Comparative Study of Tools for the Integration of Linked Open Data: Case study with Wikidata Proposal

Roxana Martinez*, Gonzalo Pereyra Metnik

Universidad Argentina de la Empresa, Buenos Aires, Argentina

Abstract
The approach of this work is based on a comparative analysis of several tools used for the management of linked open data. To do this, different sections are shown, initially an introduction to the topic is presented, identifying the importance of open data for people, and showing how the results generated through them can benefit. In addition, special emphasis is placed on the need for tools that facilitate both their interaction and manipulation. A section is also shown with related works on this topic, identifying previous studies that evaluated similar tools. The objective is the presentation of a comparative analysis of tools on the study of a proposed data model, which is used to perform tests with five selected tools: 1) API, 2) SPARQL in Wikidata Query Service (WDQS), 3) Pywikibot and 4) OpenRefine with the Wikidata plugin. The main task is to identify, for each tool, the evaluation in terms of its ability to integrate and manage Linked Open Data, its ease of use, and its performance in executing specific tasks. Finally, the results obtained in the performance of the analyzed tools are shown, allowing us to know the strengths and weaknesses of each one concerning the proposed data model. On the other hand, the findings of the study are presented that suggest recommendations for the selection of the most appropriate tool according to the specific requirements needed in aspects of data integration in a project.

Keywords
Linked Open Data, Wikidata, SPARQL Wikidata Query Services, Pywikibot, OpenRefine

1. Introduction

The use of open data is increasingly on the rise, so the integration of this data with software applications, as well as with database engines, today represents an important challenge in the treatment of technologies. Linked open data represents an important factor in the development of technological applications and services. The correct integration of this data allows for improving aspects of interoperability and accessibility of public information. Open data is available to anyone who wants to access it from anywhere in the world, so it is available on institutional websites or government websites. However, the large volume of data and its growing demand do present complex challenges in terms of its processing and availability. It is for this reason that it is necessary to incorporate appropriate tools that can deal with these new knowledge challenges efficiently.

1.1. Context

Linked data represents a set of good practices approved by the World Wide Web Consortium [1], designed for the publication of information on the web, thus facilitating communication between those who share said information. "The integration of open access policies and linked data have generated the position of Linked Open Data (also known as Linked Open Data or simply LOD)" [2], these provide a set of services for the publication of resources to through some technologies such as: SPARQL or LOD API.

One of the points worth mentioning as an important focus is that "LOD is often considered as a virtual data cloud where anyone can access any data they are authorized to see and can also add data

ICAIW 2024: Workshops at the 7th International Conference on Applied Informatics 2024, October 24–26, 2024, Viña del Mar, Chile
*Corresponding author.
mariarmartinez@uade.edu.ar (R. Martinez); gpereyrametnik@uade.edu.ar (G. Pereyra Metnik)
0000-0001-9318-6411 (R. Martinez)

without altering the original data source. This provides an open environment where data can be created, connected, and consumed at Internet scale" [3].

The World Wide Web Consortium [1] "recommends the use of URIs (Uniform Resource Identifiers) and HTTP URIs to access information about resources through a data exchange model such as RDF (Resource Description Framework). In addition, it proposes incorporating other links, as a complement, to the use of an RDF triplet query language. For example, SPARQL, which is recommended by the W3C" [4]. These recommendations are shown in the international publication of Good Practices for Linked Open Data [5], along with some technical aspects to consider for the implementation of this technology. Many authors agree that this new trend could incorporate numerous benefits, for example, according to Berners-Lee [6], LOD is based on the principles of the semantic web. It uses technologies such as RDF and URIs to establish specific connections between different data sets, thus facilitating its interoperability and reuse that could improve government management in matters of Social Development for different citizens. Linked open data is published, consumed, and managed by multiple organizations around the world, thus generating new possibilities for the analysis of large volumes of information.

On the other hand, there are different linked open data repositories, one of the best known is Wikidata [7]. This repository allows various users to work with wiki-like features and is managed by the Wikimedia Foundation. An interesting feature in this context is that it is not necessary to dynamically update the content of each wiki, since there is common data, for example, statistics, dates or locations that can be centralized and related through the Wikidata platform, thus generating, a collaborative approach. In addition, specific information on some relevant considerations for data processing, scripting, tools, and other resources to consider when processing linked open data is presented on its official website [8]. Wikidata stands out for its great potential in interoperability and its ability to connect data from various domains. This great ability to link information efficiently is what makes it one of the most powerful tools in the technological environment. On the other hand, its potential is exploited by researchers, programming developers, and organizations that use these good practices to create advanced applications, driving progress in the field of knowledge and technology.

As a last point to comment on, Wikidata has a wide thematic coverage that allows the integration of multiple data sources, this allows for generating a great benefit between the semantic connections that could be made and also enriches the interoperability between different data sets [9]. This feature gives users the ability to extract data that can be processed and obtain information that could be interconnected to obtain significant value. The latter takes on fundamental value in this context since it allows the heterogeneity of the data to be efficiently managed and at the same time maintains the quality of the linked information. These are fundamental points when it comes to data integration because they need to be accurate and functional. This technique is crucial to guarantee the linking of data, maintaining both its content and its integrated metadata so that they can be combined and used [10]. That is why it is necessary to take full advantage of Wikidata's ability to manage these characteristics, in aspects of collaborative knowledge.

1.2. State of the art

In recent years, various solutions and tools have been developed for the integration of linked open data. These tools have similar and different characteristics, for example: the use of APIs, consultation services, and plugins, among others. Although notable progress has been made, challenges remain, including data heterogeneity, quality of integration, and usability of tools. Previous research has explored different aspects of these solutions, providing a basis for understanding the current strengths and limitations in this field.

It is worth mentioning that several repositories and projects are important in the LOD context, such as DBpedia works with the extraction of structured data from Wikipedia and converts it into a set of Linked Open Data. DBpedia is used for applications that require general information and is one of the first initiatives in LOD. Although there are several investigations [11, 12], it stands out for working with assignments that are created through a global collaborative effort and allow combining knowledge from the different editions of Wikipedia.

Another repository is GeoNames, there are several works [13] in which they use this as a geographic resource that provides place names and coordinates, used in applications that require linked geographic information. For example, using a methodology to semi-automatically generate a tourism knowledge graph (TKG), which can be used to support a variety of intelligent services in this space, and a new ontology to model this domain, the analysis ontology tourism (TAO). Another work is presented by Brandt [14], which uses resources related to LOD and repositories such as Geonames and DBpedia for semantic improvement. The study allows us to conclude that the making available of government data, especially legislative data, can be done following the recommendations and good practices of the W3C.

Another notable repository is LinkedGeoData, which features a project that converts OpenStreetMap data into LOD, allowing geospatial information to be queried and used alongside other linked data. Several works indicate that with indicated redesigns it is possible to consult these data sets in English, German, French, Italian, and Spanish, among others [15, 16]. As mentioned above, there are several repositories, this work will be based on the study of Wikidata because it stands out in this context for its great interoperability and the large community that supports it, but it is necessary to understand that the choice of a reference in LOD depends on the specific context and needs of the project in question.

1.3. Motivation and Objective of the Study

Although significant progress has been made in the availability of Linked Open Data, the integration and management of this data remain considerable challenges. The large amount and diversity of data available can cause problems such as format incompatibilities, variations in data quality, and difficulties in synchronizing information between different sources [17]. These shortcomings mentioned above can limit the great benefits of interconnecting linked open data, which could make a correct analysis of the information impossible. For this reason, it is essential to have tools that facilitate the integration of this data from multiple data domains, in addition, it must be ensured that the quality and metadata are consistent between the linked information. This need to have effective solutions is increasingly vital since open data today continues to grow not only in large quantity but also in great variety, which makes it more necessary to have efficient tools to take advantage of these resources. These tools must be well designed and tested to significantly improve the ability to manage this linked open data since incorrect data or data that does not maintain quality in its presentation could encourage the development of applications with erroneous information.

The objective of this work is to analyze various tools used for the integration of Linked Open Data. The study seeks to evaluate the performance of these tools in different aspects, for this, a data model proposed by the authors was designed. Basically, four specific tools that present great potential in aspects of interaction with Wikidata will be analyzed: 1) the Wikidata API; 2) the SPARQL in the Wikidata Query Service (WDQS); 3) Pywikibot; and 4) OpenRefine with its Wikidata plugin. Also, this work will identify the strengths and weaknesses of each tool in terms of its ability to integrate and manage linked data, as well as its ease of use and performance on specific tasks. Through this tool evaluation, the authors provide good practice recommendations for selecting the most appropriate tool, which presents specific data integration requirements in future projects.

2. Methodology

To carry out the analysis of the indicated tools, a work methodology is presented that includes the selection of the tools, with the explanation of the most relevant characteristics of each one. On the other hand, the definition of the evaluation metrics that will be considered for this study is presented. Below is a comparison of each of these allowing an analysis of their most relevant functionalities, as well as their limitations.

2.1. Tool Selection Criteria

Wikidata is considered one of the most important sources of structured data on the web. It was developed in order to promote adequate access and manipulation of this type of data. For this technology, the use of APIs that allow interaction with direct queries for this linked data structure stands out. In addition, SPARQL query services are incorporated, such as Wikidata Query Service (WDQS), and automation tools such as Pywikibot. These types of technologies are specialized, such as Wikidata Toolkit (WDTK), because they offer efficient solutions for the management of large volumes of data, while applications such as OpenRefine, with its Wikidata complement, allow data processing, along with analysis techniques. data cleaning. For the latter, a great advantage is that they can be very accessible to users without programming experience.

This research will analyze four tools that correspond to the context of Wikidata: 1) the Wikidata API; 2) the SPARQL in the Wikidata Query Service; 3) Pywikibot; and 4) OpenRefine with its Wikidata plugin. These have been selected for their relevance and applicability in the integration of Linked Open Data (LOD), in addition to their ease of use and their ability to efficiently handle data manipulation tasks in contexts where large volumes of data are not required. Regarding the selection of the SPARQL API and the query service, they were considered since they are fundamental for the extraction and manipulation of data in Wikidata. On the other hand, Pywikibot was considered because it stands out in the automation of repetitive tasks, and finally OpenRefine, for being a powerful platform in aspects of data cleaning and transformation, thus guaranteeing the quality and consistency of the information linked.

Finally, for this study, it was decided to eliminate Wikidata Toolkit (WDTK) because, although it is a powerful tool for executing complex SPARQL queries and handling large volumes of data, this study focuses on solutions that are targeted to contexts for a data model proposed simple, so the data volume is not that significant. Additionally, WDTK is designed to manage and process large amounts of RDF data, which exceeds the needs of our analysis, which focuses on integrating and manipulating data at a more manageable scale. By opting for tool styles such as the Wikidata API, SPARQL in Wikidata Query Service, Pywikibot, and OpenRefine with its Wikidata plugin, the study seeks to offer a more practical and common application-oriented perspective on Linked Open Data integration, without the additional complexity that these more advanced tools would handle.

2.2. Tools Description

This section indicates a study of some of the most relevant works on the technical aspects of each of the tools used.

Wikidata API: Allows access to Wikidata data through HTTP requests, obtaining data in different formats such as JSON or XML. Wikidata uses the Stories Services API to generate multimedia stories related to people, organizations, and periodicals [18]. Application developers today have three options for exploiting the knowledge present in Wikidata: they can download Wikidata dumps in JSON or RDF format, they can use the Wikidata API to obtain data about individual entities, or they can use the Wikidata SPARQL endpoint Wikidata [19]. There is also research on the gene-drug interaction database as a web resource that provides information on gene-drug interactions and selectable genes from publications [20].

SPARQL Query: Query service that allows you to access and manipulate data stored in RDF (Resource Description Framework) format through the use of SPARQL, a query language standardized by the W3C. Some works present how Wikidata is integrated into the Linked Data Web and how the SPARQL Query Service facilitates consultation and access to linked data. Other studies show how Wikidata provides a SPARQL access point, allowing users to perform complex queries on an open data set. This not only improves Wikidata's interoperability with other linked datasets on the semantic web, but also allows researchers and developers to exploit Wikidata information [9]. Another study focuses on: Implementation of a current research information system (CRIS) that uses semantic technologies to integrate and manage data from various sources in the university environment. This system allows advanced queries through a SPARQL Point, which facilitates the retrieval of structured and detailed information on scientific production [21].

Pywikibot: Bots and automation tool for interacting with Wikidata and other Wikimedia projects. There are research papers that delve into topics such as wiki projects, voluntary contributions, notable Python contributions, an overview of PAWS (Python Data Science Environment for Jupyter) and PyWikiBot [22]. Another study focused on the work presents an overview of the Chinese Wikiproetas project, using OpenRefine and PyWikibot to improve the names of more than 4,000 Chinese women poets in Wikidata with great contributions of technical aspects for the use of this tool [23].

OpenRefine with Wikidata plugin: Data cleansing and reconciliation tool with the ability to link local data to Wikidata. Data in OpenRefine can be easily transformed into Wikidata declarations by creating a dedicated export schema that maps each column to respective elements and properties. Some works show several preparation techniques for processing directly with QuickStatements, allowing editing and creating Wikidata elements directly from OpenRefine [24]. Another study proposes generating a repository with an open database for academic publications using Wikidata with OpenRefine technology [25].

As mentioned in other sections, there are many notable features of each of the tools mentioned, these relate to the Wikidata API allowing developers to interact with Wikidata, and providing access to read, write and query data. In addition, it facilitates the management of Wikidata elements, properties and descriptors, integrating tasks such as the creation and modification of elements, data search, and retrieval of structured information in formats such as JSON and XML. On the other hand, SPARQL Query Service presents a service that allows complex queries to be performed on RDF data using the SPARQL language. Additionally, it offers the ability to search for specific patterns, filter results, and combine information from multiple linked data sources. This service allows the use of data in formats that are compatible with different applications and also allows data reading and modification operations, which allows adequate interoperability. Pywikibot works with Python, it was designed to automate interactions with MediaWiki, including Wikidata. It also allows the creation of scripts to perform repetitive tasks, update elements, and manage data, making it easier to maintain Wikidata content. Finally, OpenRefine with Wikidata Plugin is a data cleansing and transformation tool, which can be used with Wikidata data. This plugin facilitates the identification of various objects, adds characteristics to them and updates the data referring to them.

2.3. Evaluation Metrics

This section presents the evaluation metrics that were considered to analyze and compare the different selected tools. Criteria have been selected for an evaluation that includes technical aspects and also its ease of use in aspects of extraction and processing of linked data. In addition, measurements are made in aspects of data access speed, response times, and efficiency in information retrieval. Another important point that is considered is interoperability, which examines the ability of tools to interact and be compatible with other software. The scalability of each tool is also studied, considering the growing volume of data and users who use it. Finally, it is analyzed what support each tool provides in technical aspects in terms of help resources and its activity in the Community of users and programmers.

2.4. Proposed LOD model

Figure 1 shows the data model proposed by the authors, it shows the relationship between the different objects related to educational and geographical aspects using the ontology and properties of Wikidata. The hierarchical and spatial relationship of universities such as the “Universidad Privada del Valle” and the “UADE” (Argentine Business University) with their respective countries, Bolivia and Argentina, is highlighted, using the country property (P17). Furthermore, the entity of Buenos Aires is represented as the capital of Argentina through capital ownership (P1376) and as an instance of an Argentine city. It also illustrated how Argentina and Bolivia are part of a larger entity, “Latin America,” using the ownership of part of (P361). This model not only connects educational institutions to their administrative and political locations but also places these locations in a broader regional context, thus demonstrating the ability to link data across different levels of geographic and thematic granularity.

This proposal, as an example, is a data model that not only allows for a clear and structured understanding of the relationships between different educational and geographic entities but also facilitates data interoperability in a broader context. By leveraging the ontologies and properties of Wikidata, a coherent and linked representation of information is achieved, which can be used in various applications, from academic research to the development of geographic information systems (GIS).

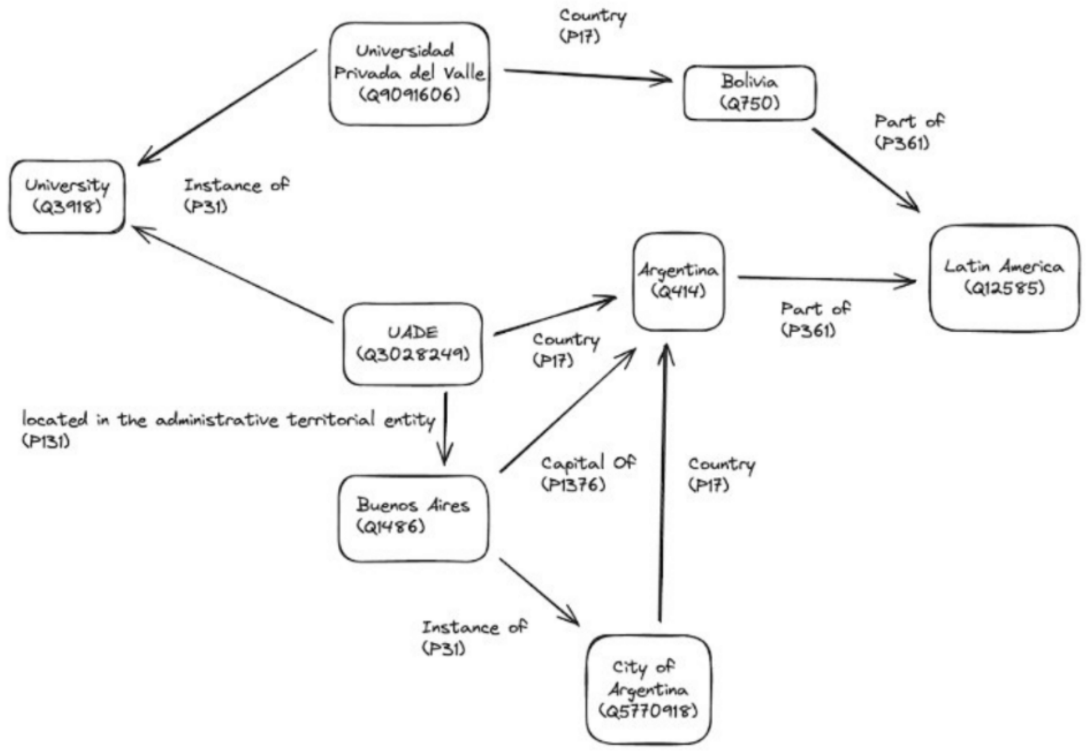


Figure 1: Proposed data model.

2.5. Testing Procedure

With the proposed linked open data model and the tools selected for study analysis, the testing procedure to be carried out will be carried out. That is, the methodology used to evaluate and validate the tools and technologies considered in this study consisted of a series of steps that allow simulating real-use scenarios with the proposed data model, allowing performance to be measured and compared using the metrics previously established in the previous section. Initially, test environments were set up, ensuring that all tools were correctly installed and updated. Data extraction and access tests were then carried out using standard data sets, measuring ease of use and response speed. Execution times were recorded and the tool's ability to handle output conversions to different formats was observed. Subsequently, the interoperability of the tools concerning other software was analyzed, as well as scalability aspects through different tests. Finally, studies were obtained on the technical support provided on public websites, which would allow finding solutions to possible problems; some of the sites analyzed were Internet forums.

3. Results

This section analyzes the main results obtained during the evaluation of the tools studied. This analysis presents the main benefits and limitations of each tool. Below are detailed: a) comparison of tools; and

b) comparative analysis.

3.1. Comparison of Tools

3.1.1. Wikidata API

For the wiki data API, some techniques were used that provided advantages when using it. The Wikidata REST API [26] provides a simple and efficient interface for performing quick queries and obtaining structured data from Wikidata, which is ideal for developers looking to integrate data efficiently. As for what you respect about the documentation provided for this API, API REST the Wikidata in Javascript, looks at a set of advanced functionalities that are developed on its website, and that are explained for integration with Javascript technology. The Wikidata REST API manual in PHP helped provide information on a fluid integration of Wikidata functions in systems based on this language. Finally, Wikidata's Stable Interface Policy [27] helped to understand how to guarantee that the interfaces and APIs used will remain stable over time, providing security and confidence to developers by ensuring that their applications will not be affected. due to unexpected changes in the API. Below is a qualitative analysis of the aspects considered by each tool:

[Documentation] *Clarity and Complexity:* The documentation is well structured and clear for experienced users but can be overwhelming for beginners. RDF and JSON-LD are not sufficiently explained to users. *Examples of use:* The examples shown are useful, but they are very simple. Additionally, there is a lack of guidance on query optimization. Finally, the documentation is not located in one place but is spread across multiple sources.

[Ease of use for data extraction] *Learning curve:* The API is intuitive for programmers dealing with RESTful APIs but is very complex for users without prior experience in Wikidata or RDF. Another point to keep in mind is that there are no graphical interfaces that facilitate its use. *Ease of access to complex data:* API allows access to complex data. *Tool compatibility:* It is compatible with data analysis tools such as Python, but its integration is not simple.

[Data access speed] *Average Latency:* The Wikidata API offers fast response times for simple queries. *Performance under load:* API can exhibit poor performance when performing very complex queries. There aren't many options to optimize the speed of these queries, which can be a problem in applications that require real time access.

[Ease of output conversion] *Available Formats:* The API allows you to obtain data in JSON, RDF, and XML formats, which provides greater flexibility in the use of different applications. *Ease of transformation:* JSON is one of the most compatible formats with other applications, but converting from RDF may require specialized knowledge, as RDF can be complicated to handle.

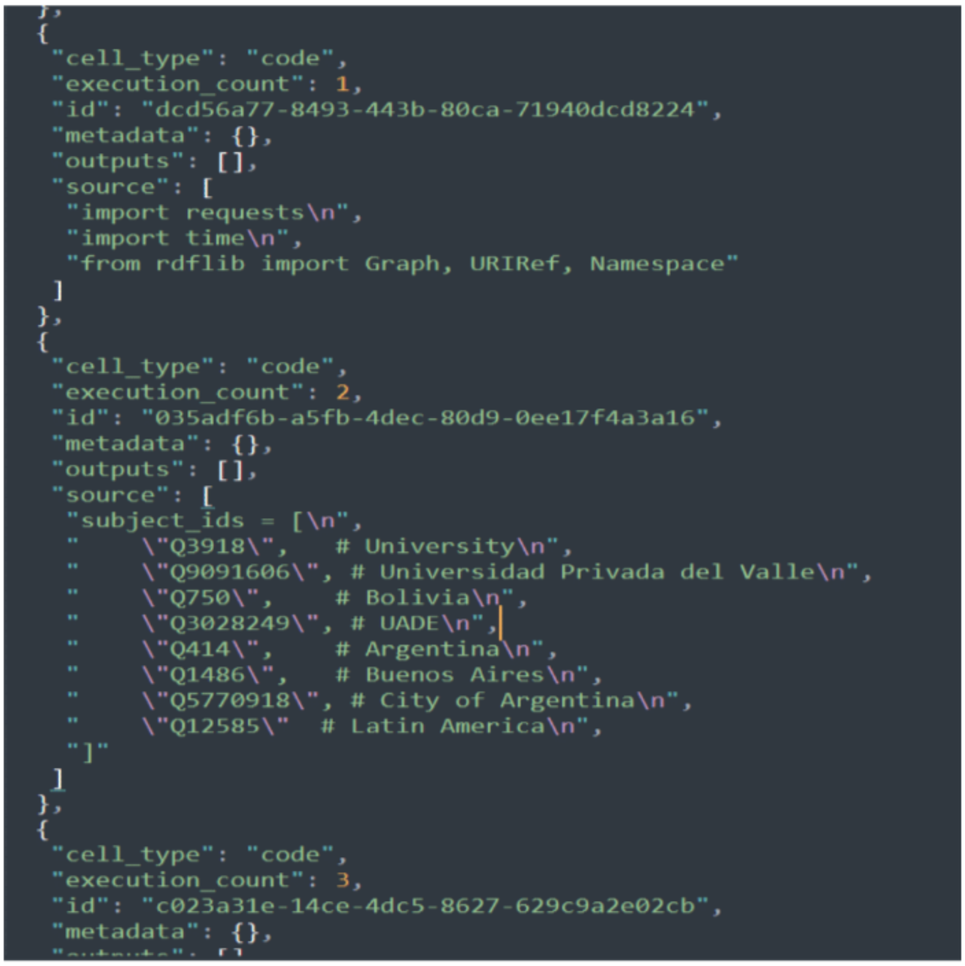
[Interoperability] Compatibility with Standards: The API uses standards such as RDF and JSON-LD, making it easy to integrate with other Linked Open Data (LOD) systems. *Integration API:* Supports other LOD APIs and services, allowing you to combine data from multiple sources for more complete analysis.

[Escalabilidad] *Request Limits:* The API has limits on the number of requests per second to prevent overloading, which can be an issue in large-scale applications. *Scalability management:* The API is suitable for small and medium-sized applications. *Support for large volumes of data:* It can handle large volumes of data, but with performance that decreases under high concurrent demands.

[Support and community] *Technical Support Availability:* Wikidata has adequate technical support, but the majority of support comes from the user community. *Community Resources:* There are active forums, mailing lists, and community resources that are valuable for resolving common questions and problems. *Documentation Updates:* Documentation and resources are not updated regularly, and support can be slow and highly dependent on community participation.

[License and terms of use] *License Type:* The Wikidata API operates under the Creative Commons CC0 1.0 license, allowing unrestricted use of the data. *Use Restrictions:* There are no significant restrictions, allowing free redistribution and modification of the data, which is ideal for open projects. *Rights and Obligations:* Users are required to follow the rules of the CC0 license but are free to use the data in commercial or noncommercial applications without the need for attribution.

A small prototype was developed that is prepared to interact with the proposed data model, in addition, it is designed to interact with Wikidata through its REST API. Defines lists of subject identifiers and relevant predicates (such as universities and countries) for performing RDF queries. Use the requests library to make API requests and rdflib to handle the resulting RDF data. In the Figure 2 shows how an RDF graph is developed and manipulated using Python and the rdflib library. The code begins by importing the necessary libraries (requests, time, and rdflib), which are essential for making HTTP requests and handling RDF data. Below are identifiers (subject_ids) corresponding to specific entities in Wikidata, such as "University" (Q3918), "Universidad Privada del Valle" (Q9091606), "Bolivia" (Q750), "UADE" (Q3028249), "Argentina" (Q414), "Buenos Aires" (Q1486), "City of Argentina" (Q5770918) and "Latin America" (Q12585). These entities represent nodes in the RDF graph that models the relationships between universities and their geographic and political locations. The model shows how the relationships between these entities are structured using Wikidata properties, such as: "Country" (P17), "Part of" (P361), "Capital of" (P1376) and "Instance of" (P31). In this context, the source code allows the creation and use, through a graph, of data connections, access to queries, and analysis of these. This approach is valuable for organizing and linking information about educational entities, and their geographic and political context in a structured and easy-to-use way.



```
 },
 {
  "cell_type": "code",
  "execution_count": 1,
  "id": "dcd56a77-8493-443b-80ca-71940dcd8224",
  "metadata": {},
  "outputs": [],
  "source": [
   "import requests\n",
   "import time\n",
   "from rdflib import Graph, URIRef, Namespace"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 2,
  "id": "035adf6b-a5fb-4dec-80d9-0ee17f4a3a16",
  "metadata": {},
  "outputs": [],
  "source": [
   "subject_ids = [\n",
   "    \"Q3918\",    # University\n",
   "    \"Q9091606\", # Universidad Privada del Valle\n",
   "    \"Q750\",     # Bolivia\n",
   "    \"Q3028249\", # UADE\n",
   "    \"Q414\",     # Argentina\n",
   "    \"Q1486\",    # Buenos Aires\n",
   "    \"Q5770918\", # City of Argentina\n",
   "    \"Q12585\"   # Latin America\n",
   "]"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 3,
  "id": "c023a31e-14ce-4dc5-8627-629c9a2e02cb",
  "metadata": {},
```

Figure 2: Python source code to interact with the Wikidata API and use RDF data.

3.1.2. SPARQL in Wikidata Query Service (WDQS)

For SPARQL in Wikidata Query Service (WDQS), various techniques were used that offer specific advantages depending on their application [28]. The source code that builds the RDF graph is made from Wikidata entities and properties, which is useful in the context of SPARQL queries in Wikidata Query Services [29], because it allows you to structure and organize Wikidata data in a standardized format that is compatible with SPARQL. By transforming entities and relationships into RDF triples, this code facilitates the creation of SPARQL [30] queries that can be run on Wikidata to extract, analyze, and visualize complex information. In essence, it allows Wikidata data to be queried more efficiently and accurately through SPARQL, which is essential for any type of analysis or application that relies on linked data.

[Documentation] *Clarity and Complexity:* The SPARQL query service documentation on Wikidata is comprehensive and explains in detail how to use SPARQL to perform complex queries. It includes an introduction to SPARQL, but due to the inherent complexity of this query language, it can be intimidating for novices. *Usage Examples:* The documentation provides numerous query examples covering a wide range of use cases, which is very helpful in understanding how to structure specific queries. *Coverage:* The documentation covers almost all aspects of using SPARQL, from basic to more advanced queries, including how to optimize queries to improve performance. However, some specific use cases, such as manipulating complex results, are not fully detailed.

[Ease of use for data extraction] *Learning Curve:* SPARQL is a powerful language but has a steep learning curve. Understanding both SPARQL and the structure of data in Wikidata is necessary to perform effective queries. *Ease of access to complex data:* WDQS is effective in accessing complex and interrelated data. That is, it allows complex queries that combine multiple entities and properties. *Tool Support:* SPARQL query results can be exported in various formats (such as JSON, XML, CSV), making it easy to integrate with data analysis and visualization tools. However, building and maintaining complex queries may require additional optimization and visualization tools.

[Data access speed] *Average Latency:* Response time is fast for simple queries. However, latency can increase significantly with complex queries. *Performance under load:* WDQS has some performance issues when performing very complex queries. Sometimes the service imposes limits on the queries that can be performed to avoid system overload. *Optimization mechanisms:* The documentation includes suggestions for optimizing queries, but optimization requires a deep understanding of both SPARQL and the structure of the data in Wikidata.

[Ease of output conversion] *Available Formats:* Various formats can be used, JSON, CSV, XML among others. This facilitates its use in different applications and analyses. *Ease of transformation:* JSON and CSV are easy to handle and transform, making it easy to integrate into other analysis tools. However, transforming complex SPARQL data queries into simpler structures can be a challenge. *Compatibility:* The formats are compatible with standard data analysis tools. Handling more complex formats, such as RDF, may require specialized knowledge.

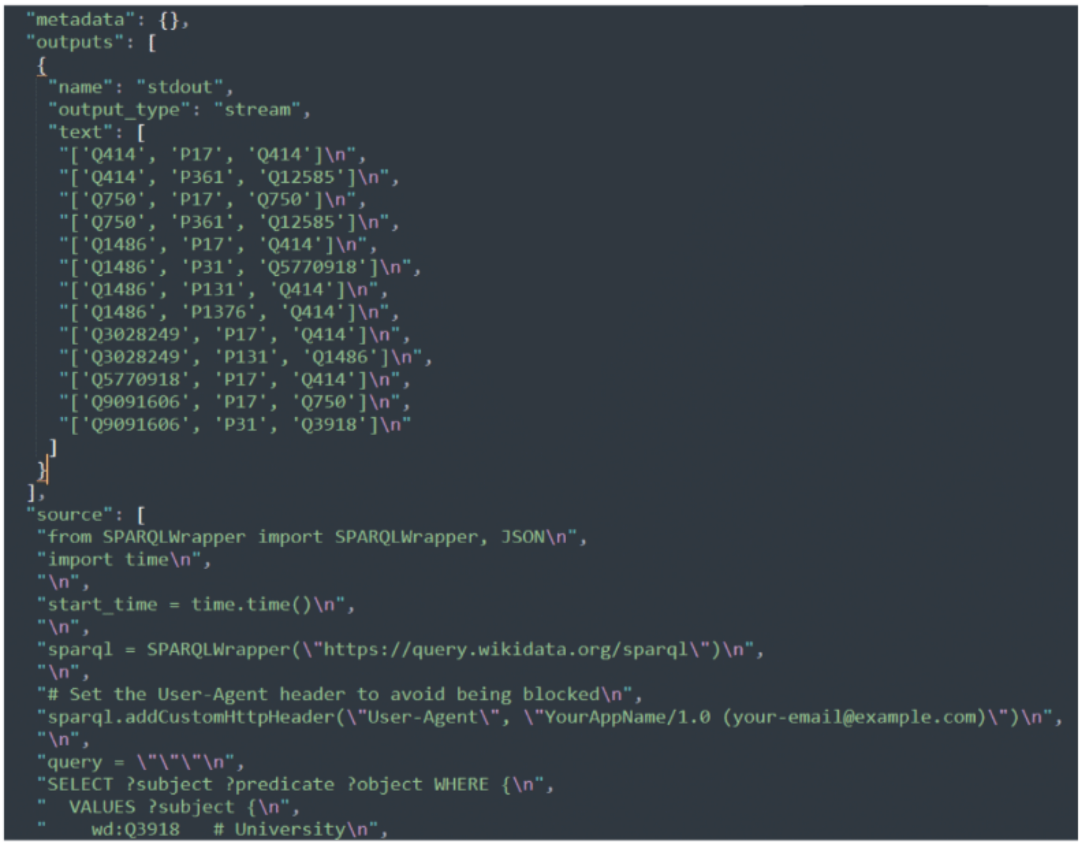
[Interoperability] *Standards Compatibility:* SPARQL is an accepted standard for queries on RDF data, ensuring its compatibility with other Linked Open Data (LOD) systems and services. *Integration API:* WDQS can be easily integrated with other services and APIs that support RDF and SPARQL, making it easy to exchange data between different platforms. *Interoperability Protocols:* WDQS supports standard interoperability protocols.

[Scalability] *Request Limits:* The service has limits on the size and complexity of queries to avoid overloading, which can be a drawback in projects that require complex queries. Scalability management: WDQS is suitable for small and medium-scale queries. Support for large volumes of data: Although it is possible to handle large volumes of data, performance can be significantly affected.

[Support and community] *Technical Support Availability:* Wikidata provides adequate technical support through its user community, but direct support is limited. *Community Resources:* There is an active community with forums and mailing lists where users can share solutions and advice, which is very valuable in solving common problems. *Documentation Updates:* Documentation and community resources are updated periodically.

[License and terms of use] *License Type:* Data obtained through SPARQL queries in WDQS are subject to the Creative Commons CC0 1.0 license, allowing unrestricted use. *Restrictions on Use:* There are no significant restrictions, allowing free redistribution and modification of the data. *Rights and Obligations:* Users are required to follow the rules of the CC0 license but are free to use the data in commercial or noncommercial applications without the need for attribution.

Figure 3 shows the source code shows how to perform a SPARQL query through the SPARQLWrapper library to extract data from Wikidata. The query result is presented as a list of triples, where each triple contains a subject, a predicate, and an object, represented by their identifiers in Wikidata (such as 'Q414' for Argentina or 'P17' for "country"). These triples describe relationships between entities in Wikidata, allowing us to understand how they are connected. The code also sets up a custom User-Agent header to avoid being blocked by the Wikidata query service.



```
"metadata": {},
"outputs": [
 {
  "name": "stdout",
  "output_type": "stream",
  "text": [
   "['Q414', 'P17', 'Q414']\n",
   "['Q414', 'P361', 'Q12585']\n",
   "['Q750', 'P17', 'Q750']\n",
   "['Q750', 'P361', 'Q12585']\n",
   "['Q1486', 'P17', 'Q414']\n",
   "['Q1486', 'P31', 'Q5770918']\n",
   "['Q1486', 'P131', 'Q414']\n",
   "['Q1486', 'P1376', 'Q414']\n",
   "['Q3028249', 'P17', 'Q414']\n",
   "['Q3028249', 'P131', 'Q1486']\n",
   "['Q5770918', 'P17', 'Q414']\n",
   "['Q9091606', 'P17', 'Q750']\n",
   "['Q9091606', 'P31', 'Q3918']\n"
  ]
 }
],
"source": [
 "from SPARQLWrapper import SPARQLWrapper, JSON\n",
 "import time\n",
 "\n",
 "start_time = time.time()\n",
 "\n",
 "sparql = SPARQLWrapper(\"https://query.wikidata.org/sparql\")\n",
 "\n",
 "# Set the User-Agent header to avoid being blocked\n",
 "sparql.addCustomHttpHeader(\"User-Agent\", \"YourAppName/1.0 (your-email@example.com)\")\n",
 "\n",
 "query = \"\"\"\n",
 "SELECT ?subject ?predicate ?object WHERE {\n",
 "  VALUES ?subject {\n",
 "    wd:Q3918   # University\n",
```

Figure 3: Extracting RDF triples from Wikidata using SPARQL.

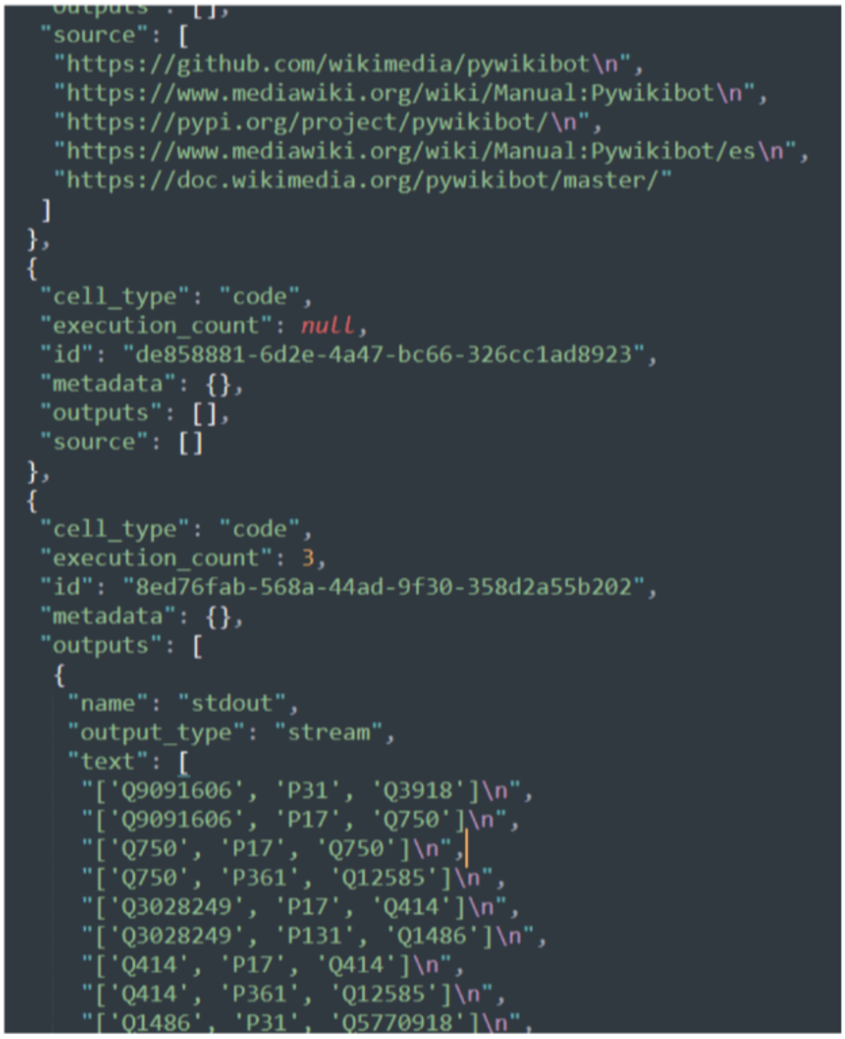
3.1.3. Pywikibot

Some sources are used as a study to carry out the test with a developed prototype [31, 32]. Below is the analysis carried out for the Pywikibot tool.

[Documentation] *Clarity:* Pywikibot's documentation is detailed and well structured, although it can be confusing for beginners due to the number of MediaWiki and Wikidata commands/functions. *Complexity:* The documentation is primarily intended for experienced developers. It's not as friendly for users just starting with Python or Wikidata. *Coverage:* It is broad and covers most of the functions that Pywikibot offers, from editing pages to manipulating data in Wikidata. However, it can be difficult to find specific documentation on optimizing scripts or handling large volumes of data.

[Ease of use for data extraction] *Learning Curve:* Learning is complex, especially for those who are not familiar with Python or Wikidata. *Access to complex data:* Pywikibot allows access to complex data from Wikidata but requires advanced knowledge to create efficient scripts. *Compatibility:* Pywikibot supports Python and can integrate with other tools. However, it is often necessary to transform the data to adapt it to these tools.

[Data access speed] *Latency:* The speed of data access may vary depending on the complexity of the operations performed by Pywikibot. For simple tasks, latency is low. *Performance:* Pywikibot may



```
   "outputs": [],
   "source": [
    "https://github.com/wikimedia/pywikibot\n",
    "https://www.mediawiki.org/wiki/Manual:Pywikibot\n",
    "https://pypi.org/project/pywikibot/\n",
    "https://www.mediawiki.org/wiki/Manual:Pywikibot/es\n",
    "https://doc.wikimedia.org/pywikibot/master/"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "de858881-6d2e-4a47-bc66-326cc1ad8923",
   "metadata": {},
   "outputs": [],
   "source": []
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "id": "8ed76fab-568a-44ad-9f30-358d2a55b202",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "['Q9091606', 'P31', 'Q3918']\n",
      "['Q9091606', 'P17', 'Q750']\n",
      "['Q750', 'P17', 'Q750']\n",
      "['Q750', 'P361', 'Q12585']\n",
      "['Q3028249', 'P17', 'Q414']\n",
      "['Q3028249', 'P131', 'Q1486']\n",
      "['Q414', 'P17', 'Q414']\n",
      "['Q414', 'P361', 'Q12585']\n",
      "['Q1486', 'P31', 'Q5770918']\n",
```

Figure 4: Source code fragment showing queries to Wikidata using Pywikibot.

experience performance issues, due to limitations of MediaWiki servers. *Optimization:* There are not many specific optimization tools within Pywikibot.

[Ease of output conversion] Formats: Pywikibot can export data in different formats via Python, but it does not offer predefined formats. Developers must take care of the conversion of the obtained data, usually to JSON or CSV. *Transformation:* Data transformation depends on the developer's ability to write Python scripts. *Compatibility:* Data extracted by Pywikibot can be easily integrated into analysis tools.

[Interoperability] *Compatibility:* Pywikibot is compatible with the MediaWiki and Wikidata standards, allowing its integration with other services that use the same platforms. *Integration:* Pywikibot can integrate with other APIs and services that use MediaWiki or Wikidata. *Interoperability:* Pywikibot supports the standard Wikidata and MediaWiki protocols, making it easy to use in open data environments.

[Scalability]: *Limits* Pywikibot has application limits with the MediaWiki API, which can be an issue in applications that require a high volume of queries. *Scalability:* It is suitable for small and medium-sized projects. Large volumes: Pywikibot can handle large volumes of data, but performance may decrease.

[Support and community] *Technical Support:* Pywikibot is supported primarily through the user community. Official support is limited, but there are many community resources. There are many examples and tutorials online.

Figure 4 shows the development carried out by the authors of this work, which shows the fragment of a small prototype with the Pywikibot library to interact with Wikidata and perform queries that return RDF triples, which are made up of a subject, a predicate and an object. Pywikibot is a collection of Python tools that makes it easy to automate tasks on MediaWiki-based wikis, such as Wikidata. In this code, queries are run to get and display specific relationships between Wikidata entities, represented by their unique identifiers (such as 'Q9091606' and 'P31'). These relationships are printed to the console, showing the connections between different entities in the format of triples. The code is useful for performing automated tasks in Wikidata, such as extracting and analyzing structured data for specific research or applications.

[License and terms of use] *License:* Pywikibot is available under the MIT license, allowing flexible and free use of the code. *Restrictions:* There are no significant restrictions on the use of Pywikibot. Developers are free to modify and redistribute the code. *Obligations:* Users must comply with the terms of the MIT license but are free to use Pywikibot in commercial and noncommercial projects.

3.1.4. OpenRefine with the Wikidata plugin

To carry out an analysis of OpenRefine with the Wikidata plugin based on the points you mentioned, some sources were worked with [33, 34]. Below is the analysis carried out:

[Documentation] *Clarity and Complexity:* The OpenRefine documentation with the Wikidata plugin is quite accessible and designed for users of different experience levels. Tutorials and examples are provided to allow beginners to quickly familiarize themselves with the functionalities. *Coverage:* The documentation covers a wide range of functionality, from installing the plugin to running SPARQL queries. However, it may lack advanced details for users who wish to perform complex data transformations. *Examples of Use:* There are practical examples that cover basic cases and some more complex ones, but examples for more specialized scenarios may be lacking.

[Ease of use for data extraction] Learning Curve: OpenRefine is intuitive, especially for users who already have experience in data cleaning and transformation. The Wikidata plugin makes it easy to integrate data without requiring advanced SPARQL knowledge. *Ease of Access to Complex Data:* Although the use of the tool allows us to simplify data extraction, performing advanced queries is more complex. *Tool compatibility:* It is compatible with multiple data analysis tools and its integration with Wikidata allows easy access to large volumes of structured data.

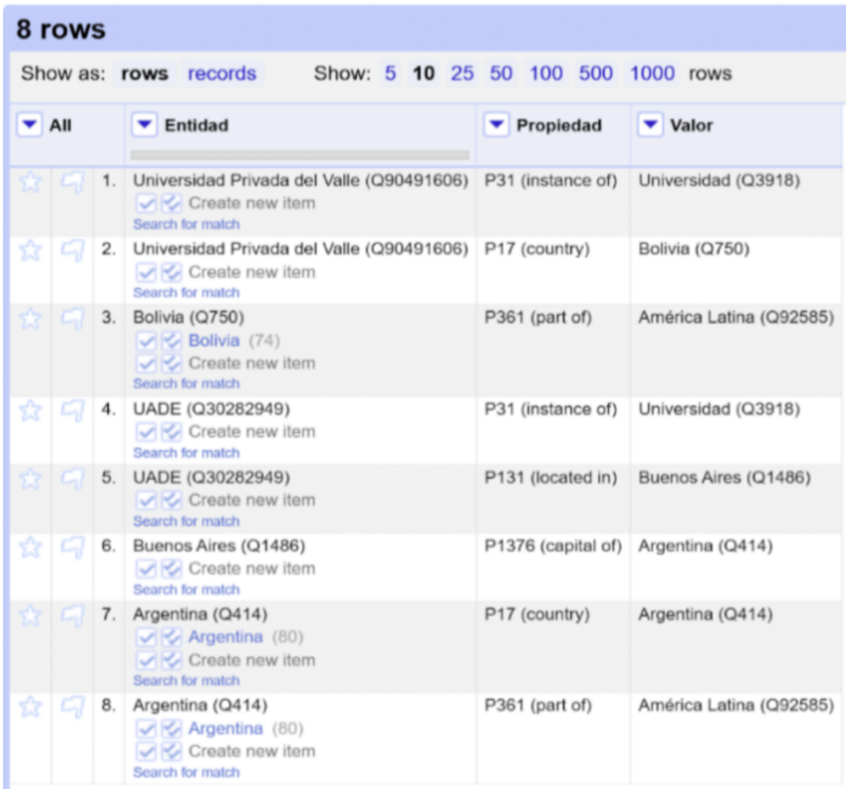
[Data access speed] *Average Latency:* Data access speed is good for simple queries. *Performance Under Load:* There may be slowdowns when performing multiple complex operations or when working with large volumes of data. *Optimization Mechanisms:* OpenRefine allows some optimizations to be performed on queries, but most of the performance depends on the complexity of the SPARQL query.

[Ease of output conversion] *Available Formats:* Extracted data can be exported in various formats, including CSV, Excel, JSON, and others, making it easy to use in various applications. *Ease of Transformation:* Data transformation is one of the strengths of OpenRefine, allowing personalized adjustments according to the user's needs. *Compatibility:* The export formats are widely compatible with other analysis and visualization tools.

[Interoperability] *Standards Support:* OpenRefine follows open standards and is compatible with a wide range of Linked Open Data (LOD) tools, especially the Wikidata plugin. *Integration APIs:* Integrates well with open data APIs, making it easy to combine data from multiple sources. *Interoperability Protocols:* Compatible with standard protocols, allowing its use in environments that require interoperability with other open data systems.

[Scalability] *Request Limits:* OpenRefine itself does not impose significant limits, but the load on the Wikidata API can be a limiting factor, especially if many requests are made simultaneously. *Scalability Management:* It is suitable for small and medium-sized projects but may not scale well without careful optimization for large volumes of data. *Support for Large Data Volumes:* Handles large data sets well, but performance may decrease under very high loads.

Figure 5 shows the OpenRefine interface, with the result of a data connection operation with Wikidata. You see a table with 8 rows, detailing different entities such as universities, countries, and cities, along with their properties (for example, "instance of" or "country") and corresponding values in Wikidata. On the left, you can see the filtering and facet options, where entity judgment facets and the best candidate score have been applied.


8 rows

Show as: rows records Show: 5 10 25 50 100 500 1000 rows

All	Entidad	Propiedad	Valor
1.	Universidad Privada del Valle (Q90491606) Create new item Search for match	P31 (instance of)	Universidad (Q3918)
2.	Universidad Privada del Valle (Q90491606) Create new item Search for match	P17 (country)	Bolivia (Q750)
3.	Bolivia (Q750) Bolivia (74) Create new item Search for match	P361 (part of)	América Latina (Q92585)
4.	UADE (Q30282949) Create new item Search for match	P31 (instance of)	Universidad (Q3918)
5.	UADE (Q30282949) Create new item Search for match	P131 (located in)	Buenos Aires (Q1486)
6.	Buenos Aires (Q1486) Create new item Search for match	P1376 (capital of)	Argentina (Q414)
7.	Argentina (Q414) Argentina (80) Create new item Search for match	P17 (country)	Argentina (Q414)
8.	Argentina (Q414) Argentina (80) Create new item Search for match	P361 (part of)	América Latina (Q92585)

Figure 5: Data reconciliation view in OpenRefine with Wikidata plugin.

[Support and community] *Technical Support Availability:* OpenRefine has an active community, and support for the Wikidata plugin is primarily provided by users and developers in the community. *Community Resources:* There are a good number of resources, including forums, mailing lists, and guides created by the community. *Documentation Updates:* Documentation is updated regularly, although it may depend on community participation to stay up to date.

[License and terms of use] *License Type:* OpenRefine is open source, allowing its use and modification without significant restrictions. The Wikidata plugin is also under a permissive license. *Use Restrictions:* There are no significant restrictions on its use. *Rights and Obligations:* Users can use and modify the software under the Open Source license, with the obligation to follow the terms of the corresponding license (generally GPL or similar).

3.2. Analysis

This work presented the analysis of four tools for the use of open data linked through a proposed data model.

The analysis of these tools was based on: a) Wikidata API, b) the SPARQL query service using Wikidata Query Service (WDQS), c) Pywikibot, and d) OpenRefine with its Wikidata plugin. From the Wikidata API point of view, this is an interesting option for integrating simple data queries, as it offers an easy-to-use RESTful interface for programmers with experience in this type of API. However, you may be limited when performing more complex queries. On the other hand, it was observed that SPARQL, through the Wikidata Query Service, is a very useful tool for performing complex queries on interrelated data in Wikidata. Although its use requires a good knowledge of RDF and the Wikidata data structure, this tool allows you to analyze data with a very good level of precision in its results. The next tool analyzed is Pywikibot, it works with Python that offers great flexibility to automate tasks in Wikidata. It is ideal for handling large volumes of data. However, its use requires a high level of knowledge in programming and Wikidata structure. Finally, OpenRefine with the Wikidata plugin is presented as a tool that presents great advantages in data cleaning techniques, integrating data sets linked with Wikidata. This tool is especially useful for those looking to improve the quality of their data before incorporating it into deeper analysis. However, its learning curve can be steep.

In terms of documentation and ease of use, all four tools feature extensive documentation, although some stand out more than others. The Wikidata API and SPARQL are well documented, but their level is complex for inexperienced users. Pywikibot, although it presents documentation, requires advanced programming knowledge. OpenRefine, with its graphical interface, is more accessible for data cleaning tasks, but its Wikidata plugin may require additional learning.

The learning curve of these tools is very varied. The Wikidata API is relatively easy to use for those with experience in RESTful APIs, while SPARQL requires in-depth knowledge of its programming syntax, and in dealing with the data structure in Wikidata. On the other hand, Pywikibot features a complex level of programming, and OpenRefine, while generally accessible and friendly, can be complex to use when used in conjunction with the Wikidata plugin for advanced tasks. In terms of performance, the Wikidata API and SPARQL offer fast responses for simple queries, although they may face limitations on complex queries. Furthermore, Pywikibot has problems in operational performance aspects, since its efficiency depends on the optimization of the source code. OpenRefine is very efficient for data cleaning, but its performance may decrease with large volumes of data.

In terms of interoperability and flexibility, SPARQL and Pywikibot stand out for their high compatibility with other Linked Open Data (LOD) systems and standards. The Wikidata API easily integrates with other tools using RESTful calls, and OpenRefine, through its Wikidata plugin, provides a robust way to link external data with entries in Wikidata. Community support for these tools is robust, facilitating problem-solving and knowledge sharing. However, for users who require quick or specialized technical support, relying exclusively on community support can be a challenge, especially in the case of SPARQL, Pywikibot, and the OpenRefine plugin, where the technical complexity is greater. All tools operate under the Creative Commons CC0 1.0 license, allowing free use of data for academic and commercial projects. However, such a permissive license could raise ethical dilemmas related to data attribution.

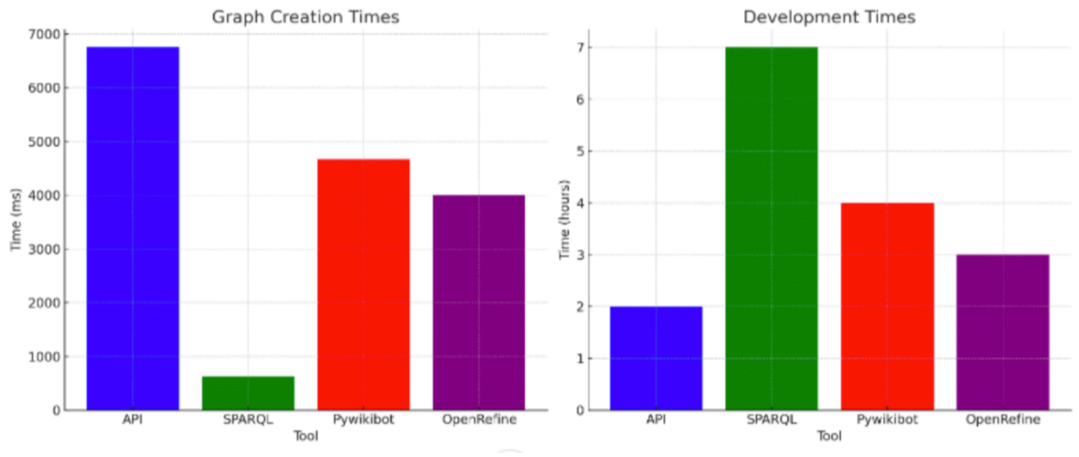


Figure 6: Comparison of Graph Creation and Development Times Across Different Tools.

As for a quantitative analysis, estimates and measurements were made with the tools worked on. Figure 6 shows a comparative graph of the graph creation times and development times for the different tools. For the graph on the left, the graph creation times are shown in milliseconds, where OpenRefine is estimated at an average of 4000 ms, being faster than Pywikibot (4665.79 ms) but slower than SPARQL (632.03 ms). The Wikidata API (6747.69 ms) is also located at an average time, which places it as a slower option, close to OpenRefine, but still slower than SPARQL. For the graph on the right, development times are shown in hours, with OpenRefine estimated at 3 hours and the Wikidata API at 2 hours.

Based on the previous analysis, the OpenRefine tool is presented as an intermediate option between the API and SPARQL (7 hours), and the Wikidata API as a more efficient option in aspects of software development, being similar to the time of the standard API. This graph visualizes how the tools compare in terms of performance and software development time, which can help you decide which is the best tool to implement based on the needs at hand.

4. Conclusions and future work

This analysis presents the importance of selecting an appropriate tool for the management of linked open data, considering the capabilities and limitations of each of these, depending on the objectives and available resources. As another option, it could be considered that the combination of the use of these tools could benefit projects that involve the manipulation and analysis of data in Wikidata. Regarding recommendations, the choice of the appropriate tool should be based on the specific needs of the project and the level of technical experience of the work team that uses them. This study has demonstrated significant differences in performance and programming development time in terms of the use of the tools. One of the points that was considered was the creation of graphs using linked open data, in which SPARQL is the most efficient tool to create these graphs, although this presents a longer design time. On the other hand, OpenRefine and the Wikidata API offer an interesting combination in terms of development speed and execution efficiency, making them efficient tools for projects with limited resources. This work shows a detailed comparative analysis, to facilitate decision-making for the best possible tool for the use of linked open data.

As for future lines of research, it would be valuable to work on optimizing development time in SPARQL to make it more accessible to users with less technical experience. In addition, the integration of artificial intelligence could be investigated in order to improve aspects of the quality and treatment of data linked to metadata.

References

[1] World Wide Web Consortium (W3C), W3c, 2024. https://www.w3.org/.
[2] E. Ávila Barrientos, Linked data and its use in libraries, 2020. http://ru.iibi.unam.mx/jspui/handle/IIBI_UNAM/56.
[3] World Wide Web Consortium (W3C), Linked open data (lod), 2010. https://www.w3.org/egov/wiki/Linked_Open_Data.
[4] A. O. Sierra, Insertion of metadata from spanish libraries in wikidata: a linked open data model, REVISTA ESPANOLA DE DOCUMENTACION CIENTIFICA 45 (2022).
[5] B. Hyland, G. Atemezing, B. Villazon-Terrazas, Best practices for publishing linked data-w3c working group note 09 january 2014, 2014.
[6] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities, 2023.
[7] Wikimedia Foundation, Wikidata: Main page, 2023. https://www.wikidata.org/wiki/Wikidata:Main_Page.
[8] Wikimedia Foundation, Wikidata: Linked open data work-flow, 2021. https://www.wikidata.org/w/index.php?title=Wikidata:Linked_open_data_work-flow&oldid=1394378153.
[9] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, D. Vrandečić, Introducing wikidata to the linked data web, in: The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13, Springer, 2014, pp. 50–65.
[10] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.
[11] I. Nishanbaev, E. Champion, D. A. McMeekin, A web gis-based integration of 3d digital models with linked open data for cultural heritage exploration, ISPRS international journal of geo-information 10 (2021) 684.

[12] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al., Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia, Semantic web 6 (2015) 167–195.

[13] A. Chessa, G. Fenu, E. Motta, F. Osborne, D. R. Recupero, A. Salatino, L. Secchi, Data-driven methodology for knowledge graph generation within the tourism domain, IEEE Access (2023).

[14] M. B. Brandt, S. A. Borsetti Gregorio Vidotti, J. E. Santarem Segundo, Legislative linked open data: a proposal of linked open data modelling for legislative information, 2018.

[15] D. Diefenbach, P. H. Migliatti, O. Qawasmeh, V. Lully, K. Singh, P. Maret, Qanswer: A question answering prototype bridging the gap between a considerable part of the lod cloud and end-users, in: The World Wide Web Conference, 2019, pp. 3507–3510.

[16] A. Uzun, A. Uzun, Semantic enrichment of mobile and wifi network data, Semantic Modeling and Enrichment of Mobile and WiFi Network Data (2019) 59–96.

[17] M. Martinez, Quality metrics to validate government public open data sets, 2022.

[18] K. Thornton, K. Seals-Nutt, M. Van Remoortel, J. M. Birkholz, P. De Potter, Linking women editors of periodicals to the wikidata knowledge graph, Semantic Web 14 (2023) 443–455.

[19] H. Chalupsky, P. Szekely, F. Ilievski, D. Garijo, K. Shenoy, Creating and querying personalized versions of wikidata on a laptop, arXiv preprint arXiv:2108.07119 (2021).

[20] S. L. Freshour, S. Kiwala, K. C. Cotto, A. C. Coffman, J. F. McMichael, J. J. Song, M. Griffith, O. L. Griffith, A. H. Wagner, Integration of the drug–gene interaction database (dgidb 4.0) with open crowdsource efforts, Nucleic acids research 49 (2021) D1144–D1151.

[21] A. M. F. García, M. I. M. García, R. B. Gallinas, M. M. Sánchez, M. E. B. Gutiérrez, Integration and open access system based on semantic technologies: A use case applied to university research facet, International Journal on Semantic Web and Information Systems (IJSWIS) 18 (2022) 1–19.

[22] A. Vinoth, S. Thangasamy, R. Nithya, C. Subalalitha, P. Mahendhiran, The impact of tamil python programming in wikisource, in: International Conference on Speech and Language Technologies for Low-resource Languages, Springer, 2023, pp. 79–90.

[23] S. Deng, G. Heng, A. Xu, L. Zhu, X. Li, Enhance the discovery and interoperability of culturally rich information: the chinese women poets wikiproject, 2022.

[24] S. C. Schmidt, F. Thiery, M. Trognitz, Practices of linked open data in archaeology and their realisation in wikidata, Digital 2 (2022) 333–364.

[25] L. Álvarez-Azcárraga, Radical openness and free knowledge: Repository of open-access mexican academic journals through wikidata, Revista científica (2023) 27–39.

[26] Wikidata, Wikidata:rest api, 2024. https://www.wikidata.org/wiki/Wikidata:REST_API.

[27] Wikidata, Wikidata:stable interface policy, 2024. https://www.wikidata.org/wiki/Wikidata:Stable_Interface_Policy.

[28] Wikidata, Wikidata:sparql query service/queries, 2024. https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/queries.

[29] Wikidata, Wikidata:sparql query service/help, 2024. https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/Wikidata_Query_Help/es.

[30] Wikidata, Wikidata:sparql tutorial, 2024. https://www.wikidata.org/wiki/Wikidata:SPARQL_tutorial/es.

[31] Wikidata, Wikidata:pywikibot - python 3 tutorial, 2024. https://www.wikidata.org/wiki/Wikidata:Pywikibot_-_Python_3_Tutorial.

[32] Wikidata, Manual:pywikibot/wikidata, 2024. https://www.mediawiki.org/wiki/Manual:Pywikibot/Wikidata.

[33] Wikidata, Wikidata:tools/openrefine/editing, 2024. https://www.wikidata.org/wiki/Wikidata:Tools/OpenRefine/Editing.

[34] OpenRefine, Overview of wikibase support, 2024. https://openrefine.org/docs/manual/wikibase/overview.