Evaluation Toolkit for API and RDF Alignment

Tobias Zeimetz[1,*], Maurice Büsching[1], Fabian Birringer[1,†], Christoph Otter[1,†], Daniel Zeiler[1,†] and Ralf Schenkel[1]

[1] *Trier University, Universitätsring 15, Trier, 54296, Germany*

Abstract

This paper presents the Evaluation Toolkit for API and RDF Alignment (ETARA), a toolkit to support the development and configuration of appropriate alignment tools. Existing benchmark systems, e.g., MELT, SEALS and HOBBIT, focus on data sets of the same format (e.g., RDF) and do not consider RESTful Web APIs. In contrast, ETARA enables the simulation and creation of RESTful Web APIs including common response formats (XML and JSON). It provides a variety of simulated APIs, which mimic well-known real world APIs, e.g., CrossRef, SciGraph, ArXiv, Semantic Scholar and others. Our contributions are threefold: (1) We present an easy to configure open source alignment toolkit including (2) a gold standard builder that enables a user to create a gold standard alignment for comparison as well as an (3) exemplary analysis of current state-of-the-art API alignment systems (FiLiPo and DORIS).

Keywords

Benchmark System, Alignment Systems, RESTful Web APIs, RDF Databases

1. Introduction

Data aligning describes the process by which relations and classes from two databases are mapped to each other. Many alignment approaches [1, 2, 3, 4] have been developed with different strength and weaknesses which makes it hard to compare them. Nevertheless, to be able to highlight different characteristics and support the development of such systems, various benchmark systems [5, 6, 7] have been developed.

While benchmark systems have been developed for several tasks, e.g. ontology or schema alignment, there are no such systems for the alignment of RDF databases and RESTful Web APIs (denoted simply as APIs from now on). When it comes to data aligning, APIs have become increasingly important over time since they provide a direct connection to the (usually) most recent data of a data provider.

The alignment of RDF databases (DBs) and APIs brings new challenges in addition to the traditional ones. The access of APIs is usually realized by using HTTP GET requests and is restricted in terms of latencies, timeouts between requests and a maximum number of requests (rate limit). HTTP GET requests contain (input) values such as an internal ID or a movie title. If

OM-2023: The 18th International Workshop on Ontology Matching collocated with the 22nd International Semantic Web Conference ISWC-2023 November 7th, 2023, Athens, Greece

*Corresponding author.

†These authors contributed equally.

zeimetz@uni-trier.de (T. Zeimetz); maurice@buesching.eu (M. Büsching); chipshort@tutanota.com (C. Otter); schenkel@uni-trier.de (R. Schenkel)

0000-0002-5436-637X (T. Zeimetz); 0000-0001-5379-5191 (R. Schenkel)

the requested information (e.g., actors of a movie) of the given entity is unknown to the API, usually an HTTP error or a response containing an error message is returned. This confronts alignment systems with the challenge to filter between valid and error responses. Moreover, traditional alignment challenges, i.e. how two different schemes can be aligned, remain as well.

Benchmark systems clearly help during the development of an alignment system. However, current state-of-the-art benchmark systems [5, 6] are not able to cover specific challenges that arise when aligning databases and APIs. Hence, alignment systems use well-known real world APIs for their evaluation. Although this seems like an appropriate approach, it prevents the reproducibility of the results since the data and the schema of APIs will evolve over time.

The main contribution of this paper is the ETARA (**E**valuation **T**oolkit for **A**PI and **R**DF **A**lignment) benchmark system[1]. ETARA was created to simulate RESTful Web APIs and is able cover all important characteristics of APIs, i.e., latency, timeouts, rate limits and provides configurable response structures (e.g. JSON or XML). It was designed to be easy to use, so that existing alignment systems can easily integrate ETARA. It provides a set of databases (of different domains, e.g., bibliographic, filmographic, etc.) and response structures which follow the FAIR[2] design principles. Moreover, it provides tools to create custom response templates or modify provided ones. Additionally, it provides a well-known and established alignment visualization, namely alignment cubes [8], that supports interactive views on different levels of granularity. The target audience are developers and researchers of alignment systems that want to perform reproducible evaluations. Additionally, ETARA provides a component to create gold standard alignments that can be used to evaluate the results of alignment systems.

The rest of this paper is structured as follows: Section 2 discusses existing benchmark systems and visualization techniques that support the development of alignment systems. Section 3 presents a set of use cases that and their core tasks. Section 4 gives an overview of the ETARA benchmark system and its possibilities whereas Section 5 explains how users interact with ETARA and perform a benchmark on existing alignment systems using an example scenario.

2. Related Work

The increasing number of schema/ontology alignment systems demands a uniform evaluation approach. The Ontology Alignment Evaluation Initiative (OAEI) is an international initiative to forge such an approach. The OAEI hosts several tracks targeting alignment issues. Currently, the OAEI uses two systems to compare the results of alignment systems.

One of these systems is MELT [5], a toolkit for ontology alignment development, fine-tuning, packaging, and evaluation. It can be used by developers to analyze the performance and errors of their alignment systems. It is a powerful tool, since it can be integrated into the process of an alignment system and makes a detailed analyses possible.

The second system used by the OAEI is HOBBIT [6]. It provides a GitLab instance which can be used to upload alignment systems in form of Docker images and define the metadata of benchmarks. The evaluation of the provided alignment system is performed on the servers of

[1]https://github.com/ETARA-Benchmark-System
[2]https://www.go-fair.org/fair-principles/

the HOBBIT platform. Moreover, HOBBIT provides a very detailed evaluation possibility and comparison with other systems tested on the HOBBIT platform.

A system that was used by the OAEI until 2020 is SEALS [7]. It is one of the best-known benchmark platforms for link discovery. The output of SEALS is the alignment between two ontologies using the Alignment API [9] format[3].

Moreover, the success of benchmark systems is strongly linked to the usability and how well they are able to visualize the evaluation results. In most cases alignments are measured only in terms of precision, recall, and f-measure. As stated in [8], these measures give a good overall assessment but they do not allow for a more detailed analysis. Hence, it is often difficult to determine the individual strengths and weaknesses of alignment systems relying only on precision and recall.

The state-of-the-art visualization that addresses these problems is called Alignment Cubes [8]. It provides an interactive visual environment for the comparative exploration and evaluation of multiple alignments at different levels of granularity. Additionally, it provides a compact way to visualize multiple alignments, since ontologies tend to become rather large and complex. To prevent information overload the approach supports multiple complementary views.

Other approaches are VOAR [10, 11] or SILK [12, 13]. However, in contrast to Alignment Cube, the visualization of VOAR and SILK tends to clutter, especially with large ontologies, and is only able to visualize the alignment between two ontologies, but not multiple ones.

The benchmark systems and visualizations mentioned above are all specialized on RDF alignments and are therefore not suitable as benchmark systems for alignment systems that focus on RDF databases and Web APIs. Koutraki et al. [14], state that *APIs seem to be a sweet-spot between making data openly accessible and protecting it.* Extending RDF databases with live data collected by APIs, e.g., by integrating missing titles, allows improving the data quality. In contrast to traditional alignment systems, the alignment of RDF databases and APIs brings new challenges.

Since APIs mostly expose their data in the form of JSON responses, traditional RDF alignment approaches are not possible. Instead, API responses (typically in a tree structure like JSON or XML) are interpreted as graphs. The paths are interpreted as relations and the values as literals. Works such as that of Koutraki et al. interpret JSON objects as nodes or entities to align relations with each other. Other work such as that of Zeimetz and Schenkel [1] is based on aligning the full path to a literal, for example the title of a publication, with the relations of an RDF database. To determine similarities between the paths and relations or the JSON values and literals and thus to determine an alignment, string similarity methods are mostly used.

3. Benchmark Requirements and Tasks.

Existing works [15, 16, 17, 18] have identified tasks that support the development of alignment systems. However, they focus only on classic alignment systems between databases of the same format, e.g., ontology alignment systems. None of the presented works focus on tasks needed to support the development of alignment systems between RDF databases and APIs. Hence, we identify and present all necessary tasks in this section.

[3]https://moex.gitlabpages.inria.fr/alignapi/

Simulation of RESTful Web APIs. To support the development of alignment systems that focus on determining an alignment between APIs and RDF databases, a benchmark system needs to be able to simulate APIs including their characteristics and their response structure. Hence, it has to cover all important characteristics of real APIs, i.e., latency, timeouts and more. In addition, users need to be able to control more complex behaviors of APIs, e.g., error and request management.

Regarding the error management, we identified three classes how APIs deal with errors, e.g., if a requested resource is unknown. In most cases APIs return an HTTP status code. In other cases an empty response with some metadata about the request or a response that contains an (http) error message is returned.

Additionally, APIs have different response strategies. Most APIs receive a key value, e.g.,a DOI, ISBN, etc., via an HTTP Get request. The requested data is queried against a database. However, there are APIs that do not search for exact values but return the top similar results. In other words, if information about the movie "The Matrix" is requested, information about all movies that have a similar title (e.g., "The Matrix Reloaded") will be returned.

Customizable Response Templates. APIs differ not only in response time, rate limits, etc., but especially in the structure of the response schema. Some APIs differ in their output formats (e.g., XML or JSON). Additionally, some data providers offer very specialized APIs that return only a single piece of information, such as the title of a publication or the name of an actor, while other APIs provide all available data in their response Furthermore, the data schema itself can be very heterogeneous. For example, some APIs return author names as a single string, while other APIs differ between first and last name.

Hence, a benchmark system must provide a component that allows to create a response template that can be customized. It can be used to mimic the response structure of existing APIs and their behavior as well as to create specific challenging templates for alignment systems.

Gold Standard and Alignment Visualization. To compare and analyze alignments and alignment systems, a suitable visualization and a perfect alignment, denoted as gold standard (GS) alignment, is needed. The creation of a GS is a time-consuming task performed by an expert. Hence, a benchmark system needs to provide suitable GS alignments and, in case new alignments need to be created, a suitable component to supports the creation of GS alignments. Moreover, as described in the work of Ivanova et al. [8], core aspects of benchmark systems cover an interactive visualization and exploration that supports various views on different levels of granularity. Further, the visualization should support the selection, combination and fine tuning of alignment systems. It needs to support the interactive development process of alignment systems and provide information about the strengths and weaknesses of the systems.

Providing Reproducible Evaluations. The target audience of alignment benchmarks are developers and researchers of alignment systems that want to perform reproducible evaluations. Using existing real-world APIs to evaluate alignment systems is not a suitable option, since APIs, their behavior, the data basis can change over time, or APIs may become disconnected. For many existing alignment systems, reproducible results are not possible because used APIs no longer exist or the data has changed significantly. Hence, a benchmark system has to provide a set of databases and a configurable component for simulating APIs, including all their characteristics, e.g., latency, rate limits and response structures. The corresponding evaluation results can be reproduced since configurations can be documented and data sets are publicly available.

4. Toolkit Overview

ETARA is a benchmark system which supports the development of alignment systems. First, we present an overview of the configuration possibilities provided by ETARA to simulate APIs. Afterwards, the Apache FreeMarker template engine will be introduced that is used to create response templates which are dynamically filled with requested data. To evaluate the quality of alignments, we will give an overview of the used alignment cube visualization and, lastly, introduce a GS component, which can be used to manually craft GS alignments. Note that for the sake of shortness many details had to be omit but can be found in the wiki on GitHub[4].

4.1. Requirements and Configuration

To simulate APIs, several requirements needs to be met. First, ETARA needs access to databases that represent the pool of available data for the simulated APIs. Currently, ETARA is only able to process RDF databases, but this aspect will be extended in the future. With a registration of the databases, requested information can be extracted and integrated into response templates.

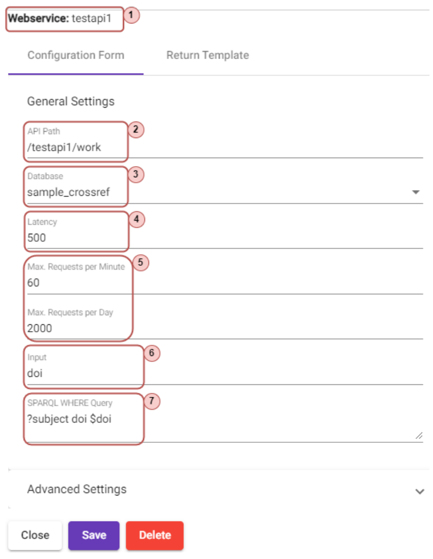
After users have registered their databases, they can be used to simulate APIs. Figure 1a shows an example of all configuration options. After specifying a label for the simulated API (see Figure 1a, marker 1), users must specify the path (URL) which is used to request the API (see marker 2). Next, they select the database to use (see marker 3) and the average response time (see marker 4). Moreover, users can configure the rate limit (see marker 5) which defines the number of requests per minute and day. Since the APIs being simulated are RESTful Web APIs, data is requested via HTTP Get requests. When a client requests a specific resource on a Web server using the HTTP protocol, the client submits certain GET parameters to the server with the requested URL. These parameters are pairs of names and corresponding values. They are appended to the URL with the ? character followed by a parameter name and tell the server which resources are meant. The parameter name to which the simulated API responds can be configured through the interface (see marker 6).

The most complex configuration is to make valid (SPARQL) queries to the database, extract desired information and insert it into the given response template. For this purpose, users can specify one or more triple patterns that are used to identify a subject in the (RDF) database (see Figure 1a, marker 7). In the example shown in Figure 1a, the database `sample_crossref` is searched for a subject that has a relation named `doi` and refers to the value of the parameter named `doi`. The found subject (an entity) is then used to fill the response template, using Apache FreeMarker, with information about the entity (e.g. the title of a publication). A detailed explanation of FreeMarker is given in the next section.

In addition to the classic configuration options such as rate limit or latency, users are able to control more complex behaviors of APIs. The "Search Type" option in Figure 1b specifies the search method to be used. There are two search options to choose from: `precise` and `fuzzy`.

The `precise` parameter sets APIs to only return data that was requested. For example, if an API receives a request containing an arxiv ID, the corresponding publication is searched. If it exists, the found publication entity is forwarded to the response template as described before. However, if no publication with the searched ID exists, no result can be returned.

[4]https://github.com/ETARA-Benchmark-System



Webservice: testapi1

Configuration Form | Return Template

General Settings

API Path: /testapi1/work

Database: sample_crossref

Latency: 500

Max. Requests per Minute: 60

Max. Requests per Day: 2000

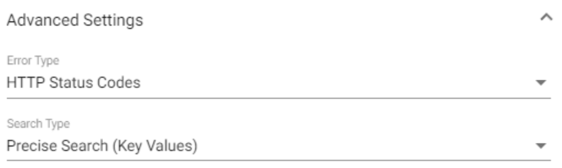
Input: doi

SPARQL WHERE Query: ?subject doi $doi

Advanced Settings

Close | Save | Delete

(a) Basic API Configuration

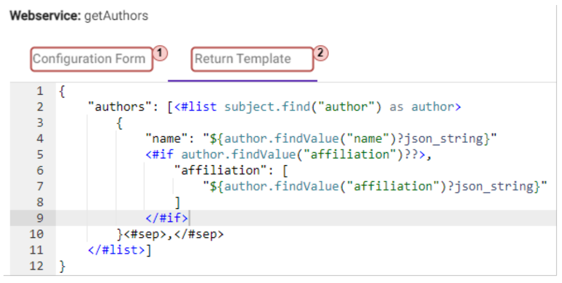


Advanced Settings

Error Type: HTTP Status Codes

Search Type: Precise Search (Key Values)

(b) Advanced Settings



Webservice: getAuthors

Configuration Form | Return Template

```
{
    "authors": [<#list subject.find("author") as author>
        {
            "name": "${author.findValue("name")?json_string}"
            <#if author.findValue("affiliation")??>,
                "affiliation": [
                    "${author.findValue("affiliation")?json_string}"
                ]
            </#if>
        }<#sep>,</#sep>
    </#list>]
}
```

(c) Example Response Template

Figure 1: Configuration of a Simulated RESTful Web API

The fuzzy parameter enables an approximate search. This allows to return search results that have a certain degree of semantic overlap with the query. This option was implemented because some APIs behave like a classic Web search and return the best results for the query. For example, when passing a title instead of an ID, such behavior may be desirable.

Furthermore, users can configure the type of error handling. For this, the "Error Type" option provides three possible parameters: `HTTP Status Codes`, `JSON With Status Codes` and `JSON Without Status Codes`. By selecting `HTTP Status Codes`, the simulated API responds with an HTTP status code including an error description, which provides information about the nature of the error. If the `JSON With Status Codes` parameter is used, the API sends a JSON object with attribute-value pairs describing the error code. Additionally, an HTTP status code is appended to the response. By `JSON Without Status Codes`, only a JSON object containing a description of the error code is sent. These three options cover all the usual ways a server handles errors or unsuccessful requests.

4.2. Apache FreeMarker Template Engine

Apache FreeMarker is a Java library and template engine. It is used to generate dynamic text output, e.g., JSON, XML, etc., based on a generalized template file. Java is used to query, clean and provide data to the template engine, which uses the given templates to dynamically create a text output, e.g. a JSON file that contains the response to an HTTP request. The FreeMarker template language provides conditional blocks, iterations, assignments, string and arithmetic operations and formatting, macros, functions and many more. The engine enables that Java objects are exposed to the template as a tree of variables, so that it can use the variables to insert them at the appropriate place in the template and thus create a text file containing data.

Figure 1c shows an example of the integration of the Apache FreeMarker template engine into the ETARA frontend. While in Figure 1a the configuration of an API was shown (accessible via the form in Figure 1c, marker 1), this section deals with the configuration and creation of a return template (accessible via the form in Figure 1c, marker 2). The template is in JSON format, which means that the API to be simulated will respond using JSON objects.

The (SPARQL) query mentioned in Section 4.1 requests an entity (subject) in the database of the API. Then all information about this entity is collected and passed to FreeMarker in the form of a tree structured Java object. Afterwards, the pre-built functionalities of FreeMarker can be used to parse the Java object, filter needed information and insert it into the template.

An example is given in Figure 1c. In this template, the Java object passed contains all information that is stored in the database about a publication. Line two of Figure 1c iterates over the values of the `author` relation. Afterwards, in line four, for each existing author variable (a publication has one or multiple authors), the name of the author is inserted.

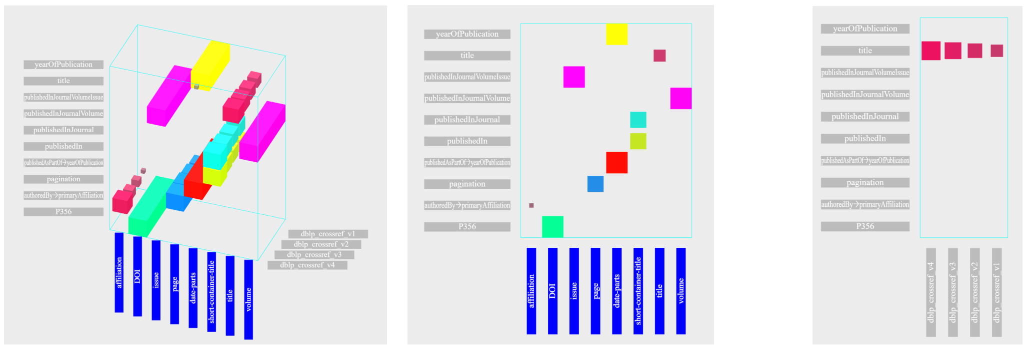
Furthermore, if-conditions (see line five) are easy to implement. However, the most powerful features of FreeMarker are the so-called Built-Ins. These are predefined functions that can be used to change or check the data that is inserted. For example there are Built-Ins to check if strings start with a certain prefix, if substrings are contained and much more. Furthermore it is possible to manipulate the data itself, for example by using a split, trim or remove method. This gives the possibility to create various individual response structures with only one database.

4.3. Alignment Cube Visualization

Many benchmark systems have developed their own visualizations such as MELT, SEALS or HOBBIT. Currently, the OAEI uses HOBBIT and MELT to compare the results of alignment systems. In contrast to HOBBIT, Alignment Cube allows an interactive visual exploration and evaluation of alignments. It is possible to analyze alignments on the level of individual correspondences. Since it was initially designed for dense dynamic networks in the first place, an advantage consists of its scalability. Hence, it allows to visualize the performance history of an alignment system and therefore allows a comparison of the alignment results of a system when different parameters are tuned or core algorithms are changed.

As described in detail by Ivanova et al. [8] two ontologies and their alignment can be seen as a bipartite network of mappings between individual concepts in two different ontologies. Hence, a matrix represents a single alignment between two ontologies/schemas. We have modified this definition slightly and consider rows as relations from an underlying RDF database and columns as paths in the JSON or XML responses of an API in our implementation. Stacking several matrices, i.e., several alignments, creates an Alignment Cube. In Figure 2a, the third dimension represents the history in form of four different alignment results.

The color of each entry of the alignment cube is determined randomly, so that entries in dense alignment cubes can be easily distinguished from each other. Often alignments come with additional meta information, e.g., a confidence score of the individual mappings. These metrics are used to determine the size of the entries in the alignment cube (i.e., the individual cubes themselves). This allows to clearly see, for example, how the level of confidence in a mapping changes when tuning parameters of alignment systems. This feature allows to visualize the performance history of an alignment system. We can analyze more precisely which effects, for



(a) Alignment Cube (b) Alignment Layer (c) Alignment History Layer

Figure 2: Visualization Options using the Alignment Cube Visualization

example, parameter tuning has on the alignment results or whether a change of the algorithm really has an improved effect on the alignments or only on individual mappings.

To analyze such effects in more detail, the visualization offers a various options. The representation of different alignments in a cube allows the restriction to four trivial views. The 3D view (see Figure 2a) visualizes the cube in its entirety using three dimensions, e.g., API paths, RDF relations, and a set of different alignments. The remaining three views are created by viewing individual layers of the cube in isolation (see Figure 2b and 2c). These reduced views can bypass classical disadvantages of 3D views, e.g. distorted depth perception and overlapping objects.

4.4. Data Sets and Gold Standards

ETARA provides ten different data sets[5] of bibliographic and filmographic domains. The bibliographic set contains sample data sets (in RDF format) extracted from dblp, ArXiv, CrossRef, Semantic Scholar and Springer Nature, consisting of meta data of several important conferences and journals (e.g. SIGIR, SIGMOD, CIKM, etc.) from different years (2014 and 2015). We created small data sets because this makes them easier to manually analyze and understand. The bibliographic one contains information about titles, publication years, author names, institutions, references and citations. For legal reasons information like abstracts is not provided.

The filmographic set contains meta data of movies created between 2000 and 2012 and covers samples from Linked Movie Database, Open Movie Database and The Movie Database. The data sets cover titles, release dates, actor names, director names and others. For legal reasons information about the movie plot are not provided in the data sets.

Additionally, the ETARA benchmark system provides 42 API configurations[6], including response templates with different structures and granularity. Moreover, many of the provided templates are based on real world APIs, like the APIs mentioned before. We made sure that the average response time (latency), the behavior in case of errors, as well as the response structure of the original APIs is the same. Moreover, we developed special modifications of

[5]https://github.com/ETARA-Benchmark-System/.github/blob/main/profile/downloads.md
[6]https://github.com/ETARA-Benchmark-System/ETARA-Hub

the aforementioned APIs, for example, that the response structure in the JSON file is flatter, abbreviations are used (for example, for author names or abbreviated path names in the JSON templates) or that data such as the name of an author is split into several relations. Thus, alignment systems can also be tested in special scenarios.

To evaluate automatically created alignments a GS alignment is required. Hence, ETARA provides a set of GS alignments for the given data sets and APIs. Moreover, it provides a component to support the creation of GS alignments. The GS builder consists of four delineated phases: (1) data selection, (2) data cleaning, (3) data mapping and (4) final alignment adjustments.

The first step is data selection, which means that a user must choose which RDF database to align with which RESTful Web API. The next phase is data cleaning, where the response schema of a Web API is to be cleaned of irrelevant components, such as metadata about the request itself. The deleted components of the response are then removed from all requests to avoid burdening the GS designer with unnecessary information.

The third and most significant phase (data mapping) provides a variety of functions [7] to find mappings between the data of a single entity from an RDF database and the response of an API. The selected entity is always chosen randomly from the RDF database to cover a wide variety of entities which leads to a wide range of possible mappings.

When GS designers have processed enough entities they can start the last phase. The individual mappings created during the third phase are used to generate a final alignment proposal. This alignment proposal can be accepted, modified or adapted.

5. Usage Example

In this section we show an example of how ETARA can be used to compare/evaluate two alignment systems. The first step consists of the configuration and simulation of APIs. Then, suitable alignment systems must be selected. Finally, the alignments of the selected systems can be compared and their strengths and weaknesses analyzed.

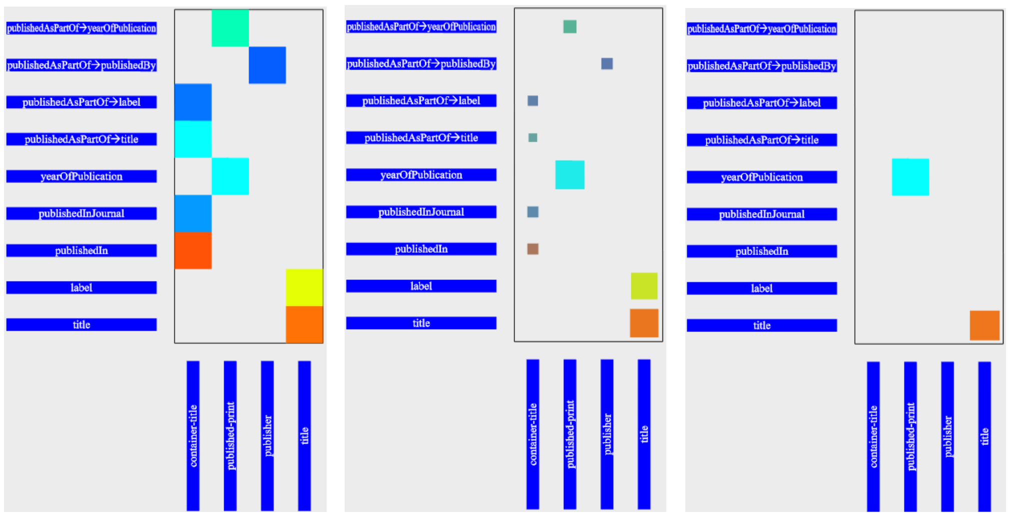
Simulation: To simulate APIs, the first step is the selection of a database for each API, which represents the data basis. Then, the first two tasks described in Section 3, i.e., simulation of APIs and customization of the response templates, must be performed.

To present a simple example of an evaluation, comparing two alignment systems, we created a new bibliographic test API based on data from CrossRef. The simulated API expects the DOI of a publication as a request and is configured to respond after 500 ms to simulate latency. Additionally, the test API will respond with an HTTP error (i.e., 404) if the requested DOI is unknown. These characteristics were configured using ETARAs interface, see Figure 1a and 1b. The created response template contains meta data like the title and year of the publication, the name of the publisher and the title of the conference where the publication was published.

The next step consists of comparing two alignment systems by performing a reproducible evaluation. Hence, the last tasks presented in Section 3, i.e., analyzing alignment results using the presented gold standards and the provided alignment visualization, must be performed.

System Selection: The current state-of-the-art API alignment systems are FiLiPo [1] and DORIS [14]. FiLiPo is a system which automatically finds an alignment between data provided

[7] Search/sort relation names, search/sort values, etc.



(a) Gold Standard Layer (b) FiLiPo Layer (c) DORIS Layer

Figure 3: Alignment Cube Representation of the Alignment Results

by APIs and RDF databases. It is a sample-driven approach that models API services as parameterized queries. Furthermore, it is able to find valid input values for APIs automatically (e.g. IDs) and can determine valid alignments between RDF databases and APIs.

DORIS builds upon the schema of an RDF database and during the alignment process, it sends several requests to an API. Users only have to specify the input class for the API and a request limit. The input class specifies which form of entities the API responds to (e.g. publications). Moreover, a key assumption of DORIS is that it is more likely to find information about popular entities (e.g. famous actors) via API calls. From this follows the assumption that KBs contain more facts of well-known entities and hence it ranks entities by descending number of facts. These entities will then be used for the alignment.

Evaluation: First, an RDF database which is to be aligned with the simulated APIs is needed. In this demonstration we chose to align the RDF version of dblp with the previously created API. Since the dblp and the simulated API (based on CrossRefs data) store bibliographic data of scientific publications, both systems should be able to determine several valid alignments.

DORIS and FiLiPo were executed with the default configurations (described in [14] and [1]). The goal was to create an alignment between the schema of dblp (RDF) and the response template of the simulated API with both systems. Since the selected API is a newly created one, we had to create a GS alignment by using the GS Builder.

Then, the created GS alignment and the alignments created by FiLiPo and DORIS were loaded into the alignment cube visualization. The result is displayed in Figure 3 in the form of three layers of the alignment cube. Each layer represents one alignment (GS, DORIs or FiLiPo).

The next step is to interpret the results. The first noticeable aspect is that the GS, consisting of seven mappings, contains significantly more mappings than the alignment output of DORIS. DORIS was only able to find a mapping for the title and the publication year, which are easy to find, since the title and year values between the API and the dblp are almost identical.

DORIS determines a mapping by comparing the values of the dblp and the API. To compare data values DORIS uses an equality method (ignoring punctuation and case). The drawback of this approach becomes clear when examining the other mappings of the GS alignment. Relations like *publishedIn*, *publishedInJournal* and *publishedAsPartOf* $\rightarrow$ *title* store the conference or journal name, in which a publication was published. Through the manual creation of a GS alignment by using ETARA's GS Builder, a deeper insight into the data can be gained. For example, during the mapping process we noticed that conference or journal titles of dblp and the created test API are very different. For example, an example conference title in dblp looks like this: "Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014". In comparison, the created test API stores conference titles in the following form: "Proceedings of the 23rd ACM International on Conference on Information and Knowledge Management". Therefore DORIS is not able to identify these mappings.

Compared to DORIS, Figure 3b shows that FiLiPo was able to determine a mapping between *publishedIn* and *container-title*. The reason is that FiLiPo uses a set of different similarity metrics and does not check for equality. However, a closer analysis shows that FiLiPo creates these mappings with low confidence values (size of the individual cubes). This indicates that either this mapping is very rare and therefore should be handled with care or (as in this case) that the mapping is not an exact mapping.

In addition to detailed analyses of the alignments and systems, ETARA also enables quick evaluations in the form of precision, recall, and f1-score. DORIS could only find two out of nine mappings and thus has a very low recall of $\frac{2}{9} = 0.22$. FiLiPo, in comparison, was able to achieve a recall of $\frac{9}{9} = 1.0$. Both systems had a precision of 1.0. However, note that that this is not a complete evaluation of both systems, but merely a demonstration that all tasks described in Section 3 are supported by ETARA.

6. Conclusion

With ETARA we have presented a system for the development of alignment systems focusing on Web APIs. We introduced several tasks that are needed to perform reproducible and challenging evaluations. ETARA provides all necessary tools to perform a deeper analysis and evaluation of the alignments generated by the alignment systems, beyond recall, precision and f1-score. ETARA comes with 42 simulated Web APIs based on several real ones. Additionally, we demonstrated the usage of ETARA by performing an evaluation using two state-of-the-art alignment systems. For future work we will focus on a more specified interface to integrate the alignment process of systems like DORIS or FiLiPo more into the visualization components of ETARA, e.g. by showing mapping examples including values from Web APIs and data bases.

References

[1] T. Zeimetz, R. Schenkel, Filipo: A sample driven approach for finding linkage points between RDF data and APIs, in: ADBIS, 2021, pp. 244–259. URL: https://doi.org/10.1007/978-3-030-82472-3_18.

[2] M. Koutraki, N. Preda, D. Vodislav, Online relation alignment for linked datasets, in: ESWC, 2017, pp. 152–168. doi:10.1007/978-3-319-58068-5_10.

[3] T. Sahay, A. Mehta, S. Jadon, Schema matching using machine learning, arXiv (2019). URL: http://arxiv.org/abs/1911.11543.

[4] O. Schmidts, B. Kraft, I. Siebigteroth, A. Zündorf, Schema matching with frequent changes on semi-structured input files: A machine learning approach on biological product data, in: ICEIS, 2019, pp. 208–215. URL: https://doi.org/10.5220/0007723602080215.

[5] S. Hertling, J. Portisch, H. Paulheim, MELT - matching evaluation toolkit, in: SEMANTiCS, 2019, pp. 231–245. URL: https://doi.org/10.1007/978-3-030-33220-4_17.

[6] M. Röder, D. Kuchelev, A. Ngonga Ngomo, HOBBIT: A platform for benchmarking big linked data, Data Sci. 3 (2020) 15–35. URL: https://doi.org/10.3233/ds-190021.

[7] S. N. Wrigley, R. Garcia-Castro, L. J. B. Nixon, Semantic evaluation at large scale (SEALS), in: WWW, 2012, pp. 299–302. URL: https://doi.org/10.1145/2187980.2188033.

[8] V. Ivanova, B. Bach, E. Pietriga, P. Lambrix, Alignment cubes: Towards interactive visual exploration and evaluation of multiple ontology alignments, in: ISWC, 2017, pp. 400–417. URL: https://doi.org/10.1007/978-3-319-68288-4_24.

[9] J. David, J. Euzenat, F. Scharffe, C. T. dos Santos, The alignment API 4.0, Semantic Web 2 (2011) 3–10. URL: https://doi.org/10.3233/SW-2011-0028.

[10] B. Severo, C. Trojahn, R. Vieira, VOAR 3.0 : a configurable environment for manipulating multiple ontology alignments, in: ISWC, 2017. URL: https://ceur-ws.org/Vol-1963/paper550.pdf.

[11] B. Severo, C. T. dos Santos, R. Vieira, VOAR: A visual and integrated ontology alignment environment, in: LREC, 2014, pp. 3671–3677. URL: http://www.lrec-conf.org/proceedings/lrec2014/summaries/851.html.

[12] J. Volz, C. Bizer, M. Gaedke, G. Kobilarov, Silk - A link discovery framework for the web of data, in: LDOW, 2009. URL: https://ceur-ws.org/Vol-538/ldow2009_paper13.pdf.

[13] R. Singh, J. Hidders, F. Xia, J. Kang, A graphical user interface for SILK data link discovery framework, in: OpenSym, 2013, pp. 24:1–24:2. URL: https://doi.org/10.1145/2491055.2491080.

[14] M. Koutraki, D. Vodislav, N. Preda, Deriving intensional descriptions for web services, in: CIKM, ACM, 2015, pp. 971–980. URL: https://doi.org/10.1145/2806416.2806447.

[15] J. Aurisano, A. Nanavaty, I. F. Cruz, Visual analytics for ontology matching using multi-linked views, in: ISWC, 2015, p. 25. URL: https://ceur-ws.org/Vol-1456/paper3.pdf.

[16] P. Lambrix, H. Tan, A tool for evaluating ontology alignment strategies, J. Data Semant. 8 (2007) 182–202. URL: https://doi.org/10.1007/978-3-540-70664-9_7.

[17] E. Peukert, J. Eberius, E. Rahm, AMC - A framework for modelling and comparing matching systems as matching processes, in: ICDE, 2011, pp. 1304–1307. URL: https://doi.org/10.1109/ICDE.2011.5767940.

[18] B. Severo, C. T. dos Santos, R. Vieira, A GUI for visualising and manipulating multiple ontology alignments, in: ISWC, 2015. URL: https://ceur-ws.org/Vol-1486/paper_58.pdf.