Running a reconciliation service for Wikidata

Antonin Delpeuch[1] [0000−0002−8612−8827]

Department of Computer Science, University of Oxford, UK
antonin.delpeuch@cs.ox.ac.uk

Abstract. Data matching is a central part of many contribution workflows in Wikidata. We present a reconciliation service for Wikidata, implementing a standard API that is supported by other data providers and consumed by multiple clients. We explain the technical choices behind the architecture of the service and review its usage patterns in 2019.

Keywords: reconciliation · record linkage · web standard · discovery · Wikidata · OpenRefine

1 Introduction

Aligning datasets which do not share common identifiers is a crucial step in many data integration workflows. This task is known in the literature under many names: data matching, record linkage, reconciliation and many others. This task is heuristic by nature and the techniques used to tackle it can vary widely depending on the application domain, but there are popular patterns for the overall architecture 1.

In Wikidata 3, matching is an important task both for contributors and for reusers. Wikidata contributors constantly need to disambiguate between items, be it for manual editing or data imports. Wikidata reusers often match their own datasets to Wikidata, for instance to pull further data from the knowledge graph, or to uncover duplicates in their own database.

The Wikidata ecosystem offers a range of tools to help with this process. First, Wikidata itself offers a search engine based on ElasticSearch, which can be used to look up items. This supports a range of search operators, such as boolean operators (`AND`, `OR`) or fuzzy search (`Lovelaec~` returns Ada Lovelace (Q7259) as first result). Custom keywords can also be used to look up entities by their property values (`haswbstatement:"P298=RUS"` returns Russia (Q159) as only result). In addition, Wikidata offers autocompletion for inputs which expect entities. This is based on a prefix search on labels and aliases. For instance, typing `USA` in such an input will propose United States of America (Q30) as first option in a drop-down menu. Another useful service to discover entities is the Wikidata Query Service, which offers a SPARQL endpoint on a RDF view of Wikidata. This can be used to formulate more elaborate logical queries than what the search endpoint supports. The search API can be called from SPARQL queries using a `SERVICE` statement, making it possible to combine fuzzy search with advanced logical querying capabilities.

In addition to these official services, the community has built a range of tools to help with matching databases to Wikidata. A popular one is Mix'n'Match[1], a platform where Wikidata users can match external datasets to Wikidata items interactively. Datasets can be uploaded via a tabular format, or fetched from the target websites using user-defined scrapers. When the datasets are associated with Wikidata properties, users can directly add statements to the matched entities to store the third-party identifier in Wikidata as they match the datasets. The platform also supports creating new items for entities which are not yet represented in the knowledge graph.

In this article, we present another such matching tool: a Wikidata reconciliation service.[2] Reconciliation services offer a web API specifically designed to match third-party datasets to some database, such as Wikidata. We will present in Section 2 the queries supported by the service. In short, it is a search API specifically tailored for automated matching, which also offers convenience endpoints for field autocompletion and entity preview. The historic client for this API is OpenRefine, a data cleaning tool initially developed to carry out data imports in Freebase. Over the years, the API has been adopted by other data providers and client software. In 2019, a W3C Community Group was founded[3] to steer the evolution of this API, which had remained mostly undocumented despite its growing adoption.

2 The reconciliation API

The reconciliation API[4] is a web API that data providers can offer, making it easier for clients to match their own data to the database that sits behind the service. Instead of querying a vendor-specific search API, clients can rely on this uniform interface to formulate search queries specifically tailored to data matching problems.

For instance, say we are interested in matching the following dataset of films to Wikidata:

Film title	Director name
The Escape	Dominic Savage
Les Ex	Maurice Barthelemy
La Douleur	Emmanuel Finkiel
Raid Dingue	Dany Boon

Fig. 1. An example table[5] to be matched to Wikidata

Film titles alone are ambiguous, therefore searching for each film title in Wikidata will not give very reliable results. We could concatenate the name of the director to our queries, but that will not let us specify that this name should be matched against the

[1] https://mixnmatch.toolforge.org
[2] https://github.com/wetneb/openrefine-wikibase
[3] https://www.w3.org/community/reconciliation/
[4] https://reconciliation-api.github.io/specs/latest/

value of director (P57) only. As we do not know the Wikidata identifiers of the directors, we will not be able to use the `haswbstatement` syntax for that. Furthermore, there is no mechanism to filter results by type using the user-contributed type system made of instance of (P31) and subclass of (P279).

Instead, we can use the reconciliation API to formulate a query, consisting of:

– a name (for instance ”The Escape”)
– a list of property pairs, each consisting of a property id (such as director (P57)) and a value (such as ”Dominic Savage”).
– a type constraint (such as film (Q11424))

Sending this query to the service will return reconciliation candidates along with scores which quantify how well they match the query. In our running example we would get The Escape (Q39073801) as the first candidate, followed by namesakes such as The Escape (Q58814699). Reconciliation queries can be sent by batch, which speeds up the process of matching large datasets.

Because matching is a heuristic process which often needs to be supervised by humans, the reconciliation API offers auto-complete endpoints which let tools validate manual matching decisions by mapping user input to Wikidata identifiers interactively. It also provides a preview service, which lets clients display hovercards summarizing the contents of Wikidata entities. These aspects of the API are optional: services can decide to support them on an opt-in basis. The reconciliation test bench[6] offers an overview of which aspects of the API are supported by a range of public reconciliation endpoints.

Since the creation of the W3C Entity Reconciliation Community Group, the API has evolved thanks to feedback from users and service providers. For instance, the API originally relied on JSONP, an old technique to perform cross-origin requests in Javascript. Services can now use CORS headers to this end. Other changes are being drafted, and we encourage other stakeholders to join the discussion to ensure that this API meets the diverse needs of the community at large.

3 Architecture of the service

Our Wikidata reconciliation service is small wrapper on top of existing APIs. Its role is to translate the reconciliation queries to Wikibase's own API, produce scores for the reconciliation candidates and return them to the user in the expected format. This wrapper architecture is used by many reconciliation services, but some data providers also provide an official reconciliation service on their own 2.

3.1 Query resolution

Reconciliation queries can have various shapes, which influences the resolution process. Consider the example query of Section 2. To compute the corresponding reconciliation candidates, we proceed as follows:

[6] `https://reconciliation-api.github.io/testbench/`

- retrieve all the subclasses of the target type (film (Q11424)) using a SPARQL query, `SELECT ?child WHERE { ?child wdt:P279* wd:Q11424 }`
- search for "The Escape" both using Wikidata's search service and auto-complete service;
- retrieve all entities appearing in the search results;
- filter out those which do not have one of the sublcasses of film (Q11424) as instance of (P31);
- retrieve the items which appear as director (P57) of the remaining candidates;
- score candidates by comparing their labels and aliases to "The Escape", and the labels and aliases of their director (P57) values to "Dominic Savage". The scores are linear combination of fuzzy-matching scores of these strings.
- return the candidates to the user.

The subclasses of target types and the contents of Wikidata entities are cached in a Redis database to speed up the processing. Because the target types used in reconciliation queries generally remain constant over a long series of batches, this initial fetching of subclasses is normally amortized by further queries.

Each reconciliation query yields two search queries via Wikidata's API, via the `action=wbsearchentities` and the `action=query&list=search` endpoints. The reason for this is that none of the two endpoints can be trusted to surface the relevant candidates systematically. For instance, searching for "USA" in `action=wbsearchentities` will return United States of America (Q30) as first result, but with the same query in `action=query&list=search`, this entity is not present in the first page of results. Conversely, searching for "Lovelace, Ada" in `action=query&list=search` will return Ada Lovelace (Q7259), but will not yield any results with `action=wbsearchentities`.

Since the reconciliation API supports batching of requests, the service performs some of these API calls in parallel to speed up the overall processing time. The service also recognizes some special values for which it avoids querying the search APIs and uses a custom processing instead. This is the case of Wikidata URIs (`https://www.wikidata.org/wiki/Q42`, `https://www.wikidata.org/entity/Q42`), bare Q-ids (`Q42`) and Wikipedia URLs (`https://en.wikipedia.org/wiki/Douglas_Adams`) which are directly resolved to the corresponding items after following redirects. Finally, when a unique identifier is provided as a property, the service first tries to retrieve the candidate items using this unique identifier (via a SPARQL query), and falls back to text search if no matches were returned.

The properties supported in the service are not limited to Wikidata properties such as director (P57): these properties can be combined into property paths, in analogy to SPARQL's property paths. The supported combinators are:

- disjunction: `P57|P58` returns the directors (P57) and screenwriters (P58) of a given item;
- sequence: `P57/P19` returns the date of birth (P19) of directors (P57);
- repetition: `P749*` returns the network of parent organization (P749);
- empty path: `.` returns the item itself, which can be useful with other combinators or to provide alternate labels during reconciliation;
- labels: `Len` returns the label of an item in English;
- descriptions: `Dfr` returns the description of an item in French;

- aliases: `Ade` returns the aliases in German;
- sitelinks: `Sitwiki` returns the sitelink in the Italian Wikipedia

In addition, there are also syntaxes to extract various fields of the datatypes which are supported by Wikibase, by prepending them to the end of the property path. For instance, `P625@lat` will return the latitude of the coordinate location (P625). The full list of supported fields is as follows:

- `@lat` and `@lng` for coordinates;
- `@year`, `@month` and `@day` for dates, with @isodate to format dates in YYYY-MM-DD format;
- `@urlscheme`, `@netloc` and `@urlpath` to return parts of URL (respectively "https", "www.wikidata.org" and "/wiki/Wikidata:Main_page" for Wikidata's main page URL).

3.2 Suggest and preview services

In the reconciliation API parlance, suggest services are the APIs that underpin the auto-complete inputs for entities, properties and types. In the Wikidata reconciliation service, these are just implemented by forwarding calls to the corresponding Wikidata endpoints, except for properties as we need to validate property paths as well. This validation is simply done by parsing the paths and returning it as drop-down option if parsing succeeded.

The preview service renders a small HTML page intended to be embedded in the client as an iframe. In the Wikidata service, entities are previewed by displaying an image associated to them and their description in the target language. If no such description is found, Magnus Manske's autodesc service is used. This generates a description on the fly using a rule-based system.

3.3 Data extension

Data extension allows the reconciliation client to pull data from the target data source, by specifying entity and property identifiers. This fetching is also done by batch. The wrapper uses the same fetching and caching strategy than for reconciliation itself.

In addition, the service also supports suggesting properties to fetch for a particular domain. Given the Wikidata identifier of a particular type, the goal is to return properties generally used on items of this type. For instance, for the type sovereign state (Q3624078), we could suggest properties such as capital (P36) or country calling code (P474).

To compute these proposals, we rely on the property for this type (P1963) property, which was designed for this purpose. It is often the case that the requested type is not annotated with any such statement. To mitigate this, we also include types for superclasses of the requested type. This is done using a SPARQL query which relies on the GAS (Gather Apply Scatter) service in BlazeGraph. This lets us order properties by relevance, from the most specific ones (annotated on the type itself) to more generic ones (on more abstract superclasses).

```
SELECT ?prop ?propLabel ?depth WHERE {
SERVICE gas:service {
    gas:program gas:gasClass "com.bigdata.rdf.graph.analytics.BFS" .
    gas:program gas:in wd:Q3624078 .
    gas:program gas:out ?out .
    gas:program gas:out1 ?depth .
    gas:program gas:maxIterations 10 .
    gas:program gas:maxVisited 100 .
    gas:program gas:linkType wdt:P279 .
}
SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
?out wdt:P1963 ?prop .
}
ORDER BY ?depth
LIMIT 50
```

4 Statistics

In this section, we give a brief overview of the nature of queries sent to the Wikidata reconciliation service in 2019. About 17.6 million queries were processed by the wrapper over this period.

The service takes a mandatory language parameter for all its queries, which determines in which language labels are returned, but also in which language the `wbsearchentities` calls are made (which can influence the query results). As Figure 2 shows, queries are overwhelmingly made in English, perhaps because it is the default language in OpenRefine, which does not make it clear from its user interface that the language used for reconciliation can be changed (because it is infered from the UI language itself).

Figure 3 gives an overview of the types used the most frequently. Because of the hierarchical structure on types, we also include the total count of queries to subclasses of each type, to give a better idea of the popularity of a given domain.

Language	Frequency
English (`en`)	73.5%
German (`de`)	8.0%
French (`fr`)	7.9%
Spannish (`es`)	5.9%
Dutch (`nl`)	1.9%
Japanese (`ja`)	0.6%
Other languages	2.2%

Fig. 2. Proportion of languages used for querying

Target type	Direct uses	Uses of subclasses
human (Q5)	4,033,719	4,033,892
organization (Q43229)	2,043,513	4,491,792
entity (Q35120)	1,172,760	14,132,144
Identificadores (Q21169908)	770,820	770,820
business (Q4830453)	463,202	604,466
scholarly article (Q13442814)	335,624	335,627
city (Q515)	319,695	433,434
film (Q11424)	282,052	322,892
taxon (Q16521)	223,538	223,547
village in India (Q56436498)	214,534	214,534
railway station (Q55488)	203,648	213,386
commune of France (Q484170)	170,390	170,629
album (Q482994)	164,653	165,267
branch post office (Q61443690)	144,909	144,909
university (Q3918)	139,461	151,569
title (Q783521)	139,411	139,419
family name (Q101352)	125,531	125,671
educational institution (Q2385804)	111,726	429,278
State Bank of India branch (Q65954115)	110,508	110,508
airport (Q1248784)	106,165	107,361
comune of Italy (Q747074)	101,629	101,852
video game (Q7889)	99,180	99,180
periodical (Q1002697)	91,813	221,330
software (Q7397)	91,772	121,857
district of India (Q1149652)	86,730	86,730

Fig. 3. Number of times the most 25 popular types were reconciled against.
The second column indicates the number of queries against the type or any of its subclasses.

5 Conclusion

This service and the underlying API could be improved in many ways. Although the reliance on existing search APIs from Wikidata makes it simple to deploy the service and keep it synchronous with Wikidata, it also limits our capacity to adapt the indexing profile to the needs of our users (which leads us to run two search queries for each reconciliation query as explained in Section 3). Running our own search index on top of Wikidata could help and it would also let us submit search queries in batch rather than individually. Ideally, the service could be offered by the Wikibase instance itself, for instance as a MediaWiki extension. The existing service can be used on other Wikibase instances as long as they provide a SPARQL endpoint.

References

Christen, P.: Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer Science & Business Media (2012)

Delpeuch, A.: A survey of OpenRefine reconciliation services. arXiv:1906.08092 [cs] (Aug 2019)

Vrandečić, D., Krötzsch, M.: Wikidata: A free collaborative knowledge base. Communications of the ACM (2014). https://doi.org/10.1145/2629489