Ein URI Konzept für den deutschen Tourismus

Kevin Angele, Dieter Fensel, Andreas Harth, Elwin Huaman, Oleksandra Panasiuk, and Umutcan Şimşek

Version: December 28, 2019

Executive Summary

Das Internet hat eine Revolution unserer Kommunikation verursacht. Frühe Formen wie Email und das Web haben bereits fundamentale Änderungen der Art und Weise bewirkt, wie wir kommunizieren und Produkte und Dienste erwerben bzw. konsumieren. Und dieser Prozess bleibt nicht einfach stehen. Immer neue Kommunikationskanäle erblühen auf dieser Infrastruktur und bieten Möglichkeiten für erfolgreiches eMarketing und eCommerce. Die stetig wachsende Vielfalt der Kanäle erfordert eine kanalunabhängige Modellierung von Information, Daten und Diensten. Unabhängig von der konkreten Darstellung in einem bestimmten Kanal wird eine maschinenlesbare Bereitstellung existentiell. Damit können die verschiedensten Bots diese Inhalte „verstehen" und ihren Benutzern zur Verfügung stellen. Anders droht die Unsichtbarkeit von Dienstanbietern. Und was sich nicht in unserer breiten Wahrnehmung manifestiert, existiert irgendwann nicht mehr.

Information, Daten und Dienste müssen also weltweit sichtbar und maschinenlesbar angeboten werden.[1] Die Deutsche Zentrale für Tourismus (DZT) hat diese Herausforderung erkannt und mit dem Aufbau eines **offenen deutschen touristischen Wissensgraphen** begonnen. Nur so kann der wirtschaftliche Stellenwert dieser Branche gesichert werden, die sich in Zukunft wahrscheinlich in ihrer relativen Bedeutung in Deutschland erhöhen wird. Eine offene und maschinenlesbare Beschreibung der touristischen Angebote ermöglicht eine optimierte Sichtbarkeit in all den neuen Kommunikationskanälen auf einer internationalen Ebene.

Ein bedeutendes Problem in der Erstellung von Wissensgraphen ist die Identifikation von Duplikaten. Wenn Daten aus unterschiedlichen und heterogenen Datenquellen integriert werden muss untersucht werden, welche Daten sich überlappen und daher unterschiedliche Beschreibungen desselben Objektes liefern. Sich dieser Aufgabe nicht zu stellen, heißt für dasselbe Objekt unterschiedliche (u.U. sich widersprechende) Beschreibungen anzubieten. Erfolgsversprechender ist es, Duplikate zu erkennen und deren Beschreibungen zu integrieren und zu vereinheitlichen. Dieses Problem der Informationsintegration wurde erstmals 1946 explizit unter dem Namen Record Linkage formuliert [Dunn, 1946]. Record Linkage[2] hat mittlerweile eine Vielfalt von Namen erhalten: Author-Name Disambiguation[3], Data Deduplication[4], Entity Linking[5], Identity Resolution[6], Schema Matching[7] und Single-Instance Storage[8] (siehe Abbildung 1). Immer geht es um das Problem, verschiedene syntaktische Bezeichner auf dasselbe semantische Objekt zu reduzieren.[9]

Es ist daher eine sehr zukunftsweisende Entscheidung des DZT dieses Problem nicht zu ignorieren, sondern unmittelbar bei dem Aufbau des Wissensgraphen anzugehen. Andernfalls würde

[1] Vorallen in einer Branche wie dem Toursimus, der einerseits viele, kleine und zersplitterte Anbieter und eine weltweite Verteilung potentieller Gäste aufweist.

[2] https://en.wikipedia.org/wiki/Record_linkage

[3] https://en.wikipedia.org/wiki/Author_name_disambiguation

[4] https://en.wikipedia.org/wiki/Data_deduplication

[5] https://en.wikipedia.org/wiki/Entity_linking

[6] https://en.wikipedia.org/wiki/Record_linkage#Identity_resolution

[7] https://en.wikipedia.org/wiki/Schema_matching

[8] https://en.wikipedia.org/wiki/Single-instance_storage

[9] "While entity disambiguation and resolution is an active research area in the semantic web, and now in Knowledge Graphs for several years, it is almost surprising that it continues to be one of the top challenges in the industry almost across the board. In its simplest form, the challenge is in assigning a unique normalized identity and a type to an utterance or a mention of an entity." [Noy et al., 2019]. Nein, es ist erkennbar keine Überraschung wenn man die prinzipiellen Probleme brücksichtigt, die mit Identität verbunden sind [Halpin et al., 2010].

sich die Gefahr ergeben, dass ein scheinbar kleines Unterproblem sich als schwarzen Loch für den Gesamtansatz erweisen würde.

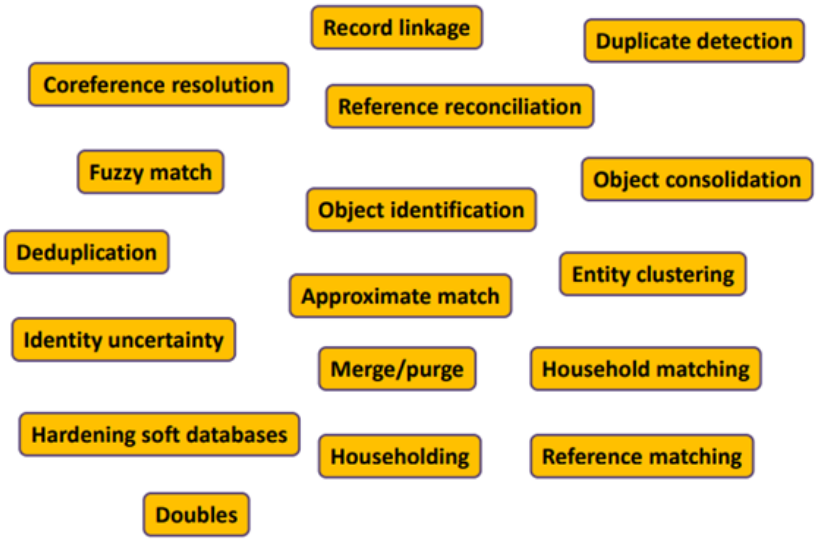


Abbildung 1. Namen für das Record Linkage Problem [Getoor & Machanavajjhala, 2012].

In der vorliegenden Arbeit wird daher der Prozess für die Bereitstellung von Bezeichnern für heterogene Datensätze im deutschen Tourismus spezifiziert und prototypisch implementiert. Nach einer Einleitung werden in Kapitel 2 die Definitionen für Uniform Resource Identifiers (URIs) bereitgestellt. Die Verwendung dieses Standards als Bezeichner für Objekte ist in einem internet- und webbasierten Umgebung die einzig plausible Entscheidung. DZT hat für einen ersten Ansatz vier relevante Themengebiete im Tourismus identifiziert:

- Ereignisse (Events),
- Sehenswürdigkeiten (Points of Interest, POIs),
- Touren und
- Unterkünfte.

Für jeden dieser Bereiche wurden mögliche webbasierte URI Quellen identifiziert (Kapitel 3) und in Kapitel 4 Auswahlkriterien für diese Quellen definiert (siehe Knowledge Assessment und Data Quality, [Fensel et al., 2020]). In Kapitel 5 werden Algorithmen zur Auswahl dieser Quellen spezifiziert und in Kapitel 6 verschiedene Ansätze für die Generierung neuer URIs entwickelt, die den Fall abdecken, dass kein existierender Bezeichner für ein neues Objekt gefunden werden konnte. Kapitel 7 und 8 spezifizieren ein prototypisch entwickeltes System, welches diese Konzepte effizient implementiert. Verwandte Arbeiten, alternative Ansätze und Schlussfolgerungen werden in den Kapiteln 9, 10 und 11 erläutert.

Das DZT hat zum richtigen Zeitpunkt die Initiative für einen deutschen touristischen Open Knowledge Graph gestartet. Ein erster Schritt ist die Definition eindeutiger Bezeichner für touristische Objekte. Im zweiten Schritt ist die Erfassung standardisierter und maschinenlesbarer Beschreibungen dieser Objekte der eigentliche Kern dieses Ansatzes. Wir freuen uns hierbei die DZT bei der Spezifikation, Umsetzung und Unterhaltung dieses Knowledge Graphen behilflich zu sein, um so dem deutschen Tourismus die internationale Sichtbarkeit und Buchbarkeit[10] zu garantieren.

[10] Ohne dass dabei unmittelbar ca. 20 % Provision unmittelbar bei der Buchung nach Kalifornien überwiesen werden.

Contents

1 Introduction

The „Deutsche Zentrale für Tourismus e.V.“ (DZT) decided together with LMOs (Landesmanagement Organizationen) and Magic-Cities to publish their data as Linked-Open-Data (LOD). Furthermore, the idea is to aggregate the touristic data throughout Germany from the LMOs, DMOs (Destination Management Organizations) and key touristic players into a Knowledge Graph (KG). In the first step, they want to start with the domains Accommodation, Event, Point Of Interest (POI) and Tour.

In most data source there are identifiers for certain entities. When integrating data from different sources, every source may use different identifiers for their entities. One issue that may occur while integrating data from various heterogeneous sources are duplicates. (e.g., hiking paths passing multiple states would be identified by different URIs by different LMOs). The DZT does not assess the data, because of that it is possible that multiple descriptions of the same entity would exist in the KG. In order to mark the duplicates, all entities should get a URI. In the context of the Web, resources are typically identified with URIs (Section 2). When publishing internal data sources, as the LMOs and DMOs do, instances may not have a real identifier for the web and are represented with *blank nodes*. Those nodes just have a local scope and are difficult to be referenced from outside. That is why data integration is more difficult, as an entity cannot be uniquely identified. What first comes into mind is that the data publisher generates URIs based on the internal entity identifiers for such anonymous instances. However this does not solve the problem because those URIs just identify the instances locally. For efficient and effective usage of the aggregated data the entities must be linked.

To solve the integration problem, we need to provide an approach to search for canonical URIs from external sources. Those URIs are needed to uniquely identify the entities in the aggregated Knowledge Graph (DZT as *URI unifier* wants to have a unique identifier for every entry in the Knowledge Graph). Instead of replacing the local identifiers with the external URIs, we link them. The idea is to provide a service the LMOs and DMOs (those are the *content creators*) can use to search for a canonical URI for an entity (e.g., Hamburg Tourism Agency looks for a canonical URI for Elbphilarmonie). The service searches for a URI in selected URI sources (Section 3) and provides a URI from the highest ranked source. If for a given entity no URI can be found, then the service constructs a new URI which is also used for the following requests for the same entity.

To achieve such a service we provide a prototype of a CMS plugin for content creators to provide an entity and a URI recommender on DZT side to return a canonical URI for the provided entity. The URI recommender consists of a preprocessor, URI source ranker, URI selector and a URI constructor. First, the DZT as URI unifier must define weights for different data quality metrics and dimensions (presented in section 4) that are used to rank the URI sources. After a preprocessing step, the *URI source ranker* calculates the rankings for the used URI sources (e.g. Wikidata, DBPedia, …). When a DMO or LMO creates a new entity and provides the information via the CMS plugin, DZT’s *URI selector* searches for a URI in the selected sources and returns the URI from the highest ranked source. If there is no URI found for the given entity the *URI constructor* creates a new URI and stores it for upcoming requests together with the information about the entity. The quality of the URI output is heavily based on the selection of the URI source. It is important to choose sources with very high quality.

The paper is structured as follows: In the second section, we give a definition of URIs. Afterward, we list a selection of URI sources we found for the given domains (Section 3). Section 4 presents the concept of how these sources are defined, including their preprocessing and definition of

the criteria, metrics, measurement functions and aggregation functions for rating URI sources. This section is followed by a description of the URI selection approach with the help of similarity measurements (Section 5). If no URI is found for the given entity, it is necessary to provide a canonical URI. The approaches for creating such a canonical URI is described in Section 6. The overall process model is presented in Section 7. Section 8 describes our implementations, namely the CMS plugin for content creators and the URI recommender that implements the approaches defined in Sections 4,5 and 6 by following the process model defined in Section 7. The related work is described in Section 9. Alternative approaches will be shown in Section 10 and conclusions are provided in Section 11.

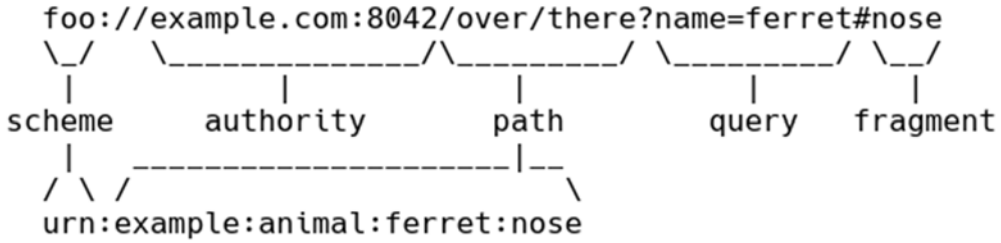
2 URIs in a Nutshell

Uniform Resource Identifier (URI) is an IETF[11] standard[12] for identifying abstract and physical resources. We can have a better understanding of what URIs are by breaking this concept down as follows:

- **Uniform**: All URIs must follow a formal syntax to guarantee uniformity. The syntax is also extensible, so new types of URIs can be created without breaking the way the existing ones work. Uniformity ensures that an application that can process URIs can also process specific types of URIs by using its generic features.
- **Resource**: The definition of a resource is very broad, that is anything that can be identified by a URI can be a resource. This can be a document on the web, a sensor in the Internet of Things, a service, or something that cannot be accessed directly by the means of the internet or the web, e.g. a person or a building.
- **Identifier**: An identifier refers to the information needed to distinguish a resource from all other resources within its context (e.g. same file path may identify different resources on different computers). The identification of a resource does not necessarily imply that the resource is accessible. An identified resource does not have to be a singular static item but can be a mapping that changes over time. (e.g. one can have an identifier for "upcoming events in a city", that may refer to a different set of objects at any given time)

In generic words, a URI is a sequence of characters that follow a standardized syntax that is used to identify a resource. The nature of the identification (e.g. only in terms of name or also describing the way of accessing the identified resource) is determined by its scheme.

There are two different types of URIs, namely URL (Uniform Resource Locator) and URN (Uniform Resource Name). URL is a subset of URIs that not only identify a resource but also provide a means of locating and accessing them (e.g. usually through a protocol, like an HTTP URL). URNs, on the other hand, have only name characteristics (e.g. an ISBN number). Having said that, a URL may also have only the function of "name", meaning that it does not guarantee that the resource is accessible. Following shows two example URIs:



- **scheme**: The URI scheme indicates the type of the URI. Typically, schemes refer to a protocol that dictates how to handle the URI, but this is not always the case.
- **authority**: Some URIs refer to authorities that define a structure for the URI. In the context of the web, the authority is usually a domain name or server IP, possibly accompanied by a username and a port.
- **path**: paths usually organize the resource being identified in a hierarchical manner.
- **query**: The optional query part typically provides non-hierarchical filtering. It follows a "?" sign.

[11] Internet Engineering Task Force

[12] See the RFC document for a detailed definition of URIs: https://tools.ietf.org/html/rfc3986

- **fragment**: The fragment part follows a "#" sign and typically refers to a specific part of the identified resource.

The standardization of URI syntax initially started to address the requirements and recommendations for locating internet resources[13] and defining names for resources[14]. Without such a standard and uniformity, it would have been impossible to develop client applications that consume the resources on a network in a scalable way. As the current consensual formalism for knowledge graphs and semantic web, RDF also makes use of URIs for naming resources. The graph nature of RDF fosters the composition of distributed data. This is only straightforwardly possible thanks to the adoption of URIs. Different resources identified by the same URI are seamlessly connected as long as they are identified by the same URI [Hitzler et al., 2010].

Although the challenge of finding the right URI for a resource is big, and the major focus of this work, there are widespread best practices for designing URIs for resources in knowledge graphs[15]. Given that a knowledge graph published on the web, the most important principle is to avoid ambiguity between a resource and a web document that possibly about that resource. For example, https://www.germany.travel/en/towns-cities-culture/palaces-parks-gardens/neuschwanstein-castle.html is a URL that identifies a web document about Neuschwanstein Castle, however, may not be the best URI to identify the POI itself as a resource. First, a URI should identify one resource, in this case, it is an HTML document. The English language document about the same POI would have a different URL. An application other than a web browser may prefer a more machine-understandable and uniform description of this POI. A solution to this ambiguity is using a different URI for the entity itself e.g. https://www.germany.travel/entity/Neuschwanstein_Castle. By benefiting from the content negotiation feature of HTTP, this URI may be used to return an HTML document when requested by a web browser[16] and an RDF serialization when requested by an intelligent application. Within the knowledge graph, the resource can be linked to the documents that describe that resource. [Sauermann et al., Eds, 2007]

From a design point of view, URIs must be "cool" [17]. Cool URIs are simple and persistent URIs. They are more likely to be persistent, thus less unlikely to break applications. Why is https://www.germany.travel/en/towns-cities-culture/palaces-parks-gardens/neuschwanstein-castle.html not a cool URI, especially in the context of a resource in a knowledge graph? First of all, it is too long[18]. Second, it directly reflects the way the topics are organized on the website, which can easily change in the future. Moreover, it contains the language of the document it identifies and its file format. These are subject to change in the short or middle term too. All these make a URI like https://www.germany.travel/entity/Neuschwanstein_Castle "cooler" than the other option for a resource in the knowledge graph.

Throughout the document, we will introduce different URI schemes that contain placeholders for certain parts of the URI. This is called URI templating, that uses a standard syntax[19] for generating URIs with variables. The parts that need to be dynamically created are defined between curly brackets (e.g. https://germany.travel/{domain}/{identifier}). More examples can be found in Section 5 and 6.

[13] https://tools.ietf.org/html/rfc1736

[14] https://tools.ietf.org/html/rfc1737

[15] Mainly as a result of semantic web and linked data research efforts

[16] Possibly through redirection to the URL of the web page in correct language

[17] A term coined by Sir Tim Berners-Lee https://www.w3.org/Provider/Style/URI

[18] Admittedly, a subjective metric.

[19] https://tools.ietf.org/html/rfc6570

Although HTTP URIs are very common ways to identify resources in knowledge graphs, they are by far not the only option. There are other schemes proposed to identify so-called “non-document” resources and even methods that do not involve URIs at all, instead use a set of property values as identifiers. We will go into more detail on these alternative approaches in Section 10.

3 Identifying URI Sources

We separate the search process for URIs into two phases. First we separately search for proper URI Sources for four domains Accommodation, Event, POI, and Tour. In a second phase we select an URI out of the chosen URI source for a given data item. In this section, we focus on the first phase, the identification of proper URI sources for our domains. We list various sources and their data / access formats, as well as their licenses.

3.1 Accommodations

In general, there are many sources where you can find URIs for accommodations. There are many small providers for private rooms/flats/houses. In addition, there is a lot of providers for camping grounds, which also offer huts. It is nearly impossible to include all providers to cover also every private landlord. Therefore, it is important to select URI sources that fulfill as many criteria as possible (one criterion, for example, is coverage). The sources will be ranked based on the criteria. In the following, you can find a selection of URI sources that provide URIs for accommodations. Table 3.1 lists the biggest accommodation service aggregators (e.g. Airbnb, Booking.com) and a few more generic sources (e.g. Wikidata, DBpedia) that can provide URIs for instances in the Accommodation domain.

Source	**Description**	**Data/Access Format**	**License**	**Example**
Airbnb[20]	An online marketplace for arranging or offering lodging, primarily homestays or tourism experiences. Acts as broker.	API[21]	Proprietary	https://www.airbnb.de/rooms/23091571
Booking.com[22]	An accommodation service aggregator website and travel metasearch engine for lodging reservations. Website has 28,425,726 listings in 148,470 destinations in more than 200 countries.	API	No fee, Limited access	https://www.booking.com/hotel/de/hilton-garden-inn-frankfurt-airport-frankfurt-am-main.de.html
DBpedia[23]	A Knowledge Graph that transforms Wikipedia articles for the semantic web. Large hotels can be found in there. Smaller accommodations are not listed.	RDF, SPARQL	Creative Commons Attribution License (CC-BY)	http://dbpedia.org/page/Hotel_Adlon

[20] https://airbnb.de
[21] Not public, only for providers
[22] https://booking.com
[23] https://dbpedia.org

Facebook Places Graph[24]	A geo service for places.	API	No fee, Limited access	Cafe Wha ? Id: 500393245803
Google Knowledge Graph[25]	The Knowledge Graph built by Google based on its search results. Information are mainly based on Wikidata enriched by additional links or related searches. Google uses public available and their own data for the KG.	API	No fee, Limited access	Hilton Frankfurt Hotel kg:/g/11fy23zmb7
Inside Airbnb[26]	An independent, non-commercial set of tools and datasets	CSV	Creative Commons Zero (CC0) License	Links to Airbnb. It just collects Airbnb data
LinkedGeoData [27]	A dataset that uses information collected by OpenStreetMap. Interlinks this data with other knowledge bases in the Linking Open Data initiative. Size: 3,000,000,000 triples. Links to DBpedia: 53,024 triples	RDF, SPARQL	Creative Commons Attribution License (CC-BY)	B.Liebig Café-Bar http://linkedgeodata.org/page/triplify/node264695865
Outdooractive [28]	The biggest outdoor platform in europe. Focused on all kind of tours. Additionally, they provide accommodations that are close to the tours.	API	No fee, Limited access	https://www.outdooractive.com/de/accommodation/mecklenburgische-ostseekueste/hotel-schloss-gamehl/11523576/
Wikidata	An open Knowledge Graph. Central storage for structured data of its Wikimedia sister projects.	JSON, RDF, SPARQL	Creative Commons Zero (CC0) License	Hotel Adlon: https://www.wikidata.org/wiki/Q698095

[24] https://developers.facebook.com/docs/places
[25] https://developers.google.com/knowledge-graph
[26] http://insideairbnb.com/index.html
[27] http://linkedgeodata.org
[28] https://www.outdooractive.com/de/

Wimdu[29]	The biggest European portal for private accommodations. More than 300.000 accommodations in over 150 countries	Crawling/Scraping	Proprietary	https://www.wimdu.de/search/5460aed009800?id=125657e99d43ce3ce7a2cb97978d0331%40e%40a

Table 3.1. URI sources for the Accommodation domain.

The listed URI sources are a good point to start. Probably a better source of URIs is the websites of the accommodations themselves. Nearly all hotels have their own website that can be used to crawl the necessary information. The URL of the website can then be used to identify the hotel. As soon as the hotels are only visible on aggregation platforms like booking.com or Airbnb or others you need to rely again on the URIs you get from them. In general, this works only for hotels respectively commercial accommodation offers. All the private offers will go over aggregators or portals like wimdu.de.

3.2 Events

In general, there are not many authoritative sources that can provide persistent URIs for individual event instances. We see two options[30]:

- Use an event website to find a URI for an individual event.
- Find the original sources of events (e.g. from the public administration websites). For this we can use Wikidata/DBpedia for the list of municipalities.

Below in table 3.2 you can find sources where we show URIs for events.

Source	**Description**	**Data/Access Format**	**License**	**Example**
DBpedia[31]	See Wikidata. The external links in DBpedia can be useful (e.g., they usually contain links to the official website of an event).	RDF, SPARQL	Creative Commons Attribution License (CC-BY)	Oktoberfest[32]: http://dbpedia.org/resource/Oktoberfest

[29] https://wimdu.de

[30] This goes a bit into the process model already

[31] https://dbpedia.org

[32] Original: https://www.bayern.by/erlebnisse/stadt-land-kultur/feste/volksfeste/muenchner-oktoberfest/

Eventful[33]	A portal that provides various types of events e.g. Kölner Lichter 2020.	API	No fee, Limited access	Kölner Lichter[34]: https://eventful.com/cologne/events/klner-lichter-2020-/E0-001-130396281-6 a specific instance of the event.
EventKG[35]	A knowledge graph of events extracted from existing KGs and some semi-structured sources.	RDF, SPARQL	MIT License	Biathlon auf Schalke[36]: http://eventkginterface.l3s.uni-hannover.de/resource/event_26205
Evensi[37]	An event aggregator.	API	No fee, Limited access	Biathlon Auf Schalke[38]: https://www.evensi.de/biathlon-auf-schalke-2019-arenaring-1/343200784
Facebook Events[39]	A dataset that provides events created on Facebook.	API	No Fee, Limited access[40]	One World Party Bonn[41]: https://www.facebook.com/events/362026011071187/

[33] https://eventful.com

[34] Original: https://www.nrw-tourismus.de/koelner-lichter

[35] https://github.com/sgottsch/eventkg

[36] Original: https://www.nrw-tourismus.de/biathlon-auf-schalke

[37] https://evensi.com

[38] Original: https://www.nrw-tourismus.de/biathlon-auf-schalke

[39] https://developers.facebook.com/docs/graph-api/reference/event/

[40] You can only access to the events that a certain user or page RSVPed or created

[41]https://www.bonn.de/veranstaltungskalender/veranstaltungen/hauptkalender/extern/ONE_WORLD_PARTY_BONN.php

Gemeinsame Normdatei[42]	An internationally curated catalog for entities like people, publications, geonames and events.	MARC21, MARC21-XML, RDF	Creative Commons Zero (CC0) License[43]	Oktoberfest[44]: http://d-nb.info/gnd/1213980-4
Google Knowledge Graph[45]	A Knowledge Graph that provides the events collected by Google from the web.	API	No fee, Limited access	Biathlon auf Schalke[46]: kg:/g/11g_wt5ndl
Wikidata[47]	The Knowledge Graph typically provides URIs for repeating significant events and their historical instances.	JSON, RDF, SPARQL	Creative Commons Zero (CC0) License	Kölner Lichter[48]: https://www.wikidata.org/wiki/Q1795850

Table 3.2. URI Sources for the Event domain.

3.3 Point Of Interests

A Point of Interest (POI) can be any "thing", since it (subjectively) depends on the interests of a user and if it is useful for her purposes. POIs go from a house (as a POI for a specific user) to a point on Mars (a generic/public POI), from touristic places to logistic decisions. Basically, POIs are a part of our daily lives around every location. A definition for a POI can be "Locations that exhibit a certain interest or serve a given purpose" [Athanasiou et al., 2019]. Below you will find a list of URI sources for POIs.

Source	**Description**	**Data/Access Format**	**License**	**Example**
archINFORM[49]	An international architecture database provides data, for example, 80000 buildings and locations.	Crawling/Scraping	Proprietary [50]	Garrel https://www.archinform.net/ort/15489.htm
BabelNet[51]	An encyclopedic dictionary contains around 6113467	SPARQL	BabelNet Non-commercial License[52]	FC Bayern Munich II

[42] https://portal.dnb.de
[43] Requires log-in to access all data
[44] Original: https://www.bayern.by/erlebnisse/stadt-land-kultur/feste/volksfeste/muenchner-oktoberfest/
[45] https://developers.google.com/knowledge-graph
[46] Original: https://www.nrw-tourismus.de/biathlon-auf-schalke
[47] https://www.wikidata.org/
[48] Original: https://www.nrw-tourismus.de/koelner-lichter
[49] https://www.archinform.net
[50] The license must be requested by governmental institution
[51] https://babelnet.org
[52] https://babelnet.org/license

	concepts. For instance, places.			https://babelnet.org/synset?word=bn:03878226n
Bavarian Palaces[53]	The Bavarian Administration of Palaces, Gardens and Lakes is the largest public authorities responsible for museums in Germany.	Crawling/Scraping	Proprietary	Hofgarten Ansbach https://www.schloesser.bayern.de/deutsch/garten/objekte/ansbach.htm
Dbpedia[54]	DBpedia offers extracted POI information from Wikipedia.	SPARQL	Creative Commons Attribution License (CC-BY)	Berlin http://dbpedia.org/page/Berlin
Facebook Places Graph[55]	A dataset that provides POIs registered on Facebook.	API	No Fee, Limited access	Hamburg https://www.facebook.com/pages/-/114829128532877
Fast Linked Data[56]	A dataset based on the Library of Congress Subject Headings (LCSH) contains around 1700000 authority records of corporate, geographic names, events.	API, N3, CSV, XML	Open	Hamburg http://id.worldcat.org/fast/1919615/
Flickr Places[57]	A dataset based on the location tags of images on Flickr.	API	Open (limited for certain types of commercial applications)	Munich https://www.flickr.com/places/3220838
Foursquare[58]	A service that offers geolocation data of POIs.	API	No fee, Limited access	Munich https://foursquare.com/v/m%C3%BCnchen/4c219c6111de20a13aa883ce

[53] https://www.schloesser.bayern.de
[54] https://dbpedia.org
[55] https://developers.facebook.com/docs/places
[56] http://experimental.worldcat.org/fast/
[57] https://www.flickr.com/places/
[58] https://enterprise.foursquare.com/products/places

GeoNames[59]	A geographical database of around 4.8 million of places.	API	No fee, Limited access	Munich https://www.geonames.org/2867714/munich.html
Google Places[60]	A dataset that provides data of routes and places.	API	Proprietary, Limited access	
Here[61]	A service that provides location data of landmarks, addresses and businesses.	API	No Fee, Limited access	Tuna Fong Herbs Here ID: 8408lxx5-a1107952bb1704de116622a1f9aabe75
Historic Hotels[62]	A website that publishes data of historical hotels, e.g., castles, chateaus, palaces, academies. which are also linked by Wikidata.	Crawling/Scraping	Proprietary	Hilton Paris Opera http://www.historichotelsworldwide.com/hotels-resorts/hilton-paris-opera
Gemeinsame Normendatei[63]	It offers access to the catalogue of the German National Library, e.g., Historical places, gardens, parks.	MARC21, MARC21-XML, RDF	Creative Commons Zero (CC0) License	Schloss Nymphenburg (München) http://d-nb.info/gnd/4040583-7
Library of Congress[64]	It offers public access to historical resources.	OAI-PMH	Various licenses per collection	Stiftung Berliner Schloss-Humboldtforum http://id.loc.gov/authorities/names/nb2013007473.html
OpenStreetMap https://www.openstreetmap.or	A service that provides data of roads, trails, places, and more.	API	Open Data Commons Open Database License (ODbL)	Berlin https://www.openstreetmap.org/node/240109189

[59] https://www.geonames.org
[60] https://cloud.google.com/maps-platform/places/
[61] https://developer.here.com/products/geocoding-and-search
[62] https://historichotelsworldwide.com
[63] https://portal.dnb.de
[64] https://loc.gov/

SLIPO[65]	A project that provides tools for integrating, enriching, and sharing POIs data.	JSON, RDF	Various open licenses[66]	POIs of Germany http://download.slipo.eu/results/osm-to-rdf/europe/germany/
TomTom[67]	A service that offers data of POIs.	API	No fee, Limited access	Wannsee route https://www.tomtom.com/en_gb/routes/details/germany/wannsee/
TripAdvisor[68]	A service that offers data of hotels, restaurants, attractions, destination, and more.	API	Propietary	Hilton Paris Opera https://www.tripadvisor.com/g187147-d207718
Wikidata[69]	The Knowledge Graph describes POIs such as landmarks, important buildings and locations.	JSON, RDF, SPARQL	Creative Commons Zero (CC0) License	Berlin https://www.wikidata.org/wiki/Q64
Yelp[70]	A service that provides data of places, events, businesses, stores, and more.	API	No fee, Limited access	Englischer Garten https://www.yelp.de/biz/englischer-garten-m%C3%BCnchen
Wikimapia[71]	It includes different POIs and routes (geo shape of a path is not always mentioned, only geo coordinates of a place). It has links to external sources.	API	Creative Commons Attribution License (CC-BY)	Schloss und Schlosspark Rosenau http://wikimapia.org/11078371/de/Schloss-und-Schlosspark-Rosenau

Table 3.3. URI sources for the POI domain.

[65] http://slipo.eu/
[66] https://github.com/SLIPO-EU?page=1
[67] https://www.tomtom.com
[68] https://tripadvisor.com
[69] https://www.wikidata.or
[70] https://www.yelp.com
[71] https://wikimapia.org/

3.4 Tours

The Tour category describes different types of paths, such as hiking, cycling, running, skiing, etc. There are not so many sources where you can find URIs for different types of tours, which cover all regions of Germany. Mostly it's dominated by Outdooractive. In the following table 3.4, you can find the most common sources and their description.

Source	Description	Data/Access Format	License	Example
Outdooractive [72]	It contains different types of trails, such as hiking, cycling, running and skating, mountaineering, winter activities, water trails and equestrian, and covers around 175000 trails for 15 regions in Germany.	API	No fee, Limited access	Hiking trail "Rundweg Preuntsfelden": https://www.outdooractive.com/en/route/hiking-trail/frankenhoehe/rundweg-preuntsfelden/120817603/
Bergfex[73]	It covers 16 regions with the types Wandern (20166), Bike(8295), Laufen (1277), Winter (1088), Wasser (165), Alpin (93), Sonstige (4).	Data access is not clear; no API were found. But Deutsche Zentrale für Tourismus is their partner.	No fee, but all content of bergfex GmbH websites is copyrighted.	Hiking trail "Der Große Hansjakobweg" : https://www.bergfex.com/sommer/baden-wuerttemberg/touren/wanderung/50946,der-grosse-hansjakobweg/
OpenStreetMap[74]	It has the following route types: bicycle, bus, canoe, evacuation, detour, ferry, fitness_trail, hiking, horse, inline_skates, light_rail, motorboat, mtb (mountain biking), nordic_walking, pipeline, piste, railway, road, running, ski, train, tram.	API	Open Data Commons Open Database License (ODbL)	Hiking route "Rundweg A3 Oberförster-Pfahl-Weg (Hohenlimburg) (3186846)": https://www.openstreetmap.org/relation/3186846#map=8/51.057/8.064
DBpedia[75]	It has two categories: Hiking_trails_in_Germany with 46 hiking paths	RDF, SPARQL	Creative Commons Attribution	Hiking route Klosterweg: https://www.dbped

[72] https://www.outdooractive.com/
[73] https://www.bergfex.de
[74] https://www.openstreetmap.org/
[75] https://dbpedia.org

	(http://dbpedia.org/page/Category:Hiking_trails_in_Germany) and German_tourist_routes with 29 routes (http://dbpedia.org/page/Category:German_tourist_routes.		License (CC-BY)	ia.org/resource/Klosterweg
Wikidata[76]	It has a category "Hiking in Germany"(https://www.wikidata.org/wiki/Q14536731), "Deutschland per Rad" (https://de.wikivoyage.org/wiki/Deutschland_per_Rad.	JSON, RDF, SPARQL	Creative Commons Zero (CC0) License	Hiking trail "Rennsteig": https://www.wikidata.org/wiki/Q139056
Kommot[77]	A route planner for hiking, bike touring, mountain biking, road cycling, and running.	API	No fee, Limited access	Bike touring "Sehr schöne Landschaft":https://www.komoot.com/highlight/1010351
Germany Travel[78]	It focuses on city leisure and recreation destinations, contains hiking and cycling routes, scenic routes, UNESCO routes and etc. Also it includes POIs, events and accommodations. It is a project of Deutsche Zentrale für Tourismus e.V. (DZT).	Crawling/Scraping	No fee, but the content of the website is copyrighted.	Cycling route "Berlin-Usedom Island Cycle Route": https://www.germany.travel/en/leisure-and-recreation/cycling/berlin-usedom-island-cycle-route.html
Bikemap[79]	It covers 743178 cycling routes in Germany.	GPX, KML	A worldwide, non-exclusive and royalty-free exploitation license	"Döhle-Heide-Undeloh-Egestorf" cycling route: https://www.bikemap.net/de/r/5004249/

[76] https://www.wikidata.org
[77] https://www.komoot.com
[78] https://www.germany.travel/
[79] https://www.bikemap.net

Wikiloc[80]	It covers 109030 trails for Germany.	Crawling/Scraping	License by Wikiloc: https://www.wikiloc.com/wikiloc/terms_en.html?v5.1	"Berliner Unterwelten" hiking path: https://www.wikiloc.com/hiking-trails/berliner-unterwelten-underworlds-berlin-germany-2858139
Wikimapia[81]	It includes different POIs and routes (geo shape of a path is not always mentioned, only geo coordinates of a place). It has links to external sources.	API	Creative Commons Attribution License (CC-BY)	Hiking trail "Weg der Besinnung": http://wikimapia.org/14004012/de/Weg-der-Besinnung

Table 3.4. URI sources for the Tour domain.

[80] https://www.wikiloc.com/
[81] https://wikimapia.org/

4 Assessing URI Sources

In the recent years, several data sources have been created. These include large, cross-domain, and openly available data sources like DBpedia, and specific and proprietary data sources like Airbnb. However, we observe that these data sources widely vary on their quality from curated data sources like Wikidata to extracted data sources like EventKG[82]. Therefore, a data consumer needs to face the challenge to define a useful data source for specific tasks. There are a number of studies, which have identified data quality dimension into various categories ([Batini & Scannapieco, 2006], [Färber et al., 2018], [Fensel et al., 2020], [Pipino et al., 2002], [Wang, 1998], [Wang et al., 2001], [Wang & Strong, 1996], [Zaveri et al., 2016]) with the aim of measuring the usefulness of a dataset. For instance, [Batini et al., 2009] describe a comparative analysis of methodologies, strategies, and techniques of data quality dimensions, and [Färber et al., 2018] adopt some of these criteria to compare several data sources such as Freebase, OpenCyc, Wikidata, and YAGO.

We define URI sources as data sources that provide URIs for entities over specific domains (see section 3). In the following, we first describe some preliminaries about formal definitions and the preprocessing of URI sources. Afterwards, we describe several quality dimensions with their corresponding metrics, measurement functions and a value calculation per dimension (i.e. aggregation function). Then, we define a total aggregation function as a weighted sum of the fulfilment degree of the aggregation functions, and finally, we describe a framework for weighting metrics and dimensions.

4.1. Preliminaries

The preliminaries give an overview of formal definitions used throughout the paper and describe the preprocessing of URI sources that involves the mapping of URI sources to a unified schema, namely schema.org, the definition of types for each domain, and the reduction of URI sources to these types.

4.1.1. Formal Definitions

Formal definitions that are used throughout the paper:

- A **Criterion** describes or defines a particular characteristic of a dataset with regard to its quality dimension.
- A **Metric** expresses a criterion in that it can be measured to which degree a certain quality dimension is fulfilled for a URI source.
- A **Measurement Function** is a operationalization of a metric expressed in terms of a function with a value.
- An **Aggregation Functions** calculates the aggregated score from distinct measurement functions.
- The **Total Aggregation Function** calculates the total aggregated score from the values of each aggregation function of each quality dimension.
- **URI Source** (r). We define a URI source as an RDF source, an RDF source r consists of a set of triples $T = \{t_1, \dots t_n\}$, $t_i = (s_i, p_i, o_i) \in T$ with $1 \leq i \leq n$, the set of IRIs[83], denoted U, the set of RDF literals, denoted L, the set of blank nodes, denoted B. Where each (s_i, p_i, o_i) is composed of a subject $s_i \in U \cup B$, a predicate $p_i \in U$, and an object $o_i \in U \cup B \cup L$. Furthermore, U, B, and L are pairwise disjoint.

[82] https://github.com/sgottsch/eventkg
[83] Internationalized Resource Identifiers

- **Value Calculation per Quality Dimension**. Given a URI source r; a set of quality dimension $D = \{d_1, \dots, d_{18}\}$[84], $d_i \in D$ with $1 \leq i \leq 18$; a set of metrics $M = \{m_{i,1}, \dots, m_{18,k_{18}}\}$, for each dimension $d_i \in D$ and a set of weights for metrics $WMd_i = \{\propto_{i,1}, \dots, \propto_{i,k_i}\}$. Each metric $m_{i,j}$ corresponds to the dimension d_i, $d_i = \{m_{i,1}, \dots, m_{i,j_i}\}$, and $m_{i,j}(r) \in [0,1]$ where a value of 0 defines the minimum fulfilment degree of a URI source regarding a metric, a value of 1 the maximum fulfilment degree. Furthermore, each metric $m_{i,j}$ is weighted by $\propto_{i,j}$. The resulted weighted function (or aggregation function) of each quality dimension d_i is calculated from the sum of the product of the value of the metrics multiplied with the corresponding weight.
- $\propto_{i,j}$ is an external weight defined by the DZT organization, such as $\propto_{i,j_i}$ defines a weight of importance for a metric $m_{i,j}$ and $\propto_{i,j} \in [0,1]$ where 0 is the minimum degree of importance regarding a metric and a value 1 is the maximum degree. Furthermore it must hold:

$$\sum_{j=1}^{j_i} \propto_{i,j} = 1 \quad \text{for all } i = 1, \dots, 18.$$

- β_i is an external weight defined by the DZT organization, such as β_i defines a weight of importance for each quality dimension d_i and $\beta_i \in [0,1]$ where 0 is the minimum degree of importance and a value 1 is the maximum degree. Furthermore:

$$\sum_{j=1}^{18} \beta_j = 1$$

- Default **prefixes** for namespaces:

```
@prefix dachkg: < https://ds.sti2.org/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix schema: <http://schema.org/>.
```

4.1.2. Preprocessing of the URI Sources

The preprocessing of URI sources plays a very important role for the evaluation of URI sources, it aims to map from unstructured and semi-structured URI sources to RDF and schema.org, define schema.org types for each domain, and reduce mapped URI sources to defined types.

- **Mapping to schema.org.** The data provided for some of the URI sources are in different formats (e.g.CSV, XML, JSON) or using different schemas (e.g. Wikidata schema). Although most of the sources provide (semi-)structured data, in order to make the evaluation and later URI selection processes simpler, we need to first represent them in a unified format (i.e., JSON-LD) and map to a unified schema (i.e., schema.org). For scalability purposes, we do not map the entire schema of a source to schema.org, but only focus to a subset of it. The first heuristic we use to define this subset is the domains we have. For now, we use one schema.org type per domain and a few properties for each type that can help to identify an entity. The mapping can be done programmatically or declaratively via mapping language like RML [Dimou et al., 2014][85] [86]. The second heuristic is the domain specifications we define per domain. We explain how these unification and reduction process works in the following two bullet points.

[84] We consider 18 quality dimensions.
[85] https://rml.io
[86] https://semantifyit.github.io/RocketRML/

- **Define types for each domain**. The DZT organization provided four domains, which are Accommodation, Event, POI, and Tour. Since, we are mapping the URI sources to schema.org, we selected the following types for them:

DZT Domain	schema.org type
Accommodation	schema:LodgingBusiness
Event	schema:Event
POI	schema:Place
Tour	dachkg:TQyCYm-r5[87]

Table 4.1. Defining types for each domain.

Because the expressivity of schema.org is limited, the DACH-KG[88] working group has extended the vocabulary of schema.org and published the dachkg-schema[89]. Furthermore, for each defined type we also consider the instances of their subtypes, e.g. Place > TouristAttraction.

- **Reduce URI sources to these types**. After the schema.org types (Table 4.1) are selected per domain, we create a domain specification for each of these types. A domain specification gives a model of how a domain should be represented in a semantically structured way, e.g., we need to select a collection of properties and ranges to cover the content of these domains (see Section 5 for more details). At last, we reduce the set of instances in each external source to the selected domains with the help of their respective domain specifications. Table 4.2 shows an instance from the Historic Hotels URI source that is mapped to schema.org with the help of the LodgingBusiness domain specification defined for the Accommodation domain (see also Section 5).

Property	Example of Property value
streetAddress* (Text)	Unter Den Linden 77
addressLocality* (Text)	Berlin
addressRegion? (Text)	
addressCountry* (Text)	Germany
postalCode* (Text)	10117
geo* (GeoCoordinates)	13.380248, 67.3455623
name* (Text)	Hotel Adlon Kempinski
starRating* (Rating)	4/5
vatID? (Text)	

Table 4.2. schema:LodgingBusiness type with mandatorty (*) and optional (?) properties for accommodations domain.

[87] Schema Tourism - dachkg:Trail: https://ds.sti2.org/TQyCYm-r5
[88] https://www.tourismuszukunft.de/2018/11/dach-kg-auf-dem-weg-zum-touristischen-knowledge-graph/
[89] dachkg-schema: https://github.com/STIInnsbruck/dachkg-schema

4.2. Dimension, Metrics and Measurements

We introduce concepts regarding each quality dimension, metrics for quantifying the quality dimensions, measurement functions for each metric, and an aggregation function for each dimension.

4.2.1. Accessibility

Criterion. Accessibility implies that data or part of it must be available, retrievable, and contain a license.

Metrics and Measurement functions:

- *Availability of the URI source*, measured by monitoring the ability of dereferencing URIs over a period of time, e.g. the monitoring process can be done by using the Pingdom[90] tool.

$$m_{1,1}(r) = m_{Availability}(r) = \frac{Number\ of\ successful\ requests}{Number\ of\ all\ requests}$$

- *Provisioning of public SPARQL endpoint*, evaluates whether an official SPARQL endpoint is publicly available. E.g., SPARQL endpoint allow users to perform queries on the URI source.

$$m_{1,2}(r) = m_{SPARQL}(r) = \begin{cases} 1\ if\ SPARQL\ endpoint\ is\ publicly\ available \\ 0\ otherwise \end{cases}$$

- *Retrievable in RDF format*, evaluates whether an RDF export dataset of the URI source is officially available.

$$m_{1,3}(r) = m_{RDF}(r) = \begin{cases} 1\ if\ RDF\ export\ available \\ 0.5\ if\ semi-structured\ data\ available \\ 0\ otherwise \end{cases}$$

- *Support of content negotiation*, evaluates whether content negotiation is supported by the URI source. For instance, the URI source returns a desired content type (e.g. RDF/XML).

$$m_{1,4}(r) = m_{ContNeg}(r) = \begin{cases} 1\ if\ content\ negotiation\ is\ supported \\ 0\ otherwise \end{cases}$$

- *Contains a license*, evaluates whether a URI source contains a license under which the URI source data may be used.

$$m_{1,5}(r) = m_{License}(r) = \begin{cases} 1\ contains\ a\ license \\ 0\ otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_1(r) \in [0,1]$ of a URI source r:

$$d_1(r) = d_{Accessibility}(r) = \sum_{j=1}^{5} m_{1,j}\ \ (r) \cdot \propto_{1,j}$$

4.2.2. Accuracy

Criterion. Accuracy defines the reliability and correctness (syntactically and semantically) of the data.

[90] The Pingdom tool regularly pings websites to check whether the site is accessible to users. https://www. pingdom. com/.

Metrics and Measurement functions:

- *Syntactic validity of RDF documents*, evaluates errors encounter in RDF documents.

$$m_{2,1}(r) = m_{SynRDF}(r) = \frac{Number\ of\ valid\ RDF\ documents}{Number\ of\ RDF\ documents}$$

- *Syntactic validity of property values*, evaluates whether the literal values in object position are syntactically correct, e.g., a property value is syntactically correct if it follows certain regular expressions.

$$m_{2,2}(r) = m_{SynProp}(r) = \frac{|\{o|(s,p,o) \in r \wedge o \in L \wedge synValid(o)\}|}{|\{o|(s,p,o) \in r \wedge o \in L\}|}$$

- *Semantic validity of property values*, evaluates whether the meanings of triples with literal values in object position are semantically correct, e.g., a value is semantically valid if the value is true. Since URI sources can be very large like Wikidata that contains around 65 million of facts [Paulheim, 2016], we propose to evaluate the semantic validity of a subset (data sample) of the URI source against a golden standard.

$$m_{2,3}(r) = m_{SemProp}(r) = \frac{|\{o|(s,p,o) \in r \wedge o \in L \wedge semValid(o)\}|}{|\{o|(s,p,o) \in r \wedge o \in L\}|}$$

Aggregation function. The fulfilment degree $d_2(r) \in [0,1]$ of a URI source r is:

$$d_2(r) = d_{Accuracy}(r) = \sum_{j=1}^{3} m_{2,j}\ \ (r) \,.\, \propto_{2,j}$$

4.2.3. Appropriate Amount

Criterion. Appropriate amount defines the quantity of data that is appropriate for a specific task or use case.

Metric and Measurement function:

- *Appropriate amount,* evaluates the degree to which a URI source contains resources for an specific task, e.g., In section 4.1., we define a domain specification "schema:LodgingBusiness" for Accommodations domain. Then, Schema:LodgingBusiness has some mandatory and optional properties that a URI source must contain. This metric, evaluates the degree to which a subset (data sample) of the URI source contains resources with these properties.

$$m_{3,1}(r) = m_{AppropiateAmount}(r) = \begin{cases} 1 & URI\ source\ provides\ data\ for\ all\ properties \\ 0.5 & URI\ source\ provides\ data\ for\ some\ properties \\ 0 & otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_3(r) \in [0,1]$ of a URI source r is:

$$d_3(r) = d_{ApproriateAmount}(r) = m_{3,1}(r) . \propto_{3,1}$$

4.2.4. Believability

Criterion. Believability or Trustworthiness, defines the degree to which the data is regarded as true, credible, correct, real, trustable, unbiased, and verifiable.

Metrics and Measurement functions:

- *Provenance information*, evaluates the existence of provenance information in the URI source, e.g., using Dublin Core for storing provenance information.

$$m_{4,1}(r) = m_{ProvInfo}(r) = \begin{cases} 1 & if\ provenance\ information\ is\ declared \\ 0 & otherwise \end{cases}$$

- *Trustworthiness*, evaluates how trustable are the data regarding their curation and insertion.

$$m_{4,2}(r) = m_{Trust}(r) = \begin{cases} 1 & data\ is\ inserted\ and\ curate\ manually\ by\ domain\ experts \\ 0.75 & data\ is\ inserted\ and\ curate\ manually\ by\ a\ community \\ 0.5 & data\ is\ inserted\ and\ curate\ semiautomatically \\ & from\ structured\ data\ sources \\ 0.25 & data\ is\ inserted\ and\ curate\ semiautomatically \\ & from\ unstructured\ data\ sources \\ 0 & data\ is\ inserted\ and\ curated\ automatically \\ & from\ unstructured\ data\ sources \end{cases}$$

- *Unknown and empty values of properties*, evaluates whether the values in object position are unknown or empty values, e.g. the birth date of a person is not known.

$$m_{4,3}(r) = m_{KnownProp}(r) = 1 - \frac{|\{o|(s,p,o) \in r \wedge (notKnown(o) \vee null(o))\}|}{|\{o|(s,p,o) \in r\}|}$$

Aggregation function. The fulfilment degree $d_4(r) \in [0,1]$ of a URI source r is:

$$d_4(r) = d_{Believability}(r) = \sum_{j=1}^{3} m_{4,j}\ (r) \cdot \propto_{2,j}$$

4.2.5. Concise Representation

Criterion. Refers to how compactly the data (schema and instance level) are represented.

Metrics and Measurement functions:

- *Avoiding blank nodes*, evaluates the existence of triples that contain blank nodes, for instance, blanks nodes are often considered ambivalent [Färber et al., 2018].

$$m_{5,1}(r) = m_{AvoidBlank}(r) = \frac{|\{(s,p,o) \in r | s \notin B \wedge o \notin B\}|}{|\{(s,p,o) \in r\}|}$$

- *Avoiding RDF Reification*, evaluates the use of RDF reification, e.g. reification is discourage since they are unclear [Bizer et al., 2007].

$$m_{5,2}(r) = m_{AvoidReific}(r) = \begin{cases} 1 & reification\ is\ not\ allowed \\ 0.5 & the\ use\ of\ reification\ is\ justified \\ 0 & otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_5(r) \in [0,1]$ of a URI source r is:

$$d_5(r) = d_{ConciseRepresentation}(r) = \sum_{j=1}^{2} m_{5,j}\ (r) \cdot \propto_{5,j}$$

4.2.6. Consistent Representation

Criterion. Refers to the consistency of values of a URI source in terms of the values do not conflict with each other.

Metrics and Measurement functions:

- *Detection of disjoint inconsistencies of classes* [Hogan et al., 2010], evaluates whether a URI source contains inconsistencies caused by disjoint classes, e.g. checking disjoint classes (like foaf:Organization ⊓ foaf:Person) by defining SPARQL query as constrains.

$$m_{6,1}(r) = m_{InconDisjClass}(r) = 1 - \frac{|inconsistent\ disjoint\ class\ assertions|}{|class\ asertions|}$$

- *Detection of inconsistent inverse functional property values* [Hogan et al., 2010], evaluates whether a URI source contains inconsistencies caused by inverse functional properties, e.g. Given a property[91] p declared as instance of owl:InverseFunctionalProperty (i.e. inverse-functional[92]), then, if two resources share a common value for a property p, reasoning will view these resources as equivalent (i.e. referring to the same resource).

$$m_{6,2}(r) = m_{InconPropValue}(r) = 1 - \frac{|inconsistent\ inverse\ functional\ property\ value\ assertions|}{|property\ value\ asertions|}$$

- *Schema restrictions*, evaluates if the schema is checked while data are inserted, e.g., checking whether the inserted data uese the right data type.

$$m_{6,3}(r) = m_{SchemaRest}(r) = \begin{cases} 1 & if\ schema\ restrictions\ are\ checked \\ 0 & otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_6(r) \in [0,1]$ of a URI source r is:

$$d_6(r) = d_{ConsistentRepresentation}(r) = \sum_{j=1}^{3} m_{6,j}\ (r) \cdot \propto_{6,j}$$

4.2.7. Cost-effectiveness

Criterion. It measures the total cost of collecting a proper amount of accurate data to support the desired usefulness [Wang & Strong, 1996].

Metric and Measurement function:

- *Cost of data*, evaluates the degree to which accurate data is necessary, e.g. calculation of the cost per triple with regards to how crucial the data is for a specific use case[93].

$$m_{7,1}(r) = m_{CostData}(r) = \begin{cases} 1 & free \\ 0.75 & low\ cost \\ 0.50 & medium\ cost \\ 0.25 & high\ cost \\ 0 & otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_7(r) \in [0,1]$ of a URI source r is:

$$d_7(r) = d_{CostEffectiveness}(r) = m_{7,1}(r). \propto_{7,1}$$

4.2.8. Ease of Manipulation

Criterion. Ease of manipulation refers to how easy it is to manipulate and apply the data to different tasks [Pipino et al., 2002].

[91] Inverse-functional property cannot be defined for datatype properties.
[92] owl:InverseFunctionalProperty: https://www.w3.org/TR/owl-ref/#InverseFunctionalProperty-def
[93] The effectiveness of data can be measured after applying the data to a specific task, e.g., analyzing the degree to which data fulfil the desired usefulness.

Metric and Measurement function:

- *Documentation*, evaluates the existence of documentation for manipulating the data, e.g., whether there exists documentation for tasks such as modification, classification, and aggregation.

$$m_{8,1}(r) = m_{Document}(r) = \begin{cases} 1 & a\ good\ documentation\ and\ use\ cases \\ 0.75 & there\ exist\ documentation \\ 0.50 & documentation\ is\ not\ complete \\ 0.25 & documentation\ poorly\ useful \\ 0 & otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_8(r) \in [0,1]$ of a URI source r is:

$$d_8(r) = d_{EaseManipulation}(r) = m_{8,1}(r).\propto_{8,1}$$

4.2.9. Ease of Operation

Criterion. It refers to how easily URI source data can be updated, downloaded, integrated, and customized for a specific task or purpose.

Metrics and Measurement functions:

- *Update data*, evaluates the possibility of updating the data.

$$m_{9,1}(r) = m_{UpdateData}(r) = \begin{cases} 1 & is\ possible\ to\ update\ the\ data \\ 0 & otherwise \end{cases}$$

- *Download data*, evaluates the possibility of downloading the data in open standards, e.g. RDF.

$$m_{9,2}(r) = m_{DownloadData}(r) = \begin{cases} 1 & data\ can\ be\ downloaded\ in\ RDF\ serialization\ format \\ 0.5 & data\ can\ be\ downloaded\ as\ machine\ readable \\ & structured\ data \\ 0 & otherwise \end{cases}$$

- *Integrate data*, evaluates the degree to which data can be integrated. For instance, evaluating whether the data use standard vocabularies like schema.org.

$$m_{9,3}(r) = m_{IntegrateData}(r) = \begin{cases} 1 & uses\ standard\ vocabularies\ for\ that\ particular\ domain \\ 0 & otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_9(r) \in [0,1]$ of a URI source r is:

$$d_9(r) = d_{EaseOperation}(r) = \sum_{j=1}^{3} m_{9,j}\ \ (r).\propto_{9,j}$$

4.2.10. Ease of Understanding

Criterion. Ease of understanding refers to how easy it is for humans to understand the data [Färber et al., 2018].

Metrics and Measurement functions:

- *Self-descriptive URI,* evaluates whether self-describing URIs or generic Identifiers are used to identify resources, e.g., the homepage of Example Inc. (http://www.example.com/), the homepage of Alice (http://www.example.com/people/alice).

$$m_{10,1}(r) = m_{SelfDescrip}(r) = \begin{cases} 1 & self\ descriptive\ URIs\ always\ used \\ 0.5 & self\ descriptive\ URIs\ partly\ used \\ 0 & otherwise \end{cases}$$

- *Various languages*, evaluates the degree to which data are described in more than one language, for instance, property values of rdfs:label or schema:name in different languages.

$$m_{10,2}(r) = m_{VariousLang}(r) = \begin{cases} 1 & labels\ provided\ in\ English\ and\ one\ other\ language \\ 0.5 & labels\ provided\ in\ only\ one\ language \\ 0 & otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_{10}(r) \in [0,1]$ of a URI source r is:

$$d_{10}(r) = d_{EaseOperation}(r) = \sum_{j=1}^{2} m_{10,j}\ \ (r) \,.\, \propto_{10,j}$$

4.2.11. Free-of-error

Criterion. It refers to the degree to which data is correct [Pipino et al., 2002]. The correctness can be measured by counting the number of correct property values.

Metrics and Measurement functions:

- *Correct property values*[94], evaluates the degree to which property values are correct, e.g., total correct property values divided by the total number of property values in a URI source.

$$m_{11,1}(r) = m_{CorrectProp}(r) = \frac{|\{(s,p,o) \in r\ |correctValue(o)\}|}{|\{(s,p,o) \in r\}|}$$

Aggregation function. The fulfilment degree $d_{11}(r) \in [0,1]$ of a URI source r is:

$$d_{11}(r) = d_{FreeOfError}(r) = m_{11,1}(r). \propto_{11,1}$$

4.2.12. Interoperability

Criterion. The level of re-using well-known standards, ontologies, or vocabularies to describe resources.

Metrics and Measurement functions:

- *Available as Linked Open Data*, evaluates the degree to which the URI source is available under Linked Open Data 5 Star[95].

$$m_{12,1}(r) = m_{AvailAsLOD}(r) = \begin{cases} 1 & data\ link\ to\ other\ data\ to\ provide\ context \\ 0.8 & data\ use\ URIS\ to\ describe\ their\ content \\ 0.6 & data\ is\ available\ in\ a\ non\ proprietary\ format \\ 0.4 & data\ is\ available\ as\ structured\ data \\ 0.2 & data\ is\ available\ in\ whatever\ format\ with\ an\ open\ license \\ 0 & otherwise \end{cases}$$

- *Use of standard vocabularies*, evaluates the degree to which data are reusing standard vocabularies. For instance, the use of standard vocabularies like RDF Schema[96].

[94] We propose to evaluate the correctness of property values on a subset (data sample) of the URI source against a golden standard.
[95] Is your Linked Open Data 5 Star? https://www.w3.org/DesignIssues/LinkedData.html#fivestar
[96] RDF Schema https://www.w3.org/TR/rdf-schema/

$$m_{12,2}(r) = m_{UseStandVoc}(r) = \begin{cases} 1 & uses\ standard\ vocabularies\ for\ that\ particular\ domain \\ 0.5 & uses\ standard\ vocabularies\ partly \\ 0 & otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_{12}(r) \in [0,1]$ of a URI source r is:

$$d_{12}(r) = d_{Interoperability}(r) = \sum_{j=1}^{2} m_{12,j}\ \ (r) \,.\, \propto_{12,j}$$

4.2.13. Objectivity

Criterion. It defines how unbiased, objective, and impartial is a URI source. It is measured by domain experts and whether independent sources or providers confirm the information.

Metrics and Measurement functions:

- *Unbiased*, evaluates the degree to which the URI source is unbiased.

$$m_{13,1}(r) = m_{Unbiased}(r) = \begin{cases} 1 & URI\ source\ is\ confirmed\ as\ unbiased \\ 0.5 & URI\ source\ is\ confirmed\ as\ partially\ biased \\ 0 & otherwise \end{cases}$$

- *Provenance information*, evaluates the degree to which the provenance information is declared on statement levels, e.g. as much a data source verifiable is, the more objective it is [Naumann, 2002].

$$m_{13,2}(r) = m_{ProvInfo}(r) = \frac{|\{(s,p,o) \in r\ | hasProvInfo(s,p,o)\}|}{|\{(s,p,o) \in r\}|}$$

Aggregation function. The fulfilment degree $d_{13}(r) \in [0,1]$ of a URI source r is:

$$d_{13}(r) = d_{Objectivity}(r) = \sum_{j=1}^{2} m_{13,j}\ \ (r) \,.\, \propto_{13,j}$$

4.2.14. Relevancy

Criterion. Relevancy defines the level of applicability of the URI source given a specific task. The degree to which data are useful for the task at hand[97] [Wang & Strong, 1996].

Metric and Measurement function:

- *Domain coverage*, evaluates the degree to which a URI source covers property values for a specific domain, e.g., we define a domain specification (DS) "schema:Event" (it has mandatory and optional properties) for Events domain (see section 4.1). Which, needs to be fed by data from the URI source. This metric, evaluates the degree to which the set of instances (in a given domain) of the URI source contains properties w.r.t. a DS.

$$m_{14,1}(r) = m_{DomainCoverage}(r) = \frac{Average\ DS\ Property\ Occurance\ on\ an\ Instance\ in\ r}{|Properties\ of\ DS|}$$

Note that a DS property is a property defined for a type representing a certain domain. If a property is used on an instance with any number of values, it counts as one occurrence. Therefore, the value of the metric ranges in [0,1].

Aggregation function. The fulfilment degree $d_{14}(r) \in [0,1]$ of a URI source r is:

[97] Relevancy dimension is strongly related to appropriate amount dimension.

$$d_{14}(r) = d_{Relevancy}(r) = m_{14,1}(r).\propto_{14,1}$$

4.2.15. Reputation

Criterion. It refers to "the degree to which data is highly regarded in terms of its source or content" [Wang & Strong, 1996]. For instance, it can be measured considering the user's reputation score, via a survey in a community or page ranks.

Metrics and Measurement functions:

- *Reputation information*, evaluates whether exists explicit trust ratings to the URI source made by organizations or page ranks.

$$m_{15,1}(r) = m_{ReputInfo}(r) = \begin{cases} 1 & is\ well\ positioned\ in\ explicit\ trust\ ratings \\ 0.5 & is\ on\ an\ average\ position\ in\ explicit\ trust\ ratings \\ 0 & otherwise \end{cases}$$

- *Reputation information of statements*, evaluates the degree to which the statements contain information of their reputation, e.g., for statements with values from external sources, each statement should contain a qualifier of its reputation.

$$m_{15,2}(r) = m_{ReputStat}(r) = \frac{|\{(s,p,o) \in r\ |hasReput(s,p,o)\}|}{|\{(s,p,o) \in r\}|}$$

Aggregation function. The fulfilment degree $d_{15}(r) \in [0,1]$ of a URI source r is:

$$d_{15}(r) = d_{Objectivity}(r) = \sum_{j=1}^{2} m_{15,j}\ \ (r)\ .\propto_{15,j}$$

4.2.16. Security

Criterion. Security dimension indicates how is the access to data restricted [Wang & Strong, 1996] in order to maintain its integrity and prevent its misuse [Zaveri et al., 2016].

Metrics and Measurement functions:

- *Digital signature*, evaluates the degree to which the URI sources use a digital signature as security, e.g., verify the identity of the publisher of the data.

$$m_{16,1}(r) = m_{DigitalSign}(r) = \begin{cases} 1 & URI\ source\ uses\ a\ digital\ signature \\ 0 & otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_{16}(r) \in [0,1]$ of a URI source r is:

$$d_{16}(r) = d_{Relevancy}(r) = m_{16,1}(r).\propto_{16,1}$$

4.2.17. Timeliness

Criterion. It measures the freshness of data for a specific task.

Metrics and Measurement functions:

- *Frequency of updates*, evaluates how fast the URI source is updated.

$$m_{17,1}(r) = m_{UpdateFreq}(r) = \begin{cases} 1 & very\ fast\ update \\ 0.75 & fast\ update \\ 0.50 & regular\ update \\ 0.25 & seldom\ update \\ 0 & otherwise \end{cases}$$

Notice in order to be operational, the user has to define this function further.

- *Freshness of statements*, evaluates the validity period of statements. e.g., it can be measured in a continuous scale from 0 (outdated) to 1(timely) that is computed by the currency of the data when delivered to the user divided by the volatility (the length of time the data remains valid).

$$m_{17,2}(r) = m_{FreshnessStat}(r) = \begin{cases} 1 & specification\ of\ modification\ dates\ supported\ for\ static\ data\ and\ specification\ of\ validity\ period\ supported\ for\ dynamic\ data^{98} \\ 0.5 & specification\ of\ modification\ dates\ supported\ for\ static\ data \\ 0 & otherwise \end{cases}$$

Aggregation function. The fulfilment degree $d_{17}(r) \in [0,1]$ of a URI source r is:

$$d_{17}(r) = d_{Timeliness}(r) = \sum_{j=1}^{2} m_{17,j}\ (r) \cdot \propto_{17,j}$$

4.2.18. Traceability

Criterion. Traceability refers to the degree to "which data is well documented, verifiable, and easily attributed to a source" [Wang & Strong, 1996]. This dimension is highly related to provenance metadata.

Metrics and Measurement functions:

- *Provenance information*, evaluates the degree to which the provenance information is supported for a URI source, e.g., the property value of a statement contains reference information.

$$m_{18,1}(r) = m_{ProvInfo}(r) = \begin{cases} 1 & provides\ provenance\ information\ on\ statement\ level \\ 0.75 & provides\ provenance\ information\ on\ named\ graph\ level \\ 0.50 & provides\ metadata\ in\ machine\ readable\ format\ about\ the\ URI\ source \\ 0 & otherwise \end{cases}$$

- *Authenticity*, evaluates the degree to which statements offers provenance information, e.g., author, publisher, and contributors.

$$m_{18,2}(r) = m_{Authenticity}(r) = \frac{|\{(s,p,o) \in r\ | hasProvInfo(s,p,o)\}|}{|\{(s,p,o) \in r\}|}$$

Aggregation function. The fulfilment degree $d_{18}(r) \in [0,1]$ of a URI source r is:

$$d_{18}(r) = d_{Traceability}(r) = \sum_{j=1}^{2} m_{18,j}\ (r) \cdot \propto_{18,j}$$

4.3. Total Aggregation Function

Given the URI source r, a set of quality dimension $D = \{d_1, \dots, d_{18}\}$ and a set of weights $WD = \{\beta_1, \dots, \beta_{18}\}$. Every dimension $d_i(r) \epsilon\ [0,1]$ where a value of 0 defines the minimum fulfilment degree

[98] Static data is information which is rarely changing, like addresses of hotels, descriptions of points of interests and alike. Dynamic data is fast changing information, like availabilities and prices." [Fensel et al., 2020]

of a URI source regarding a quality dimension, a value of 1 the maximum fulfilment degree. Furthermore, each dimension d_i is weighted by β_i.

The total fulfilment degree $T(r) \in [0,1]$ of a URI source r is then the weighted normalized sum of the fulfilment degrees of the quality dimensions $d_1, \ldots, d_{18}$:

$$T(r) = \sum_{i=1}^{18} d_i(r) \,.\, \beta_i$$

Where: $0 \leq \beta_i \leq 1$ is an external value defined by DZT organization w.r.t. each dimension (see section 4.1).

4.4. Table of Weights

We provide 36 metrics classified in 18 dimensions, which can be applied to assess URI sources w.r.t. their quality (e.g. the URI sources to evaluate are mentioned in section 3). These metrics and dimensions need to be customized by the DZT organization for each domain. In the following, we provide tables for weighting metrics and dimensions regarding each domain that need to be defined by the DZT organization.

	Weighting $\propto_{i,j_i}$			
Metric	**Accommodations**	**Events**	**POIs**	**Tours**
$m_{1,1} = m_{Availability}$				
$m_{1,2} = m_{SPARQL}$				
$m_{1,3} = m_{RDF}$				
$m_{1,4} = m_{ContNeg}$				
$m_{1,5} = m_{License}$				
Total	**1**	**1**	**1**	**1**
$m_{2,1} = m_{SynRDF}$				
$m_{2,2} = m_{SynProp}$				
$m_{2,3} = m_{SemProp}$				
Total	**1**	**1**	**1**	**1**
$m_{3,1} = m_{AppropAmount}$				
Total	**1**	**1**	**1**	**1**
$m_{4,1} = m_{ProvInfo}$				
$m_{4,2} = m_{Trust}$				
$m_{4,3} = m_{KnownProp}$				
Total	**1**	**1**	**1**	**1**
$m_{5,1} = m_{AvoidBlank}$				
$m_{5,2} = m_{AvoidReific}$				
Total	**1**	**1**	**1**	**1**
$m_{6,1} = m_{InconDisjClass}$				
$m_{6,2} = m_{InconPropValue}$				
$m_{6,3} = m_{SchemaRest}$				
Total	**1**	**1**	**1**	**1**
$m_{7,1} = m_{CostData}$				
Total	**1**	**1**	**1**	**1**
$m_{8,1} = m_{Document}$				
Total	**1**	**1**	**1**	**1**
$m_{9,1} = m_{UpdateData}$				
$m_{9,2} = m_{DownloadData}$				
$m_{9,3} = m_{IntegrateData}$				
Total	**1**	**1**	**1**	**1**
$m_{10,1} = m_{SelfDescrip}$				
$m_{10,2} = m_{VariousLang}$				
Total	**1**	**1**	**1**	**1**
$m_{11,1} = m_{CorrectProp}$				
Total	**1**	**1**	**1**	**1**
$m_{12,1} = m_{AvailAsLOD}$				
$m_{12,2} = m_{UseStandVoc}$				
Total	**1**	**1**	**1**	**1**
$m_{13,1} = m_{Unbiased}$				
$m_{13,2} = m_{ProvInfo}$				
Total	**1**	**1**	**1**	**1**
$m_{14,1} = m_{DomainCoverage}$				

Total	**1**	**1**	**1**	**1**
$m_{15,1} = m_{ReputInfo}$				
$m_{15,2} = m_{ReputStat}$				
Total	**1**	**1**	**1**	**1**
$m_{16,1} = m_{DigitalSign}$				
Total	**1**	**1**	**1**	**1**
$m_{17,1} = m_{UpdateFreq}$				
$m_{17,2} = m_{FreshnessStat}$				
Total	**1**	**1**	**1**	**1**
$m_{18,1} = m_{ProvInfo}$				
$m_{18,2} = m_{Authenticity}$				
Total	**1**	**1**	**1**	**1**

Table 4.3. Framework for weighting metrics per domain.

and $\sum_{j=1}^{k_i} \propto_{i,j} = 1$ for all $i = 1, \dots, 18$.

	Weighting β_i			
Dimensions	**Accommodations**	**Events**	**POIs**	**Tours**
$d_1 = d_{Accessibility}$				
$d_2 = d_{Accuracy}$				
$d_3 = d_{ApproriateAmount}$				
$d_4 = d_{Believability}$				
$d_5 = d_{ConciseRepresentation}$				
$d_6 = d_{ConsistentRepresentation}$				
$d_7 = d_{CostEffectiveness}$				
$d_8 = d_{EaseManipulation}$				
$d_9 = d_{EaseOperation}$				
$d_{10} = d_{EaseOperation}$				
$d_{11} = d_{FreeOfError}$				
$d_{12} = d_{Interoperability}$				
$d_{13} = d_{Objectivity}$				
$d_{14} = d_{Relevancy}$				
$d_{15} = d_{Objectivity}$				
$d_{16} = d_{Relevancy}$				
$d_{17} = d_{Timeliness}$				
$d_{18} = d_{Traceability}$				
Total	**1**	**1**	**1**	**1**

Table 4.4. Framework for weighting dimensions per domain.

and $\sum_{i=1}^{18} \beta_i = 1$

5 Selecting URIs

Selecting URIs is mainly involved with applying a set of similarity measurements to certain set of property values in order to find an external URI for a given instance. In the following, we will first introduce our methodology for selecting URIs. We will continue with a list of similarity metrics that can be measured for the content from different domains. We will then introduce identifying properties for each domain and provide a template for defining the weights for each property in a domain. After we demonstrate the similarity calculation with an example, we explain the post processing step that may be needed for certain external sources and conclude the section with a summary.

5.1 Methodology

The activity can be summarized in the following steps:

1. The URI selection system receives a structured representation of the instance (i.e. annotation) from the CMS plugin (see Section 8).
2. The system selects a URI by following the steps below:
    2.1. Take the highest ranked URI source for the domain of the content.
    2.2. Calculate the similarity between the content and the instances from this source. We calculate the similarity of values for each corresponding property value of an instance provided to the system and the instances in the taken external source.
    2.3. Aggregate the similarity scores for the property values per instance by calculating the weighted average.
    2.4. Check if there is an instance found in this source (i.e. similarity score higher than a predefined threshold). If yes:
        2.4.1.1. If the source provides a proper URI (see Cool URIs in Section 2 and Section 5.5) then return the URI of the found instance. This is mostly the case for the LOD sources.
        2.4.1.2. If the source does not provide a proper URI, apply a post-processing step (Section 5.5) to create a new URI based on the information provided by the URI source.
    2.5. If no, take the next URI source (ordered by quality ranking) and repeat this process starting from Step 2.2.
3. If all the sources are exhausted:
    3.1. Calculate the similarity scores (Step 2.2 and Step 2.3) for the instances identified by the previously constructed URIs (see Section 6).
    3.2. If 3.1 fails, then create new URI as described in Section 6.

Our methodology uses the quality ranking of sources to reduce the search space for URIs, in order to efficiently find an appropriate URI for an instance.

5.2 Similarity Metrics

Below you can find different similarity metrics grouped by the nature of the data being compared. We provide certain similarity (or distance) metrics in the following categories[99]:

[99] A subset of metrics that have implementations in popular programming language or can be efficiently calculated.

- String-based metrics: They are calculated by set-based or edit-distance-based measurements to measure how similar two strings are. They can be used on the values of properties like name or streetAddress.
- Vector Space metrics: They are calculated based on the distance of two vectors in an n-dimensional space. These metrics are useful when the geo coordinates of two places are known.
- Point-set distance metrics: These are calculated by measuring the distance between two sets of points. A typical application is to find out how similar are two sets of geo coordinates that comprise a hiking path or a POI.
- Temporal similarity metrics: These metrics are used to find the temporal relationship between two events.
- Topological similarity metrics: These metrics are used to find out if two shapes on a plane are somehow related. A typical application can be to find out if two geometrical shapes drawn by two sets of coordinates are equal of contain each other.

While some similarity metric measurement values are already in 0,1 range, others need to be normalized (0 is the least similar, 1 is the most similar). A comprehensive list of similarity metrics and relevant tools can be found in [Fensel et al., 2020] and in the user documentation of the LIMES tool[100]. A selection is provided in Table 5.2.1.

Metric	**What it does**	**Normalization needed**
String-based		
Cosine[101]	Calculates the cosine of the angle between the vectoral representations of two strings.	The value is already bound in [0,1].
Exact Match	Compares two input strings. Returns 1 if they match exactly, returns 0 if they do not match exactly. Useful for comparing single normalized property values that uniquely identify two instances	The values is already bound in [0,1].
Overlap	Calculates the ratio between the size of the intersection of two strings and the size of the smaller set.	The values is already bound in [0,1].
Jaro[102]	Calculates the edit distance between two strings based on matching characters and the	The values is already bound in [0,1].

[100] http://dice-group.github.io/LIMES/#/user_manual/configuration_file/defining_link_specifications?id=implemented-measures

[101] https://blog.nishtahir.com/2015/09/19/fuzzy-string-matching-using-cosine-similarity/

[102] https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance#Jaro_Similarity

	transpositions needed on one string to obtain another.	
Jaro-Winkler[103]	This is a variant of Jaro similarity metric that favours prefixes of two strings, based on an empirical study that shows that it is less likely to have spelling mistakes in the beginning of strings	The value is already bound in [0,1]
Levenshtein[104]	Calculates the edit distance between two strings.	It should be normalized. Possible normalization formula: $1 - \frac{d}{\max(\lvert s_1 \rvert, \lvert s_2 \rvert)}$ where d is the Levenshtein distance between s_1 and s_2.
Monge-Elkan [del PilarAngeles & Espino-Gamez, 2015]	A hybrid similarity method that splits a string into tokens and takes the average of the scores of most similar tokens.	It is already bound in [0,1] assumed that an appropriate internal string similarity function to compare tokens is selected (e.g. Jaro-Winkler).
Gestalt Pattern Matching[105]	It calculates the ratio of the number of matching characters between two strings and the total length of the strings.	The value is already bound in [0,1].
Normalized Ngram[106]	Calculates the ratio between matching tokens of two strings with a given length (n) and total amount of unique tokens. (can be also called Jaccard index)	The value is already bound in [0,1].
Vector Space Metrics		
Euclidean Distance[107]	Calculates the distance of two points on an arbitrary dimensional space. Could be especially useful to calculate the distance between two geo coordinates.	It should be normalized. Possible normalization formula: $\frac{1}{1+d}$ where d is the euclidean distance.

[103] https://en.wikipedia.org/wiki/Jaro%E2%80%93Winkler_distance

[104] https://en.wikipedia.org/wiki/Levenshtein_distance

[105] https://en.wikipedia.org/wiki/Gestalt_Pattern_Matching

[106] https://en.wikipedia.org/wiki/N-gram#n-grams_for_approximate_matching

[107] https://en.wikipedia.org/wiki/Euclidean_distance

Manhattan Distance[108]	Calculates the distance based on the sum of absolute difference of two coordinates.	Should be normalized similar to Euclidean distance.
Point-Set Distance Metrics [Sherif & Ngomo, 2018]		
Hausdorff Distance	Maximum of the pairwise minimum distances between two sets of points.	Should be normalized like Euclidean distance. In fact, to calculate pairwise minimum distances, Euclidean distance can be used as a metric.
Mean Distance	The great circle distance between the centroids of two shapes.	Should be normalized like Euclidean distance.
Average Distance	Average of great circle distance between pairwise points in two shapes.	Should be normalized like Euclidean distance.
Surjection Distance	The sum of minimum great circle distances of the surjection of a larger set of points onto a smaller one. This distance may be especially useful to get the similarity when the granularity of the geolocations are different in two different sources.	Normalized like Euclidean distance.
Temporal Similarity Metric		
Concurrent events	Checks if two events happen in the same time range and organized by the same agent.	Returns 1 if the events are concurrent, 0 if they are not. No normalization needed.
Topological Similarity Metrics		
Equal Geo Shapes	Checks if two geometries are equal. The equality implies that their interiors intersect and no part of the interior or boundary of one geometry intersects the exterior of the other.	Returns 1 if the shapes are equal, 0 if they are not. No normalization needed.
Covering Geo Shapes	Checks if one geometry covers another. The coverage implies that no points of the geometry that is covered by another one	Given geometries x and y, covers(x,y) is 1 when x covers y, and 0 when at least one point

[108] https://en.wikipedia.org/wiki/Taxicab_geometry

	lie in the exterior of that geometry.	of y is in the exterior of x. No normalization needed.

Table 5.2.1. Similarity metrics.

5.3 Instance Identifying Properties per Domain

Below we list a set of schema.org properties whose values can be used to identify an instance. These properties will be used to create domain specific patterns for the CMS extension, which generates forms for each domain (which corresponds to a schema.org type) based on those patterns.

5.3.1 Accommodation

Type: schema:LodgingBusiness and its subtypes

Property	Example of Property value
address.streetAddress* (Text)	The Squaire am Flughafen,
address .addressLocality* (Text)	Frankfurt am Main
address .addressRegion? (Text)	Hessen
address .addressCountry* (Text)	Deutschland
address .postalCode* (Text)	60549
geo* (GeoCoordinates)	46.564343, 67.3455623
name* (Text)	Hilton Garden Inn Frankfurt Airport
starRating* (Rating)	3/5
vatID? (Text)	DE12345667

Legend: * mandatory ? optional

Table 5.3.1.1. Accommodation properties.

5.3.2 Event

Type: schema:Event and its subtypes

Property	Example of Property value
endDate* (DateTime)	2020-10-04
name* (Text)	Oktoberfest 2020
location* (Place)	Theresienwiese
organizer* (Person or Organization)	Landeshauptstadt München
startDate* (DateTime)	2020-09-19

url* (URL)	https://oktoberfest.eu

Legend: * mandatory ? optional

Table 5.3.2.1. Event properties.

5.3.3 POI

Type: schema:Place and its subtypes

Property	Example of Property value
address .addressCountry* (Text)	Germany
address .addressLocality* (Text)	Berlin
address .geo* (GeoCoordinates)	52.3733153, 13.5064497
name* (Text)	Berlin Schönefeld Airport
address .postalCode* (Text)	12521
address .streetAddress* (Text)	Schönefeld
url* (URL)	https://www.berlin-airport.de/
address .addressRegion? (Text)	Berlin

Legend: * mandatory ? optional

Table 5.3.3.1. Point Of Interest properties.

5.3.4 Tour

Type: dachkg:Trail and its subtypes[109]

Property	Example of Property value
name* (Text)	Teufelsberg Berlin
dachkg:endLocation* (Place)	Olympiastadion (S-bahn) 52.488573, 13.2616
dachkg:startLocation* (Place)	Grunwald (S-bahn) 52.504552, 13.250166
dachkg:wayPoints* (Place/ItemList[110])	52.504552, 13.250166 52.488573, 13.2616
geo* (GeoCoordinates/GeoShape)	
dachkg:typeOfTrail*	Hiking

[109] dachkg:Trail schema can be found in: https://ds.sti2.org/TQyCYm-r5

[110] We use ItemList property for Trail and we consider the first element as a starting point, and the last element as the end point of a trail.

url* (URL)	https://www.wikiloc.com/hiking-trails/teufelsberg-berlin-26632514
distance? (Number)(Length unit of measure)	5.96 kilometers

Legend: * mandatory ? optional

Table 5.3.4.1. Tour properties.

5.3.5 Summary

In this section, we introduced a set of properties for the types in different domains. These properties or a certain combination of them can be used to calculate the similarity of two instances of a certain type in a certain domain. A limited set of property values is crucial to have an efficient similarity calculation.

5.4 Property Weights for Similarity Measurement

The properties listed below may have different levels of significance when it comes to identifying an instance. Therefore, for each domain, a weight must be assigned to each property. These weights must be determined by DZT and can be configured declaratively during the course of the URI recommendation. The sum of property weights for each domain must be bound in [0,1]. Below in table 5.4.1 you can find a template for the property weights (**ɣ**) that are used to calculate the aggregated similarity of two instances.

Properties per Domain	**ɣ**
Accommodation (schema:LocalBusiness)	
streetAddress* (Text)	
addressLocality* (Text)	
addressRegion? (Text)	
addressCountry* (Text)	
postalCode* (Text)	
geo* (GeoCoordinates)	
name* (Text)	
starRating* (Rating)	
vatID? (Text)	

Weight Total	1.00
Event (schema:Event)	
endDate* (DateTime)	
name* (Text)	
location* (Place)	
organizer* (Person or Organization)	
startDate* (DateTime)	
url* (URL)	
Weight Total	1.00
POI (schema:Place)	
streetAddress* (Text)	
addressLocality* (Text)	
addressRegion? (Text)	
addressCountry* (Text)	
postalCode* (Text)	
address* (PostalAddress)	
geo* (GeoCoordinates)	
name* (Text)	
url* (URL)	
Weight Total	1.00
Tour (dachkg:Trail)	
name* (Text)	
dachkg:endLocation* (Place)	

dachkg:startLocation* (Place)	
dachkg:wayPoints* (Place/ItemList)	
geo* (GeoCoordinates/GeoShape)	
typeOfTour*	
url* (URL)	
distance? (Number)(Length unit of measure)	
Weight Total	1.00

Table 5.3.4.1. Weights for Similarity Measurement.

5.5 Postprocessing

The URI sources listed in Section 3 are very heterogeneous in terms of providing proper URIs (see Section 2 about Cool URIs) for the instances from LMOs. Given an instance submitted by an LMO, we can divide the source into three categories:

1. Sources that provide URIs that identify/locate the instance
2. Sources that provide internal identifiers for the instance
3. Sources that provide URIs for documents that are somehow related to the instance

After the URI selection process, if the selected URI source belongs to the first category, no post-processing is needed, since the URIs already represent the entity, not a form of representation of the entity[111]. These sources are typically the well-established open Knowledge Graphs such as DBpedia and Wikidata.

If the source falls into the second category, then there are two possible ways to obtain canonical URIs from that source. The URI can be generated based on the internal identifiers provided by the URI source. These sources are typically APIs that provide some sort of hierarchical data. A deterministic URI scheme can be applied to the instance collected from these sources during preprocessing. Alternatively, URIs can be generated after an instance is identified to be the most similar to the given instance by LMO. Let us assume the "Kölner Lichter" event instance was submitted to the system. Let us say that the most similar instance has been found in the Eventful API. The internal identifier (e.g. 123456) of the API can be used to generate a URI with the following scheme:

https://germany.travel/{domain}/{source}/{internal_id}

which leads to the following URI:

https://germany.travel/event/eventful/123456

If the source belongs to the third category, the safest choice would be to construct canonical URIs as described in Section 6. This is because such sources are typically websites that contain documents about the instance for which we want to find a URI. These URIs mostly contain information

[111] They at least provide content negotiation

that are likely to change for Search Engine Optimization purposes (e.g. the language of the document, website hierarchy); therefore, they are not suitable for identifying an instance. When a webpage is identified to be a document about the instance through simple Information Extraction techniques (e.g. Named Entity Recognition), we connect the URI of that webpage to the instance identified with the newly constructed URI via a property like foaf:page.

5.6 Example

Below we demonstrate the URI selection process with a small example. Assume that LMO NRW (Nordrhein Westfallen) creates a content about the Kölner Dom POI, and submits the following data through the CMS plugin (see table 5.6.1):

Type: Place

Property	Example of Property value
address.streetAddress* (Text)	Domkloster 4
address.addressLocality* (Text)	Köln
address.addressRegion* (Text)	Nord-Rhein Westfallen
address.addressCountry* (Text)	Deutschland
address.postalCode* (Text)	50667
geo* (GeoCoordinates)	50.9408075,6.9538434
name* (Text)	Kölner Dom
url* (URL)	www.koelner-dom.de

Table 5.6.1. Example property value of an entity.

Let us assume that based on the quality assessment we made, we have obtained the following rankings for POI sources in descending order[112]:

Wikidata: 0.95
SLIPO: 0.92
Dbpedia: 0.89
Yelp: 0.88
TripAdvisor:0.87
TomTom: 0.86
BabelNet: 0.85
OpenStreetMap: 0.84
Google places: 0.83
Fast Linked Data 0.83
Flickr Places: 0.82
FOURSQUARE: 0.81
Here: 0.80

[112] The source rankings are normalized.

Deutsche National Bibliothek: 0.78
Library of Congress: 0.77

Given the properties and pre-configured default similarity metrics, we assume that DZT has given the following weights to the properties:

Property	Similarity Metric	ɣ
address.streetAddress* (Text)	Jaro-Winkler	0.15
address.addressLocality* (Text)	Exact Match	0.1
address.addressRegion* (Text)	Jaro-Winkler	0
address.addressCountry* (Text)	Exact Match	0.1
address.postalCode* (Text)	Exact Match	0
geo* (GeoCoordinates)	Euclidean Distance	0.5
name* (Text)	Jaro-Winkler	0.05
url* (URL)	Exact Match	0.1

Table 5.6.2. Example weights.

Our algorithm takes the top-ranked source, namely Wikidata. Then it measures the property value-wise similarity as described in Section 5.1 by applying the assigned metrics for each property that has a weight greater than 0.

The algorithm finds the Wikidata instance of Kölner Dom (https://wikidata.org/wiki/Q4176) with the following similarity score:

address.streetAddress: 0.97
address.addressLocality: 1
address.addressCountry: 1
geo: 0.99
name: 1
url: 1
The weighted similarity score: 0.97*0.15 + 1*0.1 + 1*0.1 + 0.99*0.5 + 1*0.05 + 1*0.1 = 0.99.

Assuming this is the highest value in the source and it is above the predefined similarity threshold, we return https://wikidata/wiki/Q4176 as the canonical URI for Kölner Dom.

5.7 Summary

In this section, we explained the overall process of selecting URIs for entities submitted by content creators via the CMS plugin. We started with describing our methodology. The URI selecting process is realized mainly by measuring the similarity of an instance provided by the LMO and the instances collected from the external sources. The similarity is calculated on selected property values separately for each property in a domain, and then aggregated in a weighted manner to have a single similarity scores of two instances. The weights (γ values) are determined externally by DZT and can be adjusted at any time. In some cases, the URI from a source cannot be used directly. This can happen because the source does not provide a URI but only an internal identifier (e.g. an Event API) or the URI identifies a document (e.g. HTML document) about an entity but not the entity itself. For such cases, the URIs are constructed based on the information provided by the external source. The next section describes several generic procedures for constructing URIs when the process described in this section fails to find a proper URI in an external source.

6 Construction of Canonical URIs

In the last section we described the process of selecting URIs for a given entity. In this section we present different approaches to construct canonical URIs for entities we didn't find a URI for. In the previous section (section 5) it is described how to find a URI for a given entity. If this mechanism does not find one, a new URI will be created. The different approaches on how to create a URI based on the given entity will be described in this section. The entity we get from the CMS-Plugin is schema.org conform. We can therefore rely on the schemas of schema.org for the given domains (Accommodation, Event, POI and Tour). As defined in the previous section, every domain has mandatory fields. Those fields can be used for the URI construction.

The IETF defines a standard[113] for URI patterns (see also section 2 for a detailed description). For the creation of canonical URIs three parts of the standard pattern are necessary:

- **scheme**: indicates the type of the URI. In our case the scheme will be *http*.
- **authority**: usually a domain name or server IP. The entities will become part of the german Knowledge Graph, therefore we use *germany.travel* as authority for the constructed URIs.
- **path**: organize the resource being identified in a hierarchical manner. The first part of the path will always contain the domain, afterwards we use either a created number or the information itself in the path.

The activity flow for constructing URIs is based on four steps. First it is necessary to determine the type of entity[114] the URI should be created for. Afterward, based on the type the pattern is selected. Out of the pattern and the given information the canonical URI is created and finally returned. In the following we will present three different approaches to construct canonical URIs.

- The first approach makes use of numbering the new entities sequentially. For every new URI the numbering increases by one.
- The second approach uses the mandatory information and creates a hash value. This hash value is then one part of the URI.
- The final approach uses the mandatory information without hashing it. This means, the information will be put in plain into the path. This approach results in huge maybe also hard readable URIs.

All the approaches will present one pattern for each of the selected domains (Accommodation, Event, POI and Tour).

6.1 Sequential Numbering

A simple approach of creating canonical URIs is by defining a default pattern which ends with a number that is increased with every new URI creation. In that case the URIs are always unique and easy to create. The pattern used in that approach consists of the name of the domain followed by a sequential number (*http://germany.travel/{domain}/{sequential number]/)*. In the following we give the pattern of each domain and an example.

- **Accommodation**: *http://germany.travel/accommodation/{sequentialNumber]>*
  Example: http://germany.travel/accommodation/1061

[113] See the documentation for a detailed description of the pattern https://tools.ietf.org/html/rfc3986.

[114] We assume here and in the following that we only work with one schema.org type per domain.

- **Event**: *http://germany.travel/event/{sequentialNumber]*
  Example: http://germany.travel/event/234
- **POI**: *http://germany.travel/poi/{sequentialNumber]*
  Example: http://germany.travel/poi/35467
- **Tour**: *http://germany.travel/tour/{sequentialNumber]*
  Example: http://germany.travel/tour/1095

For every domain the sequential number starts with one. Creating a URI for one domain will increase the number for that domain only. We create for example one URI for an accommodation, a URI for an event and another URI for an accommodation. The created URIs will be the following:

1. http://germany.travel/accommodation/1
2. http://germany.travel/event/1
3. http://germany.travel/accommodation/2

An even simpler pattern for URIs is one without domain in the URI (e.g. *http://germany.travel/u/{sequential number}*). For every new created URI the sequential number will be increased by one independent of the domain.

6.2 Hashing Property Values

In this approach we use the mandatory information that need to be provided to the CMS-Plugin for each domain. Based on the given information we calculate a hash value using the SHA-1[115] hash function and use this hash value for the construction of the URI. In the previous section we defined the mandatory properties for each domain. In the following we select some of those mandatory properties for the URI construction. Those properties will be concatenated together with the current unix timestamp. Afterward, we calculate a hash value out of the concatenated text and use it for the URI. For each domain we list the *mandatory properties,* example values for those properties (*example*), the concatenated text that will be used for the hash function (*text2hash*) and the final *URI*.

6.2.1 Accommodation

For an *accommodation* the properties *address, geo, name* and starRating are mandatory (see previous section). For this approach we use the properties *address, name* and additionally the unix timestamp, see table 6.2.1.1.

mandatory properties	*address, name, unix timestamp*
Example	*address*: The Squaire am Flughafen, 60549 Frankfurt am Main, Deutschland *name*: Hilton Garden Inn Frankfurt Airport *unix timestamp*: 1574337407 (2019-11-21T11:56:47+00:00)
text2Hash	The Squaire am Flughafen, 60549 Frankfurt am Main, Deutschland, Hilton Garden Inn Frankfurt Airport, 1574337407

[115] For further information on SHA-1 see http://www.hjp.at/(st_a)/doc/rfc/rfc3174.html

Hash	5E11053DC8C0691ED4850339560C5D7EC6E4D215
URI	http://germany.travel/accommodation/5E11053DC8C0691ED4850339560C5D7EC6E4D215

Table 6.2.1.1. Properties used for hashing.

6.2.2 Event

The CMS-Plugin expects for an event the mandatory properties *endDate, name, location, organizer, startDate* and *url*. We base the calculation of the hash value on the properties *endDate, name, location, startDate* and the current unix timestamp.

mandatory properties	*endDate, name, location, startDate, unix timestamp*
Example	*endDate*: 2020-10-04 *name*: Oktoberfest 2020 *location*: Theresienwiese, München *startDate*: 2020-09-19 *unix timestamp*: 1574337407 (2019-11-21T11:56:47+00:00)
text2Hash	2020-10-04, Oktoberfest 2020, Theresienwiese, München, 2020-09-19, 1574337407
Hash	78069ACE2389F833E77FBF8B596B9E47E08D0B88
URI	http://germany.travel/event/78069ACE2389F833E77FBF8B596B9E47E08D0B88

Table 6.2.2.1. Properties used for hashing.

6.2.3 POI

In our URI concept, POIs are compared using the properties *location, name, type* and *url*. For the construction of unique URIs for a POI we use information from the *location* object, especially *addressLocality, postalCode* and *streetAddress.* In addition, the properties *name* with the unix timestamp are used.

mandatory properties	*addressLocality, name, postalCode, streetAddress, unix timestamp*
Example	*addressLocality*: Schönefeld *name*: Berlin Schönefeld Airport *postalCode*: 12521

	streetAddress: Flughafen Schönefeld *unix timestamp*: 1574337407 (2019-11-21T11:56:47+00:00)
text2Hash	Schönefeld, Berlin Schönefeld Airport, 12521, Flughafen Schönefeld, 1574337407
Hash	083A151A3448F35C039AD2F80CA53D931A768EC1
URI	http://germany.travel/poi/083A151A3448F35C039AD2F80CA53D931A768EC1

Table 6.2.3.1. Properties used for hashing.

6.2.4 Tour

The last of the selected domains are tours. For a tour the CMS-Plugin expects the properties *endLocation, name, startLocation, dachkg:typeOfTrail* and *url*. For the construction of a URI we use the properties *endPoint, name, startPoint* and *dachkg:typeOfTrail* .

mandatory properties	*endLocation, name, startLocation, dachkg:typeOfTrail , unix timestamp*
Example	*endLocation*: ● *geo*: 52.488573, 13.2616 *name*: Teufelsberg Berlin *startLocation*: ● *geo*:52.504552, 13.250166 *dachkg:typeOfTrail*: Hiking *unix timestamp*: 1574337407 (2019-11-21T11:56:47+00:00)
text2Hash	52.488573, 13.2616, Teufelsberg Berlin, 52.504552, 13.250166, Hiking, 1574337407
Hash	760126B7CDC5DAFD7D5976C5DC71AAFC2FC72018
URI	http://germany.travel/tour/760126B7CDC5DAFD7D5976C5DC71AAFC2FC72018

Table 6.2.4.1. Properties used for hashing.

6.3 Property Value-based

In this approach, we include the mandatory information that needs to be provided for each domain. Additionally, we attach a current unix timestamp in the URI to avoid that two similar entities may get the same URI. For each domain, we defined a URI pattern, which based on the prioritized order of mandatory properties. The goal of this approach is to include some human-readable information in

the URI to give a possibility to identify a source and information about it. In the sections below, we describe the URI patterns for each domain and give some examples.

6.3.1 Accommodation

The pattern for describing canonical URI for accommodations is given below. We list the mandatory properties in order to their priority for a given domain.

mandatory properties	*name, address, geo, unix timestamp*
Example	*name*: Hilton Garden Inn Frankfurt Airport *address:* The Squaire am Flughafen, 60549 Frankfurt am Main, Deutschland ● *addressCountry:* Deutschland ● *postalCode*: 60549 ● *addressLocality*: Frankfurt am Main ● *streetAddress*: The Squaire am Flughafen 14 *geo:* ● *latitude*: 46.564343 ● *longitude*: 67.3455623 *unix timestamp*: 1574337407 (2019-11-21T11:56:47+00:00)
URI	http://germany.travel/accommodation/hilton-garden-inn-frankfurt-airport/deutschland/60549/frankfurt-am-main/the-squaire-am-flughafen-14/46.564343/ 67.3455623/1574337407

Table 6.3.1.1. Mandatory Properties.

6.3.2 Event

The pattern for describing canonical URI for events is given below. We list the mandatory properties in order to their priority for a given domain.

mandatory properties	*name, location, startDate, endDate, unix timestamp*
Example	*name*: Oktoberfest 2020 *location*: ● *name*:Theresienwiese ● *address*: ○ *addressCountry*: Deutschland ○ *postalCode:* 80336 ○ *addressLocality:* München *startDate*: 2020-09-19 *endDate*: 2020-10-04 *unix timestamp*: 1574337407 (2019-11-21T11:56:47+00:00)

URI	http://germany.travel/event/oktoberfest-2020/theresienwiese/Deutschland/80336/muenchen/2020-09-19/2020-10-04/1574337407

Table 6.3.2.1. Mandatory Properties.

6.3.3 POI

The pattern for describing canonical URI for POIs is given below. We list the mandatory properties in order to their priority for a given domain.

mandatory properties	*name, address, unix timestamp*
Example	*name*: Berlin Schönefeld Airport *address:* ● *addressCountry*: Deutschland ● *postalCode*: 12521 ● *addressLocality*: Schönefeld ● *streetAddress*: Flughafen Schönefeld *geo:* ● *latitude:* 52.3738 ● *longitude:* 13.5190 *unix timestamp*: 1574337407 (2019-11-21T11:56:47+00:00)
URI	http://germany.travel/poi/berlin-schuonefeld-airport/deutschland/12521/schuonefeld/flughafen-schuonefeld/52.3738/13.5190/1574337407

Table 6.3.3.1. Mandatory Properties.

6.3.4 Tour

The pattern for describing canonical URI for tours is given below. We list the mandatory properties in order to their priority for a given domain.

mandatory properties	*type, name, startLocation, endLocation, unix timestamp*
Example	*dachkg:typeOfTrail* : Hiking *name*: Teufelsberg Berlin *startLocation*: ● *name*: Olympiastadion ● *geo*: ○ *latitude*: 52.504552, ○ *longitude*: 13.250166

	endLocation: ● *name:* Grunwald ● geo: ○ *latitude*: 52.488573, ○ *longitude*: 13.2616 *unix timestamp*: 1574337407 (2019-11-21T11:56:47+00:00)
URI	http://germany.travel/tour/hiking/teufelsberg-berlin/Olympiastadion/52.504552/13.250166/Grunwald/52.488573/13.2616/1574337407

Table 6.3.4.1. Mandatory Properties.

6.4 Summary

In this section we presented three different approaches to construct URIs for entities. The first approach is a simple very straightforward one. We use a sequential number for the URI construction independent for each domain. The URIs are easy to generate, very short and good readable (e.g. http://germany.travel/accommodation/1). The second approach uses some of the mandatory properties that must be entered in the CMS-Plugin to search for a URI. For each domain we selected some of the mandatory properties to create a hash value that is used for the URI (e.g. http://germany.travel/accommodation/5E11053DC8C0691ED4850339560C5D7EC6E4D215). The last approach uses the mandatory properties and creates a URI containing those properties. This results in very long URIs, e.g.,

> http://germany.travel/poi/berlin-schuonefeld-airport/deutschland/12521/schuonefeld/flughafen-schoenefeld/52.3738/13.5190/1574337407).

It is up to DZT to select one of those approaches.

Also there are still some other open issues. Imagine this scenario: We could not find a URI for an entity therefore we generated one. Then one of the external sources were updated with a URI that'd normally could have been given for that entity. We always first check the external sources first when a new entity comes from an LMO. Therefore, we would give the newly added URI to the new entity, even though we have already generated a URI for the same entity. We end up with two different URIs for the same entity. This would not happen very often, but still could happen. To avoid this, we could first make a similarity measurement between a submitted entity and already given URIs, then look for a URI in the external sources. However, this would prevent us from choosing a proper external URI. Alternatively we could choose a new external URI for the “new” entity and also backward update the URI given to its original submission. More generalized this raises the question whether the URIs of the Knowledge Graph should periodically checked for more suitable URIs found in updated external sources.

7 Process Model

In this section, we describe the process model for resolving the identity of an entity (typically represented with a blank node or internal URI) by reusing a URI from reliable external sources or deriving a new canonical URI. The process model consists of the following parts:

1. Preprocessing of the URI sources is a process of identifying relevant URI sources for domains, such as Accommodation, Event, POI, and Tour, and mapping them to uniform format and schema.
2. Ranking of URI sources is a process of assigning a rank to the URI sources by applying quality metrics.
3. Selection of URIs is a process of selecting a proper URI by applying similarity metrics or construction of a canonical URI.

In the three sub-sections below we describe these three processes.

7.1 Preprocessing of the URI sources

The preprocessing of URI sources is performed by URI unifier - DZT and shown in Figure 7.1.1. This process is based on the population of URI sources, which are relevant and widely used for the domains (see Section 3), and their preprocessing (see Section 4.1.2). The preprocessing covers mapping URI sources to RDF and schema.org, defining types for each domain (e.g. type schema:LodgingBusiness for Accommodation domain), and reducing URI sources to these types by using domain specifications.

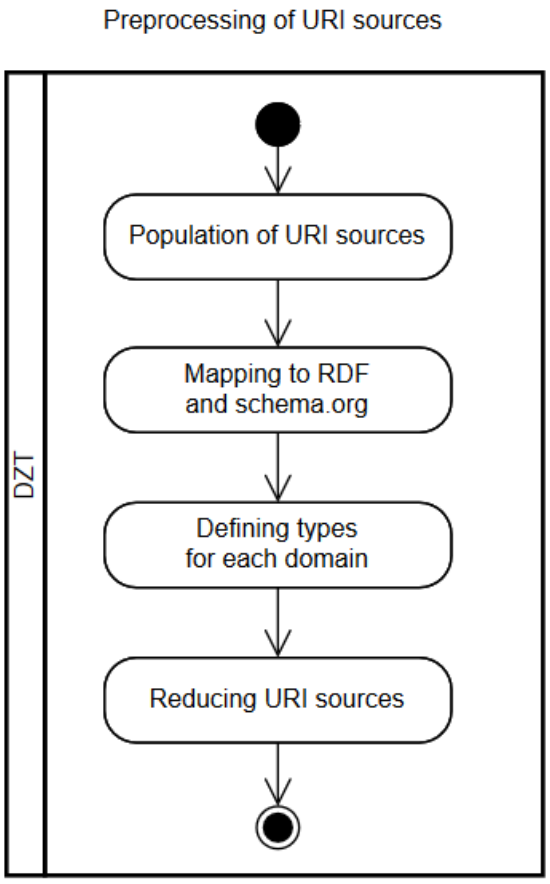


Figure 7.1.1: An activity diagram for the preprocessing of URI sources.

7.2 Ranking of URI sources

To evaluate the obtained URI sources, DZT needs to assign ranks by applying quality metrics (see Figure 7.2.1). In Section 4, we provide 36 metrics classified in 18 dimensions, which can be applied to URI sources. Additionally, for each metric and dimension per domain, DZT needs to provide a list of weights.

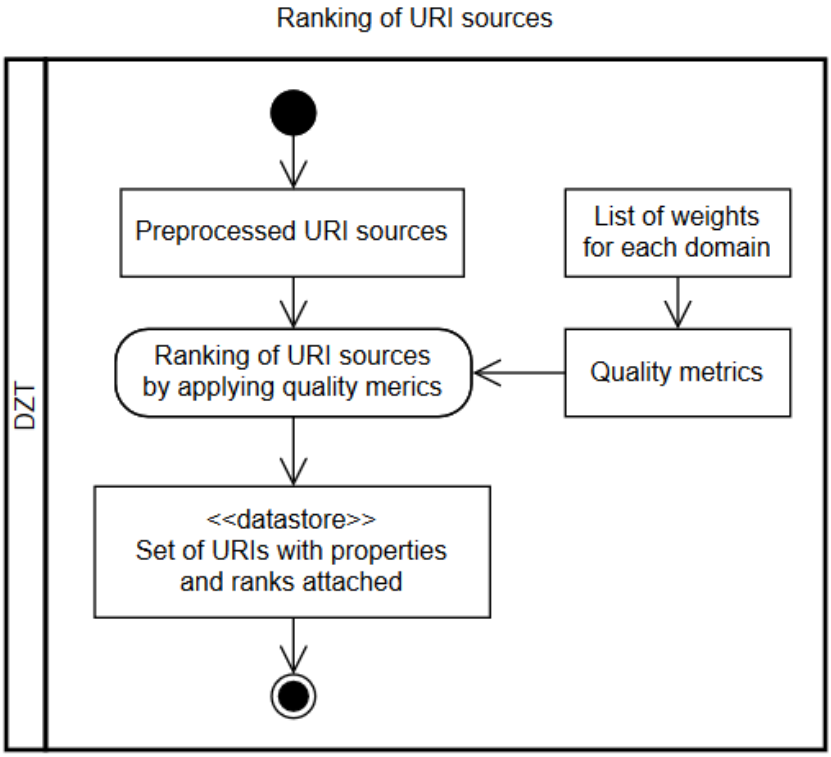


Figure 7.2.1: An activity diagram for the ranking of URI sources.

According to this, the data store with a set of URI sources with properties and ranks attached is built.

7.3 Selection of URIs

The selection and assessment of URIs are performed by content creators – DMO and LMO. As input for a query request, the content creator needs to fill in the CMS plugin form, which generates a schema.org annotation (see Section 8.1). The obtained annotation is used for selecting URIs. The URI Recommender (see Section 8.2) searches for matching entities by applying similarity metrics (see Section 5). It returns a matching URI or constructed URI (see Section 6). As output, a content creator gets the response with a URI for the entity. See Figure 7.3.1.

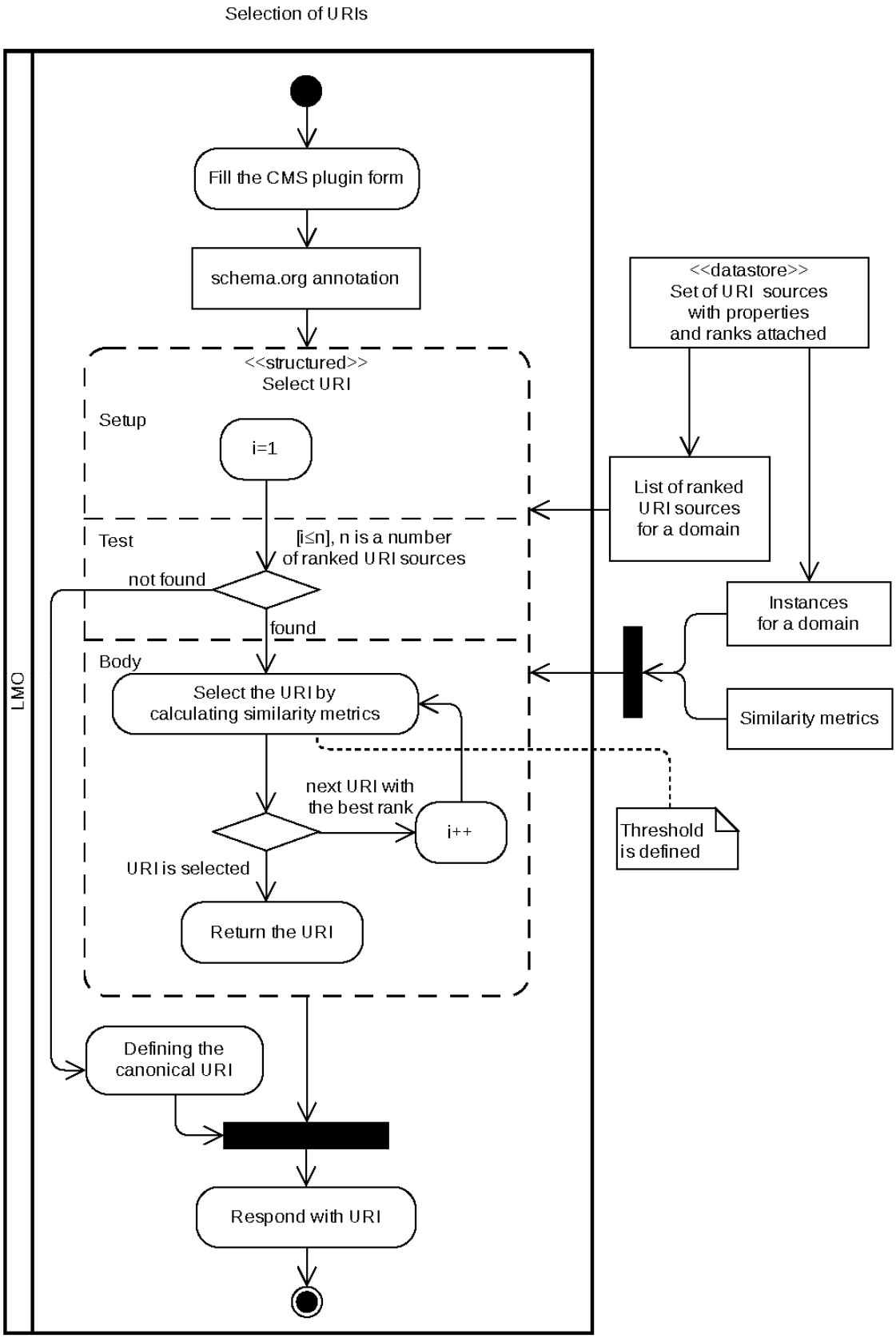


Figure 7.3.1: An activity diagram for the selection of URIs.

8 Implementation

We implement two prototypes to demonstrate our approach. First we give implementation details and user documentation for the CMS plugin that will be used by LMOs to send annotations to the URI Recommender system. Then we describe our URI Recommender system implementation in detaiö.

8.1 CMS Plugin for LMO Input

We prototyped a Wordpress plugin that provides a form for each domain. Below we will describe the implementation details and provide an explanation.

The CMS plug-in is a variation of the Semantify.it Instant Annotation plugin that is implemented with PHP and JQuery for Wordpress. It loads a specific set of domain specific patterns created based on the properties listed in Section 5.3. The plugin generates a form that asks for the values of mandatory and optional properties based on the selected domain specific pattern (e.g. Event). The created annotation is then sent to the URI Recommender API to receive a canonical URI, see Figure 8.1.1. Although the plugin is developed for Wordpress, the core JQuery code is available and allows easy adaption to other CMS.

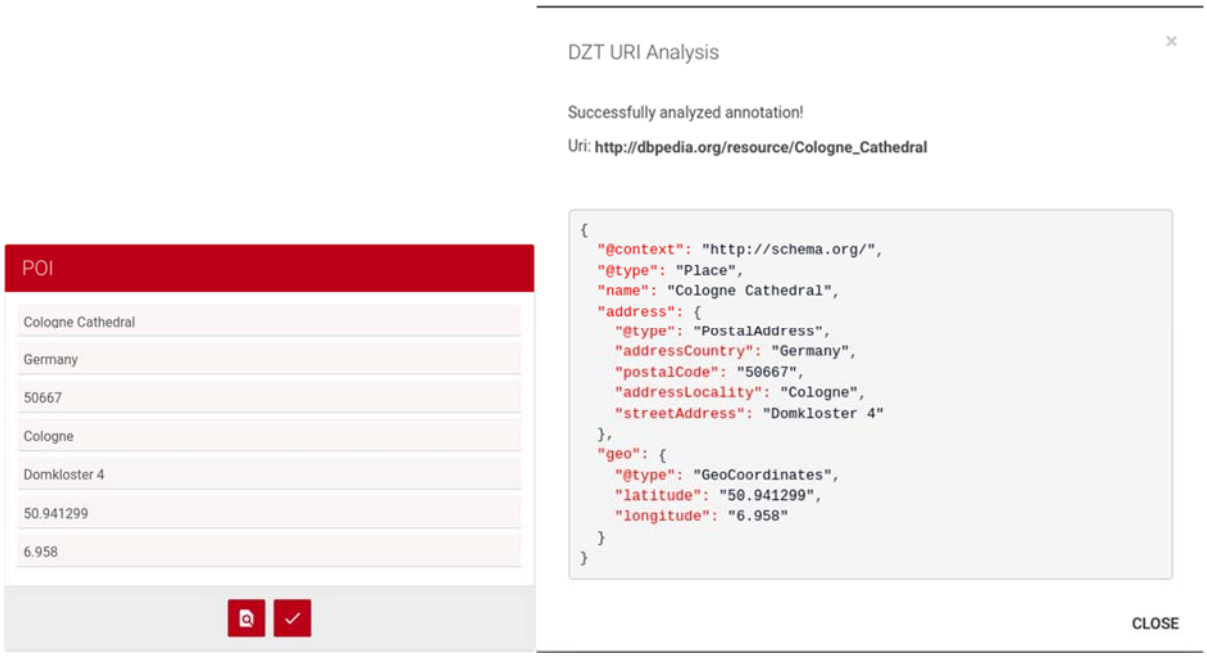


Figure 8.1.1: Content management plugin.

The user of this plugin is a content creator such as LMO. An LMO should follow the following steps to use the plugin to obtain a canonical URI for their content:

1. Upload the plugin files to the /wp-content/plugins/ directory, or install the plugin through the WordPress plugins screen directly.
2. Activate the plugin through the 'Plugins' screen in WordPress. The plug-in will be available on content creation interfaces (e.g. page or post).
3. Use the Settings->DZT URI Plugin screen to enter their API key for the URI Recommender[116].
4. Navigate to Add Post interface and prepare your content.

[116] May be ignored for the prototype

5. Below the text editor, select a domain for your content.
6. Enter the property values into the form.
7. Click on Save to obtain a URI.

The plugin and its source code will be made available for the DZT.

8.2 URI Recommender

The URI Recommender is the implementation of the core concept presented in this paper (see Figure 8.2.1). It implements the approaches described in Section 4,5 and 6. It consists of a URI Source Preprocessor, a URI Source Ranker, URI Selector and URI Constructor. It is a modular NodeJS implementation that provides a RESTful API. The figure below shows the overall architecture of the tool. The implementation currently uses MongoDB as a storage for rapid prototyping, but it can be also adopted towards a Triple store implementation.

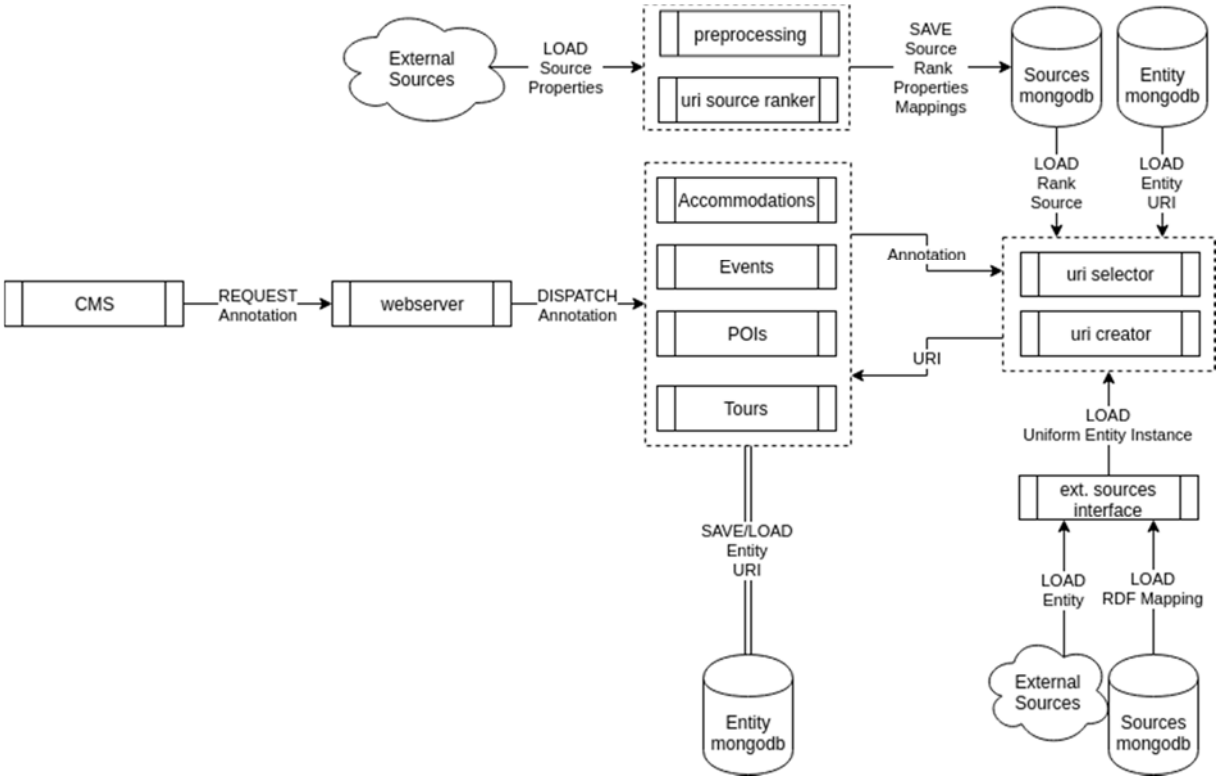


Figure 8.2.1: URI recommender.

In the remainder of this section, we will give implementation details and necessary user documentation for four core modules namely: URI Source Preprocessor, URI Source Ranker, URI Selector and URI Constructor.

8.2.1 URI Source Preprocessor

The URI Source Preprocessor module is responsible for mapping heterogenous external URI sources to a uniform data format (i.e. JSON-LD) and schema (i.e. schema.org) For our prototype, we support the URI sources in Table 8.2.1.

Source	Supported Domains in the Prototype
DBPedia	Event, POI, LodgingBusiness

Wikidata	Event, POI
Google Knowledge Graph	Event, POI
Outdooractive	Tour
Openstreetmap	Tour
Linkedgeodata	LodgingBusiness

Table 8.2.1.1 The sources and their domains used in our implementation

For each data source, we create a source specific SPARQL query or an HTTP request to retrieve the entities with a certain type depending on the domain. The mapping process is currently done programmatically, but for a production-ready system, we recommend a declarative solution such as a mapping language in order to increase modularity and reusability.

The entities are stored in the storage of the URI selector to be used later for similarity calculations. Alternatively, this unification process can be done on-the-fly whenever a canonical URI for an entity is requested by a content creator. On one hand, this has the advantage of getting fresh data, especially from dynamic sources that change frequently. On the other hand, it would have a disadvantage on the response time of the URI selector.

8.2.2 URI Source Ranker

The URI Source Ranker implements the calculation of quality scores for each source per domain as based on the dimensions and metrics described in Section 4. The module processes a JSON file (weights/weights.json) that contains the weights for different dimensions and metrics within each dimension per domain.

In our tool we implemented the source rankings based on the calculation of quality scores of seven selected dimensions for rapid prototyping. These dimensions are namely Accessibility, Accuracy, Appropriate Amount, Believability, Concise Representation, Free-of-error, and Interoperability. These dimensions contain 18 metrics. Some of these metrics require the observation of intrinsic features of the URI sources, some requires the observation of their metadata. For instance the metrics in the Accuracy dimension require running queries or sending HTTP requests in order to check syntactic validity or obtain a sample to check the semantic validity, whereas the license dimension requires the observation of metadata of the URI source (e.g., “About” page of Wikidata).

The helpers/appConfig.js file contains the pre-calculated metric measurements (e.g. if a URI source has a license) and SPARQL queries that help to measure certain metrics. For instance, for Wikidata, the “avoid RDF reification” metric score is automatically assigned to 0.5 for each domain[117], since Wikidata uses a form of reification but its use is justified (e.g., to attach provenance information). For the same URI Source, a SPARQL query is defined to retrieve the instances per domain based on the domain specifications, in order to check whether they follow the syntactic rules of their datatypes.

The ranker provides a RESTful API for other modules. For instance, the URI Selector module that will be described in the next section uses the /data sources endpoint to trigger a ranking and obtain the scores per domain.

[117] Because this metric is domain independent.

8.2.3 URI Selector

URI Selector calculates the similarity between the entity submitted by a content creator and the entities from external sources within the same domain, as well as the entities identified by already constructed URIs. The URI Selector implements the methodology defined in Section 5.

The URI Selector is configured via JSON files per domain that contain property weights for similarity calculation and their respective similarity metrics. For rapid prototyping, we implemented Jaro-Winkler, Levenshtein and Euclidean Distance similarity metrics. An additional configuration should be done for the similarity threshold. This threshold determines whether a URI is acceptable as a canonical URI for an entity. The value of the threshold can be set an ad-hoc value initially, then it can be adjusted based on observations. The URI Selector works with the following workflow:

1. Dispatch incoming request depending on the annotation type (the entity coming from the CMS plugin).
2. Call the URI Selection module for the domain of the entity.
    a. Load rankings of data sources (retrieved from the URI Source Ranker).
    b. For each data source ordered by ranking:
        i. Load candidate entities..
        ii. For each entity its similarity to the submitted entity is calculated.
        iii. If there is a match above the pre-defined threshold:
            1. If the source provides URIs (like DBPedia and Wikidata) use theirs.
            2. Else use the source's internal identifier to create a URI. (see Section 5.5)
        iv. If there are no matches, retrieve the constructed URI (from the URI Constructor).
3. Save the annotation with the respective recommended URI for later lookups.
4. Send response (to the CMS extension) with recommended URI.

The URI Selector provides the following RESTful API endpoints (see Table 8.2.3.1):

Route	**Method**	**Description**	**Parameter type**	**Returns**
/api/entities	POST	New URI recommendation for any supported type.	JSON-LD	URI
/api/pois	POST	New URI recommendation for a POI annotation.	JSON-LD	URI
/api/events	POST	New URI recommendation for a Event annotation.	JSON-LD	URI
/api/tours	POST	New URI recommendation for a Tour annotation.	JSON-LD	URI
/api/accommodations	POST	New URI recommendation for an Accommodation annotation.	JSON-LD	URI
/api/weights	POST	Create new weight configurations for similarity calculation.	JSON	Updated Weight Config
/api/property-metrics	POST	Create new metric setup.	JSON	

/api/similarity-threshold	POST	Create new similarity threshold.	JSON	Old and new Threshold value
/api/datasource-ranks	POST	Trigger an update to data source rankings.	JSON	Source ranks in JSON format

Table 8.2.3.1. The RESTful API endpoints provided by the URI Selector.

8.2.4 URI Constructor

The URI constructor builds and stores canonical URIs for submitted entities, when there are no entities found in the external sources. Our implementation currently supports two alternatives described in Section 6, namely, the sequential number and property-value based URI construction. The URI constructor is automatically triggered as soon as all external sources are exhausted.

8.3 Outlook

In this section we provided a proof of the concept described throughout in this work. We implemented a Wordpress plugin for content creators to submit their entities to the URI recommender that will be hosted by DZT. We provide a Docker container that automatically sets up all services required to run the URI recommender. The Docker container can be simply run with the following command in the project folder: `docker-compose build && docker-compose run`.

The URI recommender requires access to different APIs (e.g., Google Knowledge Graph Search API). DZT should obtain their own keys for the production system.. Our implementation is of prototypical nature. A production-ready implementation should have other considerations like implementing all quality dimensions and more similarity metrics. We will provide additional documentation with the source code including JSON request and response examples, if required.

9 Related Work

In the following we give an overview on related work. First we present some alternative object identification systems. Second we discuss some entity identifier generation systems.

Digital Object Identifier (DOI)[118] was originated in a joint initiative of three trade associations. Originally, it was intended for text publishing, but it conceived as a generic framework for managing the identification of content. The DOI system was announced in 1997 at Frankfurt Book Fair and in the same year the International DOI Foundation (IDF) was created. Unique identification of objects of any type is what the DOI system provides. A DOI name is permanently assigned to an object, linking to current information about that object, including where you can find more information about it or the object itself on the internet. An example of a DOI name is the following: doi:10.1006/jmbi.1998.2354

Another identifying system is the data universal numbering system (D-U-N-S)[119]. The system provides a unique nine-digit identifier for businesses. Furthermore, those identifiers are provided and managed by Dun & Bradstreet[120]. Moreover, the identifiers uniquely identify a company that is managed in the Dun & Bradstreet Data Cloud. Currently more than 330 million global businesses are identified with D-U-N-S. In addition, it enables the identification of relationships between corporate entities. An example of a D-U-N-S name is the following: 341301537

International Standard Book Number (ISBN)[121] is an identifying system used by publishers, booksellers, libraries, internet retailers and other supply chain participants for ordering, listing, sales records and stock control purposes. Up to December 2006 ISBN has a length of up to 10 digits. Since January 2007 it has up to 13 digits. ISBN is calculated using a specific mathematical formula and it consists of five parts (Prefix element, registration group element, registrant element, publication element, check digit). ISBNs are assigned to text-based monographic publication (Books and so on). An example for an ISBN number identifying the book “Knowledge Graphs: Methodology, Tools and Selected Use Cases” is the following: 978-3030374389

The idea of OKKAM [Bouquet et al., 2007] is to enable a Web of Entities. One of the problems of the Semantic Web is the uncontrolled multiplication of identifiers used to refer to the same entity. OKKAM provides a service for the integration of knowledge about entities based on simple identity management. Two basic principles are used by OKKAM. EntityBase, which is an entity centric knowledge base. Any entity is characterized by a collection of attributes whose semantics is defined and known in advance and not based on a schema. Second principle is an EntityProfile. It stores untyped data about entities. In detail, description about entities. OKKAM processes descriptions over an API and decides whether a URI exists for the given description of the entity, or a new URI needs to be created. In addition, to the description of entities OKKAM stores URIs to external sources, which simplifies the integration of different data sources in a later step. OKKAM stores all the descriptions about entities locally. Three different methods are provided to populate OKKAM. An OkkamListsManager provides wrappers that try to map sources (like Wikipedia) into the OKKAM specific language (different fields describing an entity) and import the information. If there exists already an entity with the same labels in the system, the entity is shown to an administrator together with the similar entity. If no similar entity is found, the new entity is added immediately. OkkamDBManager imports entities and their descriptions from generic (mostly relational databases). The mapping to the OKKAM language is very simple for this method. As third method OKKAM provides

[118] https://www.doi.org/doi_handbook/1_Introduction.html
[119] https://www.dnb.com/duns-number.html
[120] https://www.dnb.com/
[121] https://www.isbn-international.org/content/what-isbn

the possibility to manually enter descriptions about entities via web interface. Searching for a URI in OKKAM is done via keyword-based search. Based on the given keywords the system searches for instances where a label matches the keyword. Obviously, a detailed comparison of OKKAM with our approach and a benchmarking of some of the chosen domains would be quite interesting.

TAP [Guha & McCool, 2003] provides a general query interface called GetData. The goal is to reach a semantically unified and coherent Semantic Web. In other words, the Semantic Web should be viewed as one large virtual database. Important for the goals are a way of referring to resources independent of sites and global names for the given entities. To address the issue of globally agreed vocabularies TAP introduces Semantic Negotiation. In principle, it describes a process by which two parties start with a small shared vocabulary and they extend it to a larger shared vocabulary. By exchanging descriptions of resources they want to create understandable references for the same entity. Those descriptions are shared in a formal language and use only vocabulary both parties already share. In principle, the Semantic Negotiation is a back and forth until a common description of the resource can be identified between the two parties. If they come up with a common description, this means that the description identifies the same resource on both sides.

[Gawriljuk et al., 2016] present a scalable approach to incrementally building Knowledge Graphs. The idea came up as they needed a reference data set containing all or most of the artists from the 13 museums and archives of the American Art Collaborative. To achieve the reference data set, it was necessary to consolidate different data sources. When constructing a consolidated set, the challenge is to link the common entities. Linking is very hard due to inconsistencies in data. They started with an initial dataset to create an initial Knowledge Graph. Then they incrementally consolidated further datasets into the Knowledge Graph using a five-step approach. Two of the five steps are *candidate generation* and *linking*, those steps are related to the concept we provided in this paper. By applying MinHash/LSH algorithm they found candidate entities in the Knowledge Graph for each entity in the data source. Afterward only those candidates need to be compared. Similar entities, found by similar MinHash/LSH hashes, are stored in the same cluster. After the candidate generation the similar entities need to be linked. In that step the clusters of candidates are evaluated. Rule-based matching functions are used to decide whether candidate and target entity in the Knowledge Graph are the same. If the candidate fulfills the matching rules, the candidate stays in the cluster, otherwise it is removed from the cluster.

10 Alternative Approaches

In this section, we introduce several alternative approaches to the URI identification and construction concepts we provided.

10.1 URI Normalization

Our approach tries to find URIs from a set of selected sources. Based on the ranking of the given sources a URI will be returned, if found. If there is no URI for the given entity, we create a new one based on the given information. An alternative approach would be the normalization of URIs. This means we don't search for URIs in external sources, but for every new entity we simply create a new URI (basically we restrict our approach to Section 6). Once seen an entity we store it and for upcoming requests we do a similarity comparison with the already existing entities. If we find a matching entity we return the URI we already created. If not, we create a new URI. In this way the search scope is very small and we can efficiently construct and return URIs for given entities. One benefit of normalizing the URIs is the persistent URI scheme (e.g. http://germany.travel/{domain}/{sequentialNumber}). For example, we get information about "Hotel Adlon" and we construct a normalized URI (using approach 6.1) like *http://germany.travel/accommodation/3456*. In addition, the information about the entity are stored in our Knowledge Graph for future requests. The downside of the normalization of URI is that you don't get the knowledge enrichment for free. If you decide at one point to include an external source in your Knowledge Graph, you just have the normalized URIs in your Knowledge Graph. You would need an extensive duplication detection effort to access external knowledge [Fensel et al., 2020].

10.2 Bottom Up URI (Source) Search

The presented URI selection concept for entities uses a pre-selected and ranked set of sources to find URIs. Another alternative for selecting URI sources (Section 3) would be to adopt a "follow-your-nose" approach. This approach is only feasible with Linked (Open) Data sources, but does not require the manual discovery of sources in advance. Instead, a seed source (typically DBpedia, since it is the central node of the LOD cloud) is used as a starting point for an entity. The new source are then discovered by following the sameAs links. After several iterations, for each domain we would collect a set of linked data sources.

Assuming that an LMO submitted an annotation about Kölner Dom. If we start from DBpedia, we would find http://dbpedia.org/resource/Cologne_Cathedral instance. This instance is linked to other sources with sameAs links as follows:

Wikidata: http://www.wikidata.org/entity/Q4176
Freebase: http://rdf.freebase.com/ns/m.0m4nl
Geodata: http://sws.geonames.org/6324464/
Yago: http://yago-knowledge.org/resource/Cologne_Cathedral
Deutsche Nationalbibliothek: http://d-nb.info/gnd/4122597-1

In this example, starting from one source, we discovered five more sources for the POI domain. Note that not all of the URIs are necessarily dereferencable. This means, for some cases, the discovered source may not be usable.

The advantage of this approach is that it dynamically discovers the URI sources, meaning it can potentially reach to a larger set of sources. However, this approach would leave the non-Linked Data sources such as APIs out.

This approach can be extended to find URIs. Like finding URI sources with a follow-your-nose approach, similarity metrics can be applied to the instances that are found by following the same as links. In the example above, we have minimum 6 candidate URIs for Kölner Dom, starting from the DBpedia instance. Similarity metrics can be applied to the submitted instance and the external instances pairwise and the URI of the instance with higher similarity can be selected as the canonical URI for Kölner Dom.

10.3 Identification by Description and Not by URIs

A more radical alternative approach to our proposed concept is to not using URIs at all. Instead, all instances are anonymous (i.e. blank nodes in terms of RDF) and described with the certain property values they contain [Sauermann et. al. (Eds.), 2007]. A typical approach is to benefit from Inverse Functional Properties[122]. These properties uniquely identify an instance. (e.g. two people should not have the same e-mail address). These unique values can be used as an identifier instead of a URI.

In the context of the web, however, there are certain challenges attached to this approach. First of all, it is not always the case that there is one property value that identifies an instance, but a combination of different values[123, 124]. Moreover, since there are no URIs involved, it is virtually impossible to refer and locate an instance on the web. Still Google is using this approach for catching semantic annotations for entities from the web and construct an internal identifier (similar to what we discussed in 10.1).

10.4 Google Search

This approach is based on the crawling information from web sites by using Google search. Google provides search algorithms and ranking systems[125]. The Google ranking systems are designed to sort through hundreds of billions of webpages in the search index to find the most relevant and useful results. Search algorithms look at many factors, including the words of a query, relevance and usability of pages, the expertise of sources, location and account settings. The weight applied to each factor varies depending on the nature of a query. Additionally, external search quality raters are applied to measure the quality of search results.

To prevent inconsistencies in returning different results for the same search query through privatization, localization, etc.), we need to establish browser settings according to an independent user (e.g. use of a private window) and construct a search query by including required information collected from a user (see Sections 5.3.1 – 5.3.4). In the example below, the search results of the query "***The Squaire am Flughafen, Frankfurt am Main, Deutschland, 60549, 46.564343, 67.3455623, Hilton Garden Inn Frankfurt Airport***" are shown in Figure 10.4.1. From these results we choose and respond with the first URI, which comes after advertisement ones, as the most relevant according to Google. From the given example, the respond URI is "***https://www.hiltonhotels.de/deutschland/hilton-garden-inn-frankfurt-airport/***". However, this approach doesn't give a guaranty that selected URI is "cool" (see Section 2) and normalized (see Section 10.1) which however could be added as additional processing.

[122] https://www.w3.org/TR/owl-ref/#InverseFunctionalProperty-def
[123] Analogous to composite primary keys in relational databases.
[124] https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Keys
[125] https://www.google.com/search/howsearchworks/algorithms/

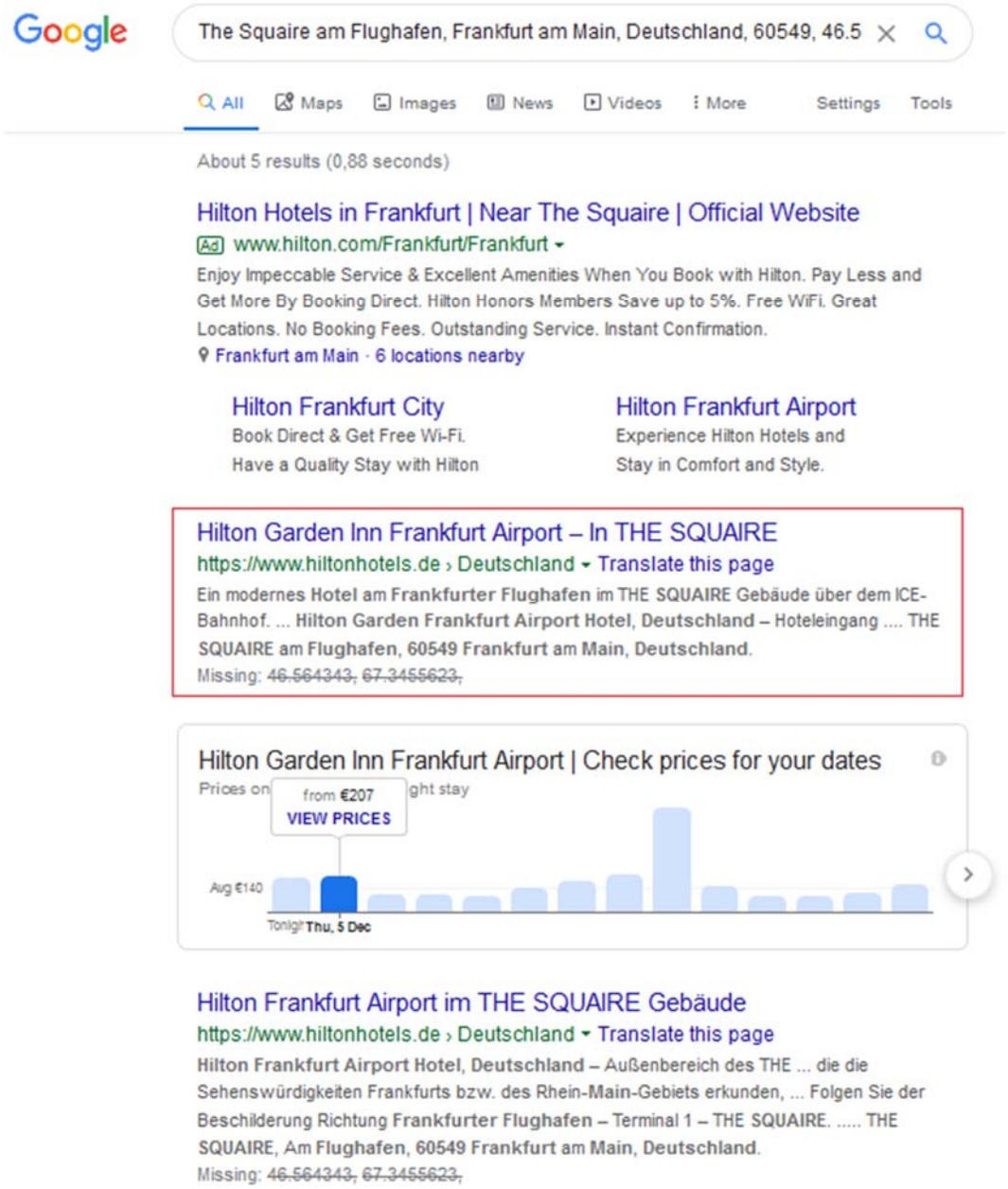


Figure 10.4.1: Google search results.

11 Conclusions

We have presented a methodology and prototypical implementations for assigning identifiers to entities in a tourism knowledge graph in four domains, namely accommodations, events, points of interest (POIs) and tours. Shared identifiers allow for the integration of knowledge from different sources, such as from the different tourism boards in Germany. A straightforward approach would have been to create a stand-alone centralized naming authority, which assigns identifiers.[126]

When thinking in terms of a knowledge graph that will become extended by external data a more complex approach is recommended. Knowledge Enrichment, i.e., the extension of a Knowledge Graph with new knowledge, has the severe issue of duplication detection. Realizing that a new entity is actually just a potentially enriched description of an already existing entity. Therefore, our work has investigated the use of URIs (Uniform Resource Identifiers) from already established web sites to identify entities, so that tourism boards can create and provide data independently of a central system and can use these URIs directly to simplify the duplication detection process (cf. [Fensel et al., 2020]).

The presented approach broadly consists of two steps: first, select web sites that can serve as sources for URIs; and second, give the users in the tourism boards a means to identify URIs as they add data.

- The first step, selection of URI sources, is to be carried out by the central tourism board when the system is being set up. To support the selection of URI sources, we have identified numerous web sites that contain data and stable URIs about instances from each of the four domains. In addition, we have classified the URI sources according to the license under which the data is available and the access methods to the data. The different characteristics of each potential URI source serve as input to a framework for ultimately deciding on the utility of each source.
- The second step, identifying URIs for instances, is to be carried out by the users in each tourism board when they add data to their own knowledge graph. We have presented the process that the tourism board users follow when they want to link to entities in or add data to their regional tourism knowledge graph. The process uses similarity metrics to help decide on URIs from the URI sources for the entities the tourism board users would like to identify. The paper outlines several alternatives for coining URIs in the tourism knowledge graph itself. Finally, we have presented a plugin that illustrates the process of identifying URIs for instances.

The approach outlined in the paper points towards an overall approach for collecting, linking and integrating instance data into a national tourism knowledge graph. Starting with initial data in the regional tourism knowledge graphs, the data available via the URI sources and the instances requested by the tourism board users could be used to integrate and grow a national tourism knowledge graph in an iterative way. Local users could create new instances in case a search request does not yield a result from the regional tourism knowledge graph. If the search request yields results from the URI sources, the new instance data would automatically include a link to an external source that could provide addition information about an entity. Having their local knowledge graphs linked to URI sources then helps with the integration into a national tourism knowledge graph.

[126] Even in this case you have the issue to prevent assigning different internal URIs for the same entity.

12 References

[Athanasiou et al., 2019] S. Athanasiou, G. Giannopoulos, D. Graux, N. Karagiannakis, J. Lehmann, A. Ngonga Ngomo, K. Patroumpas, M. A. Sherif, and D. Skoutas: Big POI data integration with Linked Data technologies. In Proceedings of the 22nd International Conference on Extending Database Technology (EDBT2019), OpenProceedings.org, Lisbon, Portugal, March 26-29, 2019.

[Batini & Scannapieco, 2006] C. Batini and M. Scannapieco: *Data Quality: Concepts, Methodologies and Techniques. Data-Centric Systems and Applications*, Springer, 2006.

[Batini et al., 2009] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino: Methodologies for Data Quality Assessment and Improvement, *ACM Computing Surveys*, 41(3):1–52, 2009.

[Bouquet et al., 2007] P. Bouquet, H. Stoermer, and D. Giacomuzzi: OKKAM: Enabling a Web of Entities. In Proceedings of the Workshop on Identity, Identifiers, Identification, Entity-Centric Approaches to Information and Knowledge Management on the Web (I32007), CEUR Workshop Proceedings, vol. 249, Banff, Canada, May 8, 2007.

[del Pilar Angeles & Espino-Gamez, 2015] M. del Pilar Angeles and A. Espino-Gamez: Comparison of methods Hamming Distance, Jaro, and Monge-Elkan. In Proceedings of the 7th International Conference on Advances in Databases, Knowledge, and Data applications (DBKDA2015), International Academy, Research, and Industry Association (IARIA), Rome, Italy, May 24-29, 2015. https://www.thinkmind.org/download.php?articleid=dbkda_2015_4_20_50058.

[Dimou et al., 2014] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. V. de Walle: RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In Proceedings of the Workshop on Linked Data on the Web (LDOW2014) co-located with the 23rd International World Wide Web Conference (WWW2014), CEUR Workshop Proceedings, vol. 1184, Seoul, Korea, April 8, 2014.

[Dunn, 1946] H. L. Dunn: Record linkage. *American Journal of Public Health and the Nation's Health*, 36(12):1412–1416, 1946.

[Färber et al., 2018] M. Färber, F. Bartscherer, C. Menne, and A. Rettinger: Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO, *Semantic Web Journal*, 9(1):77–129, 2018.

[Fensel et al., 2020] D. Fensel, U. Şimşek, K. Angele, E. Huaman, E. Kärle, O. Panasiuk, I. Toma, J. Umbrich, and A. Wahler: *Knowledge Graphs Methodology, Tools and Selected Use Cases*, Springer, 2020.

[Gawriljuk et al., 2016] G. Gawriljuk, A. Harth, C. Knoblock, and P. Szekely: A scalable approach to incrementally building knowledge graphs. In Proceedings of the 20th International Conference on Theory and Practice of Digital Libraries (TPDL2016): Research and Advanced Technology for Digital Libraries, Springer LNCS 9819, Hannover, Germany, September 5-9, 2016.

[Getoor & Machanavajjhala, 2012] L. Getoor and A. Machanavajjhala: Entity Resolution: Theory, Practice & Open Challenges, *Proceedings of the Very Large Data Bases (VLDB) Endowment*, 5(12):2018–2019, 2012.

[Guha & McCool, 2003] R. Guha and R. McCool: TAP: a Semantic Web platform, *Computer Networks*, 42(5):557-577, 2003.

[Halpin et al., 2010] H. Halpin, P. J. Hayes, J. P. McCusker, D. L. McGuinness, and H. S. Thompson: When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data. In Proceedings of the 9th International Semantic Web Conference (ISWC2010). Springer LNCS 6496, Shanghai, China, November 7-11, 2010.

[Hitzler et al., 2010] P. Hitzler, M. Krötzsch, and S. Rudolph: *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC Press, 2010.

[Hogan et al., 2010] A. Hogan, A. Harth, A. Passant, S. Decker, and A. Polleres: Weaving the Pedantic Web. In Proceedings of the 3rd Workshop on Linked Data on the Web (LDOW2010), CEUR Workshop Proceedings, vol. 628, Raleigh, USA, April 27, 2010.

[Noy et al., 2019] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor: Industry scale Knowledge Graphs: Lessons and Challenges, *ACM Queue*, 17(2):48-75, 2019.

[Pipino et al., 2002] L. Pipino, Y. W. Lee, and R. Y. Wang: Data Quality Assessment, *Communications of the ACM*, 45(4):211–218, 2002.

[Sauermann et al. (Eds.), 2007] L. Sauermann, R. Cyganiak, and M. Völkel (Eds.): Cool URIs for the Semantic Web (Technical Memo TM-07-01), Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, 2007.

[Sherif & Ngomo, 2018] M. A. Sherif and A. N. Ngomo: A systematic survey of point set distance measures for link discovery, *Semantic Web Journal*, 9(5):589-604, 2018.

[Wang & Strong, 1996] R. Y. Wang and D. M. Strong: Beyond Accuracy: What Data Quality Means to Data Consumers, *Journal of Management Information Systems*, 12(4):5–33, 1996.

[Wang et al., 2001] R. Y. Wang, M. Ziad, and Y. W. Lee: *Data Quality*, Kluwer Academic Publisher, 2001.

[Wang, 1998] R. Y. Wang: A Product Perspective on Total Data Quality Management, *Communication of the ACM*, 41(2):58–65, 1998.

[Zaveri et al., 2016] A., A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer: Quality Assessment for Linked Data: A Survey, *Semantic Web Journal*, 7(1):63-93, 2016.