graph, the RML mappings below connect with the RDF example given in the work. Application of RML mappings represents part of the syntactic translation pipeline.

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix orkgc: <http://orkg.org/orkg/class/> .
@prefix orkgp: <http://orkg.org/orkg/predicate/> .

<#PaperMapping> a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "<JSON:IReadData>" ;
        rml:referenceFormulation ql:JSONPath
    ] ;
    rr:subjectMap [
        rr:template "http://orkg.org/orkg/resource/{paper_id}" ;
        rr:class orkgc:Paper
    ] ;
    rr:predicateObjectMap [
```

```
        rr:predicate rdfs:label ;
        rr:objectMap [ rml:reference "paper_title" ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate orkgp:authors ;
        rr:objectMap [ rml:reference "author_names" ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate orkgp:P31 ;
        rr:objectMap [
            rml:constant "http://orkg.org/orkg/resource/{contribution_id}"
        ]
    ] .

<#ContributionMapping> a rr:TriplesMap ;
    rml:logicalSource [
        rml:source "<JSON:IReadData.po_records>" ;
        rml:referenceFormulation ql:JSONPath
    ] ;
    rr:subjectMap [
        rr:template "http://orkg.org/orkg/resource/{contribution_id}" ;
        rr:class orkgc:Contribution
    ] ;
    rr:predicateObjectMap [
        rr:predicate rdfs:label ;
        rr:objectMap [ rml:constant "Contribution 1" ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate orkgp:P40003 ;
        rr:objectMap [ rml:reference "Extraction methods" ]
    ] ;
    rr:predicateObjectMap [
        rr:predicate orkgp:P2004 ;
        rr:objectMap [ rml:reference "Worst complexity" ]
```

```
52      ] ;
53      rr:predicateObjectMap [
54          rr:predicate orkgp:P20912 ;
55          rr:objectMap [
56              rml:reference "f1 measure" ;
57              rr:datatype xsd:float
58          ]
59      ] .
```

Appendix B

Widget Modals

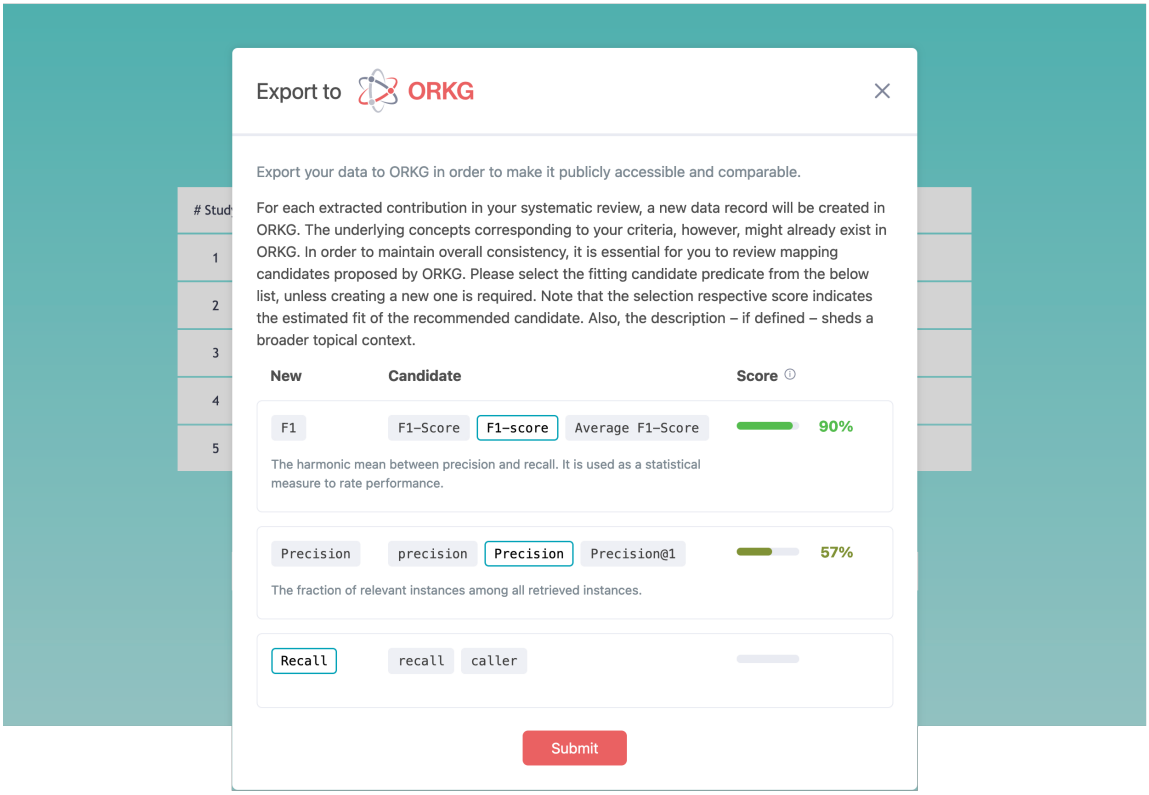


Figure B.1: Screenshot of the OSyRIS client application widget predicate recommendations modal. The recommendations are the result of the the predicate disambiguation measure applied in the OSyRIS server application. The Systematic Review Service user is asked to select recommended predicates if suitable or their own ones to force creation of a new predicate, respectively.

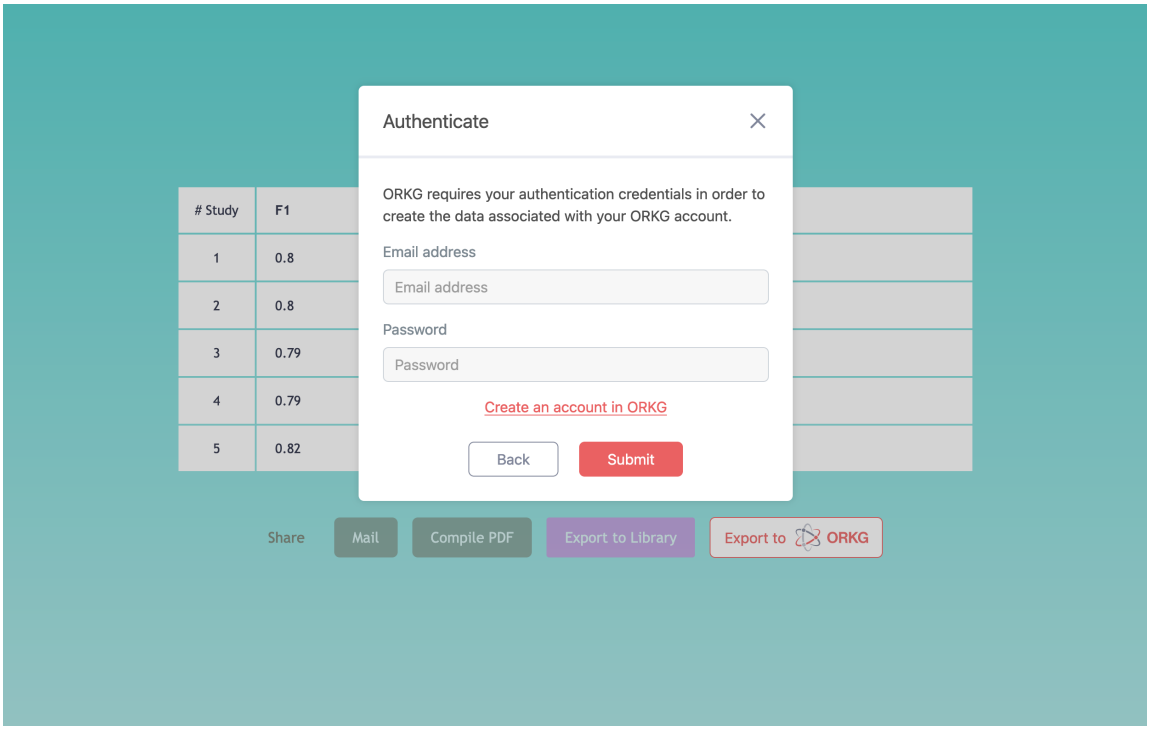


Figure B.2: Screenshot of the OSyRIS client application widget authentication modal. The Systematic Review Service user has authenticate at ORKG in order to associated the to-be-create resources with a known ORKG user. In order the Systematic Review Service user is not yet registered in ORKG, they can use the redirect to the registration form, intermediately.

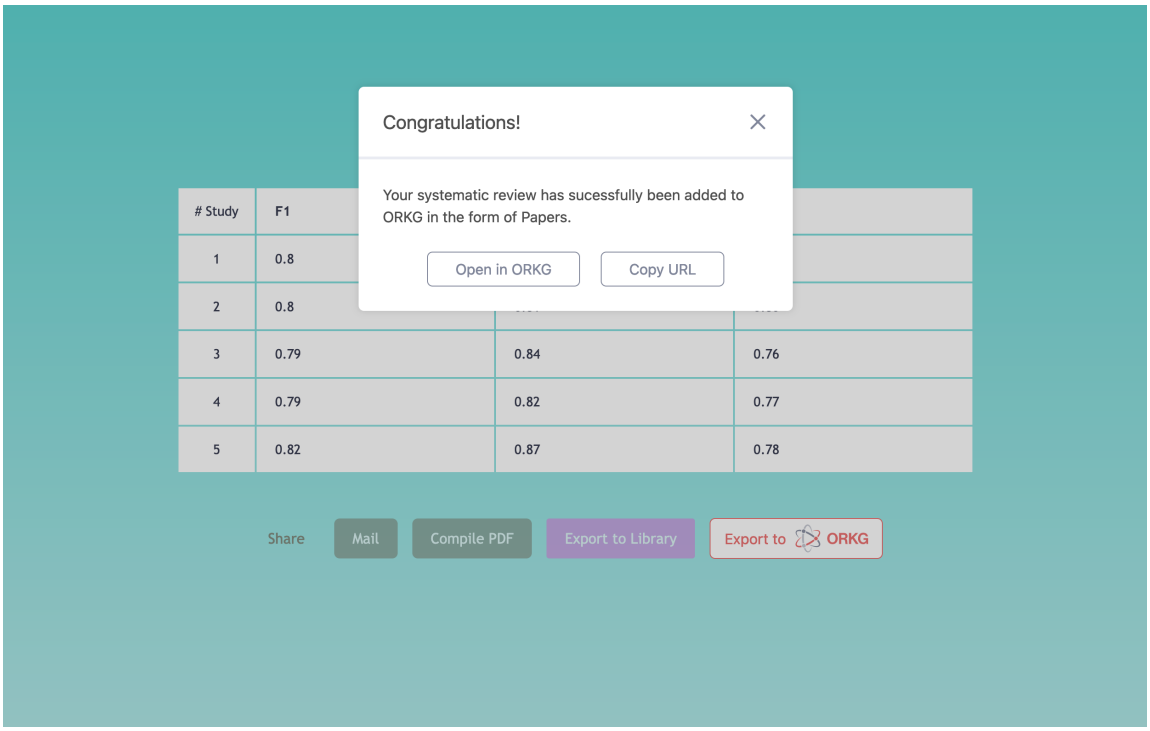


Figure B.3: Screenshot of the OSyRIS client application widget success modal. The Systematic Review data export to ORKG was successful. The Systematic Review Service user is provided with a link to the virtual comparison in ORKG that represents the exported Systematic Review.

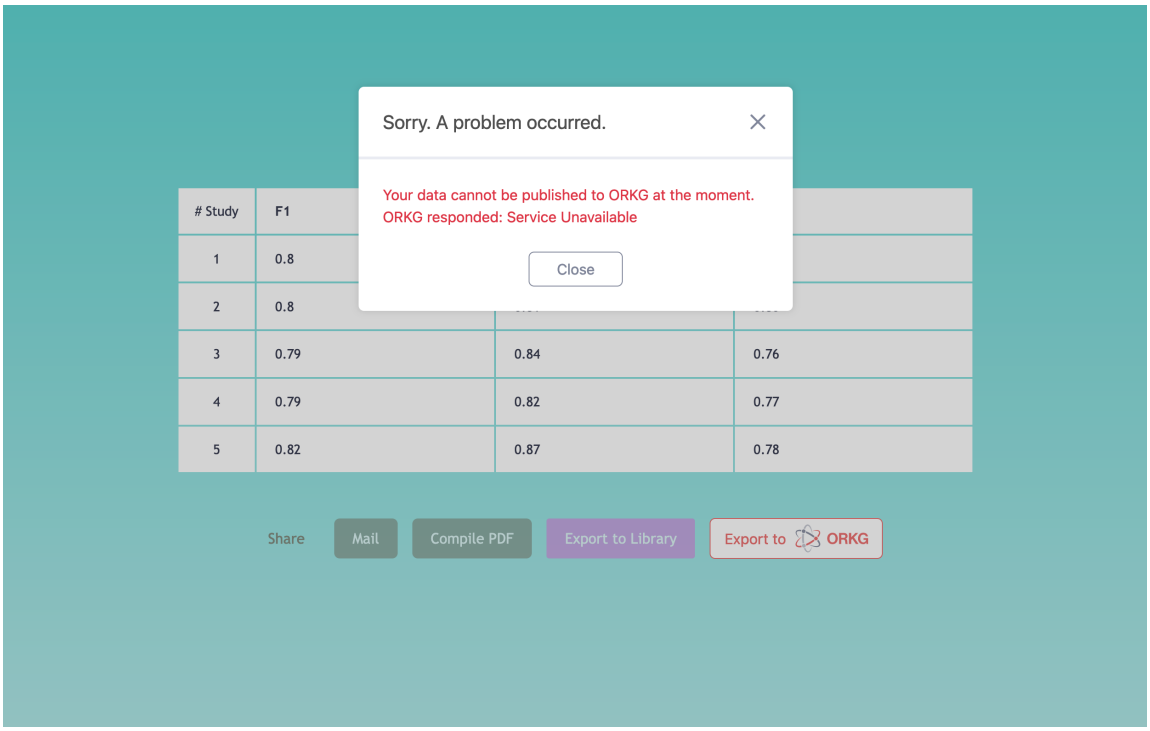


Figure B.4: Screenshot of the OSyRIS client application widget error modal. The error modal is shown after an uncaught or passed through error occurred. For known errors, a specific error message is shown to the Systematic Review Service user. For unknown errors, a default message is shown in place.