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

Roxana Martínez and Gonzalo Pereyra Metnik

Universidad Argentina de la Empresa (UADE), Lima 757, Ciudad Autónoma de Buenos Aires, C1073, Argentina

Abstract
In this study, a comparative analysis of various tools used for the integration of Linked Open Data is carried out, to evaluate their performance and suitability in different use scenarios. The increasing availability of open data and its efficient integration into applications and services represents a significant challenge in the field of data science and information technology.
An introduction to the topic is presented, highlighting the importance of Linked Open Data and the need for tools that facilitate its integration and manipulation. Next, an analysis of related works that have addressed the topic is carried out, identifying previous studies that have evaluated similar tools.
The objective of this study focuses on a proposed data model, which is used for testing with five selected tools: 1) API, 2) SPARQL in Wikidata Query Service (WDQS), 3) Pywikibot, and 4) OpenRefine with the Wikidata plugin. Each of these tools is evaluated 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 the strengths and weaknesses of each one to be identified in relation to the proposed data model. The study's findings provide recommendations for tool selection based on specific data integration requirements in future projects, as well as suggestions for improvement of these tools.

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

1. Introduction

In the context of the semantic web, Linked Open Data (LOD) has become a key element for the development of advanced applications and services. Effective integration of this data offers an important opportunity to improve interoperability and accessibility of information. By allowing connections between different data sets, Linked Open Data facilitates richer and more meaningful use of information. However, the increasing availability of this data presents complex challenges related to its integration and management, highlighting the need for appropriate tools that can address these challenges effectively.

1.1. Context

In recent years, the concept of Linked Open Data is booming, as it is a key methodology for publishing and accessing data on the web in a structured and connected way. 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 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). Furthermore, it suggests adding links to other URIs and using an RDF triplet query language, such as SPARQL, which is recommended by the W3C*" [4]. These recommendations and others are found in the international publication of Good Practices for Linked Open Data [5]. Many authors agree on the importance of this new trend, for example, according to Berners-Lee [6], LOD is based on the principles of the semantic web, using technologies such as RDF and URIs to establish explicit connections between different data sets, thus facilitating their interoperability and reuse. This approach allows data, often published by multiple organizations around the world, to be queried and combined efficiently, generating new possibilities for the analysis of large volumes of information.

Within this same context, there are different Linked Open Data repositories, one of the most relevant is Wikidata [7]. This centralized repository allows access to other users, similar to the wikis managed by the Wikimedia Foundation. Another aspect of this paradigm is that it is not necessary to dynamically update the content on each wiki individually, since there are common data such as statistics, dates and locations that can be centralized in Wikidata. On its official website, enter other points, it also presents information on the most relevant considerations for data processing, scripts, tools, media and other resources to consider for the flow of Linked Open Data works [8] . Wikidata stands out for its focus on interoperability and its ability to connect data from various domains. This ability to efficiently link information makes it a key tool for researchers, developers, and organizations that want to take advantage of Linked Open Data. By enabling integration and access to structured data from multiple sources, Wikidata makes it easy to create advanced applications and conduct in-depth research, thereby driving progress in numerous fields of knowledge and technology.

On the one hand, Wikidata's flexible structure and extensive thematic coverage facilitate the integration of data from multiple sources, thus creating an environment rich in semantic connections that enriches interoperability between different data sets [9]. This allows users to extract significant value from interconnected information. A fundamental aspect in this area is that managing data heterogeneity and maintaining the quality of linked information present considerable challenges. These challenges require the use of specialized tools and techniques to ensure integrations are both accurate and functional. Implementing these techniques is crucial to ensure that linked data maintains its integrity and usefulness as it is combined and updated [10]. In this context, Wikidata's ability to handle these aspects is fundamental to its success as a collaborative and accessible knowledge base.

1.2. State of the art

Over the years, various solutions and tools have been developed for the integration of Linked Open Data. These tools vary in their focus and capabilities, ranging from APIs and SPARQL query services to specialized libraries and software plugins. 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 there are several repositories and projects that 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, 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 difficulties can limit users' ability to extract and use information consistently and effectively. For this reason, it is essential to have tools and technologies that not only facilitate the integration of data from multiple domains, but also ensure the quality and consistency of the linked information. The need for effective solutions becomes more urgent as open data continues to grow in quantity and variety, making proper management and efficient integration essential to make the most of these resources. For this reason, it is important to find well-designed and used tools that can significantly improve the ability to manage this data effectively, allowing for richer and more meaningful use that encourages the development of innovative applications and data-driven services.

The objective of this study is to carry out a detailed comparative analysis of various tools used for the integration of Linked Open Data. The study seeks to evaluate the performance of these tools in different use scenarios by implementing a data model proposed by the authors. In particular, four selected tools that present great potential in aspects of Wikidata will be examined: 1) the Wikidata API; 2) the SPARQL in Wikidata Query Service (WDQS); 3) Pywikibot; and 4) OpenRefine with its Wikidata plugin. Apart from raising awareness about the use of LOD, the main purpose is to 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 evaluation, the study aims to provide practical recommendations for tool selection based on specific data integration requirements in future projects and offer suggestions for the improvement of the analyzed tools.

2. Methodology

In order to carry out an analysis of the tools selected for the integration of Linked Open Data (LOD), a methodology has been developed that includes the selection of the tools, the detailed presentation of each one, the definition of the evaluation metrics and the implementation of a systematic testing procedure. This methodology is designed to ensure that each tool is evaluated impartially and under uniform conditions, thus allowing a fair and meaningful comparison of its functionalities and limitations. Next, the criteria used to select the tools are explained, followed by an introduction to each one, the definition of the key metrics for their evaluation, and a detailed description of the testing procedure carried out in this study.

2.1. Tool Selection Criteria

Wikidata has established itself as one of the most important sources of structured data on the web, driving the development of a variety of technologies designed to facilitate its access and manipulation. Among these technologies, the APIs that allow the direct query and manipulation of linked data, SPARQL query services, such as the Wikidata Query Service (WDQS), and automation tools such as Pywikibot stand out. In addition, specialized libraries such as Wikidata Toolkit (WDTK) offer robust solutions for handling large volumes of data, while applications such as OpenRefine, with its Wikidata plugin, allow data refinement and cleaning accessible even to users without programming experience.

However, this research will analyze four technological solution tools aimed at Wikidata: 1) the Wikidata API; 2) the SPARQL in Wikidata Query Service; 3) Pywikibot; and 4) OpenRefine with its Wikidata plugin. These tools have been selected for their relevance and applicability in the integration of Linked Open Data (LOD), their ease of use, and their ability to efficiently handle data manipulation tasks in contexts where large volumes of data are not required. The SPARQL API and Query Service are central to data extraction and manipulation in Wikidata, Pywikibot excels at automating repetitive tasks, and OpenRefine provides a powerful platform for data cleansing and transformation, ensuring quality and consistency in the linked information. For the latter, it also allows maintaining ease of use and flexibility, which are fundamental characteristics for users, even without programming experience, to interact with Wikidata effectively.

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 technical aspects for 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. There are works that 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. [21]. 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 [22].

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, overview of PAWS (Python Data Science Environment for Jupyter) and PyWikiBot [23]. 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 [24].

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. There are works that show several preparation techniques for processing directly with QuickStatements, allowing editing and creating Wikidata elements directly from OpenRefine [25]. Another study proposes generating a repository with an open database for academic publications using Wikidata with OpenRefine technology [26].

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, 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 data recovery in formats compatible with various applications and supports data reading and modification operations, promoting interoperability in the semantic web. Pywikibot is a Python library designed to automate interactions with MediaWiki, including Wikidata. It also allows the creation of scripts to perform repetitive tasks such as mass editing of pages, updating elements, and data management, facilitating the maintenance and enrichment of Wikidata content through a programmatic approach. Finally, OpenRefine with Wikidata Plugin is a data cleaning and transformation tool, it can be extended with a Wikidata plugin that allows you to connect, reconcile and enrich local data with information from Wikidata. This plugin makes it easy to identify entities, add properties, and update data, improving the quality and consistency of datasets by aligning them with Wikidata standards.

2.3. Evaluation Metrics

This section presents the evaluation metrics used to analyze and compare the different tools and technologies in terms of their performance and ability to handle linked open data. Key criteria have been selected for comprehensive evaluation, including ease of use for data extraction, which measures how intuitive and accessible the tool is to users; the speed of data access, which evaluates the response time and efficiency in information retrieval; and ease of output conversion, which analyzes the tools' ability to transform data into different required formats. Additionally, interoperability is considered, which examines the ability of tools to interact and be compatible with other systems and standards; scalability, which evaluates how tools handle a growing volume of data and users; support and community, which assesses the availability of help resources and the activity of the community of users and developers; and finally, the license and terms of use, which reviews the restrictions and permissions associated with the use of the evaluated tools and technologies. These metrics offer a complete perspective to assess the suitability of solutions in different scenarios and needs.

2.4. Proposed LOD model

The proposed data model is shown in Figure 1. The image shows the relationship between different educational and geographic entities using the Wikidata ontology and properties. 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 is 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).

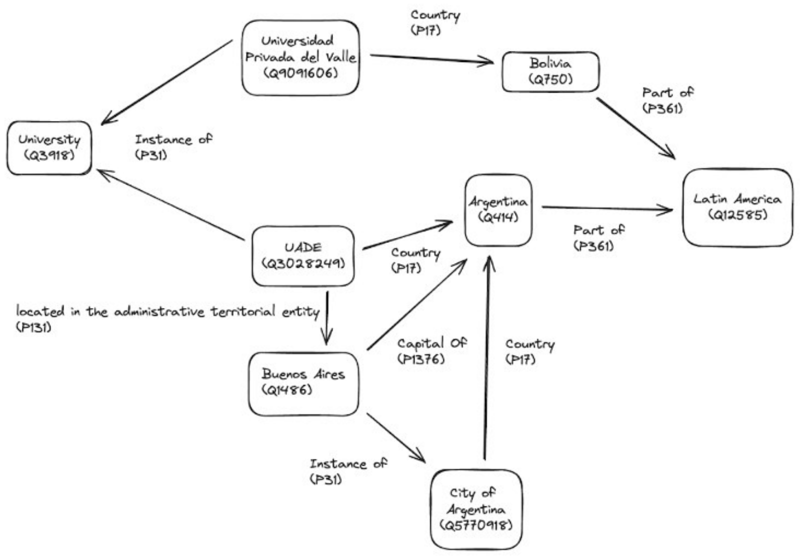


Figure 1: Proposed data model.

2.5. Testing Procedure

With the proposed linked open data model and the tools selected as 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 tools' ability to handle output conversions to different formats was observed. Interoperability was then assessed by integrating the tools with other systems, and scalability was tested by running the tests. Finally, observations on the support and community of each tool were collected in general Internet forum aspects, as well as the license terms were reviewed to ensure the legality of its use in various contexts.

3. Results

In this section, the findings obtained during the evaluation of the selected tools are presented and analyzed. A methodological approach has been carried out that combines quantitative and qualitative analysis to provide a comprehensive view of the capabilities and limitations of each tool under study. The results are detailed in parts: Comparison of tools, qualitative analysis and quantitative Analysis.

3.1. Comparison of Tools

3.1.1. Wikidata API

For the analysis of the Wikidata API, various techniques were used that offer specific advantages depending on their application. The Wikidata REST API [27] provides a simple and efficient interface for performing quick queries and obtaining structured data from Wikidata, which is ideal for developers looking to agilely integrate data into applications. The Wikibase REST API documentation in JavaScript helped facilitate the implementation of advanced functionalities in web applications, thanks to its integration with JavaScript technology, widely used in web development. The Wikibase 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 [28] 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. Advanced concepts like RDF and JSON-LD are not sufficiently explained for less technical users. *Examples of Use:* The examples provided are useful but limited to basic cases, without covering more complex scenarios. *Coverage:* Coverage is broad and details all endpoints, but guidance on query optimization is lacking. The documentation is not in one place, as it is distributed in several sources.

[Ease of use for data extraction]: *Learning Curve:* The API is intuitive for developers familiar with RESTful APIs but may be difficult for users without prior experience in Wikidata or RDF. There are no graphical interfaces to facilitate its use. *Ease of Access to Complex Data:* The API allows access to complex data through parameter combinations but requires a good understanding of the Wikidata structure to do so effectively. *Tool Compatibility:* It is compatible with data analysis tools such as Python, but the integration is not always direct and may require data transformation.

[Data access speed] *Average Latency:* The Wikidata API generally offers fast response times for simple queries. *Performance Under Load:* The API may experience slowdowns under heavy loads or 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. *Optimization Mechanisms:* The API lacks clear tools for query optimization, leaving this task to the developer.

[Ease of output conversion] *Available Formats:* The API allows obtaining data in JSON, RDF, and XML formats, which provides flexibility in its use in different applications. *Ease of Transformation:* JSON is the most versatile and easiest format to transform, but converting from RDF may require specialized knowledge. *Compatibility:* Supported formats are widely compatible with data analysis and visualization tools, although RDF transformation can be complicated without the right tools.

[Interoperability] *Compatibility with Standards:* The API follows open standards such as RDF and JSON-LD, which facilitates its integration with other Linked Open Data (LOD) systems. *Integration APIs:* It is compatible with other APIs and LOD services, allowing you to combine data from multiple sources for a more complete analysis. *Interoperability Protocols:* The API supports protocols that allow interoperability with other LOD platforms, although this may require specialized knowledge in some cases.

[Escalabilidad] *Límites de Solicitudes:* La API impone límites sobre el número de solicitudes por segundo para evitar sobrecargas, lo que puede ser un problema en aplicaciones de gran escala. *Gestión de Escalabilidad:* La API es adecuada para aplicaciones pequeñas a medianas, pero puede no escalar bien para grandes volúmenes de solicitudes simultáneas sin optimizaciones cuidadosas. *Soporte para Grandes Volúmenes de Datos:* Puede manejar grandes volúmenes de datos, pero con un rendimiento que decrece bajo altas demandas concurrentes.

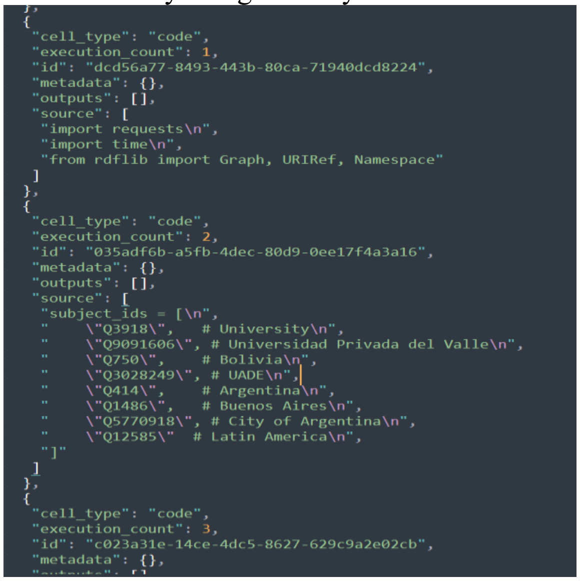
[Support and community] *Technical Support Availability:* Wikidata has adequate technical support, but the majority of support comes from the user community. *Community Resources:* There are active forums, mailing lists, and community resources that are valuable for resolving common questions and problems. *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 free and 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 non-commercial 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. Then, a series of identifiers (subject_ids) are defined 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 visual 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 code has the purpose of creating and manipulating a graph that reflects these connections, allowing subsequent queries and analysis. This approach is valuable for organizing and linking information about educational entities and their geographic and political context in a structured and easily navigable 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 snippet for interacting with the Wikidata API and manipulating RDF data.

3.1.2. SPARQL in Wikidata Query Service (WDQS)

For the analysis of SPARQL in Wikidata Query Service (WDQS), various techniques were used that offer specific advantages depending on their application [29]. The code snippet that constructs an RDF graph from Wikidata entities and properties is useful in the context of SPARQL queries in Wikidata Query Services [30] 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 [31] 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 extremely powerful for accessing interrelated and complex data. It allows you to perform sophisticated queries that combine multiple entities and properties, something that would be difficult to do with other methods. *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.

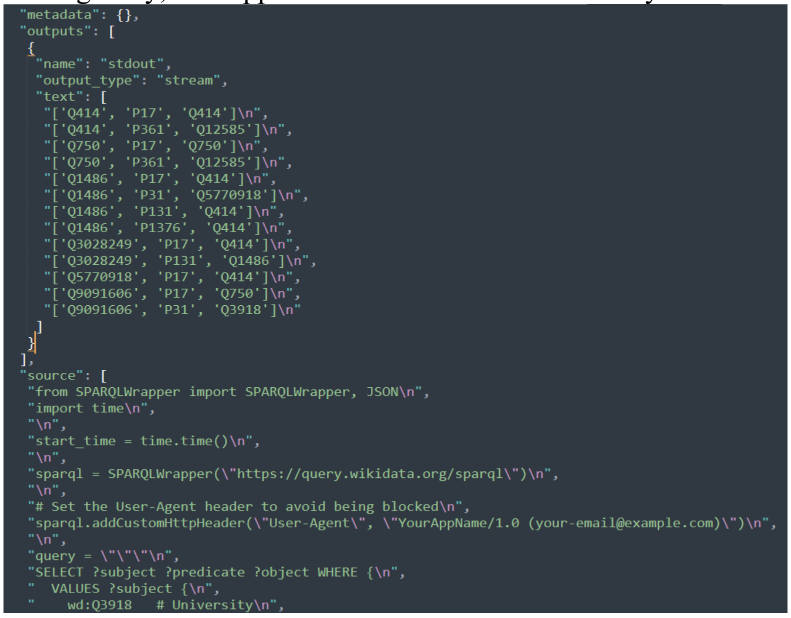
[Data access speed] *Average Latency:* Response time is generally fast for simple queries. However, latency can increase significantly with complex queries involving many relationships or large volumes of data. *Performance Under Load:* WDQS may experience performance issues under heavy load or when performing very complex queries. The service sometimes imposes limits on the queries that can be performed to prevent system overload. *Optimization Mechanisms:* The documentation includes suggestions for optimizing queries, but optimization requires in-depth knowledge of both SPARQL and the structure of the data in Wikidata, which may not be trivial for all users.

[Ease of output conversion] *Available Formats:* SPARQL query results can be obtained in a variety of formats, including JSON, CSV, XML, and 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 the results into other analysis tools. However, transforming complex SPARQL results into simpler structures can be challenging without additional tools. *Compatibility:* Supported formats are widely compatible with common data analysis tools, although conversion and manipulation of more complex formats such as RDF may require specialized knowledge.

[Interoperability] *Standards Compatibility*: SPARQL is a widely accepted standard for queries on RDF data, ensuring its compatibility with a wide range of Linked Open Data (LOD) systems and services. *Integration APIs:* WDQS can be easily integrated with other services and APIs that support RDF and SPARQL, facilitating data exchange between different platforms. *Interoperability Protocols:* WDQS supports standard interoperability protocols, making it easy to use in open data ecosystems, although effective implementation may require specialized knowledge.

[Scalability] *Request Limits:* The service imposes limits on the size and complexity of queries to avoid overloading, which can be an inconvenience in projects that require very large or complex queries. *Scalability Management:* WDQS is suitable for small and medium-scale queries, but scalability can be an issue for projects that require processing large volumes of data or performing intensive concurrent queries. *Support for Large Volumes of Data:* Although it is possible to handle large volumes of data, performance can be significantly affected, and larger queries may be rejected by the system.

[Support and community] *Technical Support Availability:* Wikidata provides adequate technical support through its 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 regularly, but support can be slow and relies heavily on community participation.



```
"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.

[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 free and 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 non-commercial 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.

3.1.3. Pywikibot

Some sources are used as a study to carry out the test with a developed prototype [32] [33]. 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 commands and the need to know the basics of MediaWiki and Wikidata. *Complexity*: The documentation is primarily aimed at experienced developers. It's not as friendly for novice users, and the learning curve can be high if you're not familiar with Python or Wikidata. *Examples*: The Pywikibot documentation includes basic and advanced examples but may lack specific examples for complex use cases. The community often complements this with tutorials and examples on forums and blogs. *Coverage*: The coverage is broad, covering most of the features 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*: Pywikibot has a steep learning curve, especially for those unfamiliar with Python or Wikidata. Requires a basic understanding of the MediaWiki API and how bots work in general. *Access to Complex Data*: Pywikibot allows access to complex data from Wikidata, but creating efficient scripts to obtain this data may require advanced knowledge of the structure of Wikidata and programming in general. *Compatibility*: Pywikibot is highly compatible with Python and can integrate with other data analysis libraries such as Pandas. However, it is often necessary to transform data to adapt it to these tools.

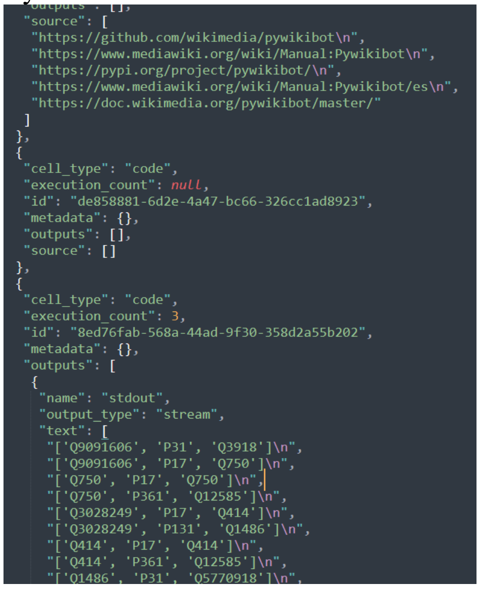
[Data access speed] *Latency*: The speed of data access may vary depending on the complexity of the operations performed by Pywikibot. For simple tasks, latency is low, but more complex queries or bulk edits can slow down the process. *Performance:* Pywikibot may experience performance issues if used to make a large number of edits or simultaneous requests to Wikidata, due to limitations imposed by MediaWiki servers. *Optimization:* There are not many specific optimization tools within Pywikibot; The efficiency of the script depends mainly on how the code is written and the careful handling of requests to the Wikidata API.

[Ease of output conversion] *Formats:* Pywikibot can export data in different formats through Python, but it does not offer predefined formats. Developers must handle the conversion of the obtained data, usually to JSON or CSV. *Transformation*: Data transformation depends on the developer's ability to write Python scripts that extract and format the data. *Compatibility*: Data extracted by Pywikibot can be easily integrated into analysis tools, as long as the script is designed to export in compatible formats.

[Interoperability] *Compatibility*: Pywikibot is compatible with 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, but this usually requires custom scripts. *Interoperability:* Pywikibot supports the standard Wikidata and MediaWiki protocols, making it easy to use in open data environments, although advanced knowledge is often required for effective implementation.

[Scalability] *Limits*: Pywikibot is subject to the same request limits as the MediaWiki API, which can be an issue in applications that require a high volume of edits or queries. *Scalability*: It is suitable for small and medium-sized projects but may not scale well without careful optimization when handling large volumes of data or requests. *Large Volumes*: Pywikibot can handle large volumes of data, but performance may decrease under high demand, especially if many edits are made in a short period of time.

[Support and community] *Technical Support:* Pywikibot is supported mainly through the community. Official support is limited, but there are many community resources. *Resources*: The Pywikibot community is active, with forums and mailing lists that are useful for solving common problems. There are many examples and tutorials online created by the community. *Documentation*: Community documentation and resources are updated regularly, although speed of response may depend on contributor availability.



```
 "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.

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 non-commercial projects.

3.1.4. OpenRefine with the Wikidata plugin

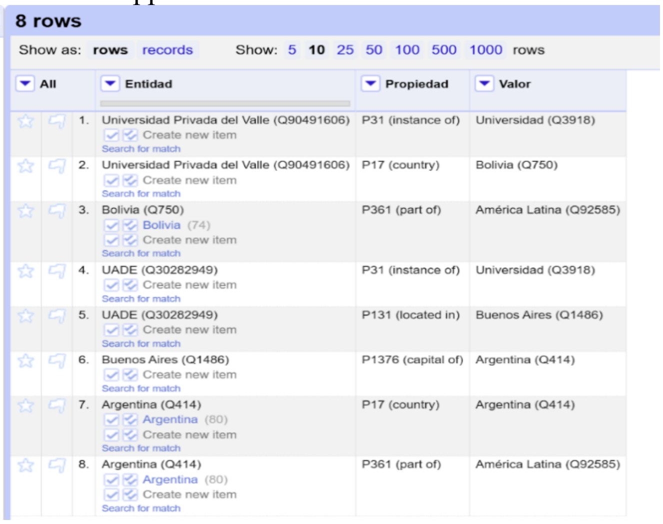
To carry out an analysis of OpenRefine with the Wikidata plugin based on the points you mentioned, some sources were worked with [34] [35]. Below is the analysis carried out:

[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 from this database without requiring in-depth knowledge of SPARQL. *Ease of Access to Complex Data:* Although the tool simplifies data extraction, complexity increases when you want to

obtain very specific data or perform advanced queries. *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 or moderately complex 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, allowing its use in both personal and commercial

projects. *Rights and Obligations*: Users are free to 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 study evaluated four key tools for data extraction and manipulation from Wikidata: the Wikidata API, the SPARQL query service using Wikidata Query Service (WDQS), Pywikibot, and OpenRefine with its Wikidata plugin. From the Wikidata API point of view, it is a valuable option for direct integration into applications through simple queries, offering a RESTful interface that is easy to adopt for developers with experience in this type of APIs. However, it can be limited when it comes to performing more complex queries or managing large volumes of data. On the other hand, SPARQL, through the Wikidata Query Service, was observed to be a powerful tool for performing detailed and complex queries on interrelated data in Wikidata. Although its use requires a good understanding of RDF and the Wikidata data structure, it allows you to extract and analyze data with a level of precision and detail that other tools do not offer. The next tool analyzed is Pywikibot, which is a Python library 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 the structure of Wikidata, which can be an obstacle for less experienced users. Finally, OpenRefine with the Wikidata plugin presents itself as a powerful tool for data cleaning, integrating capabilities to enrich data sets by comparing and linking with Wikidata entries. This tool is especially useful for those looking to normalize or improve the quality of their data before incorporating it into deeper analysis. However, its learning curve can be steep, especially for users who are not familiar with data reconciliation operations.

In terms of documentation and ease of use, all four tools have extensive documentation, although with varying degrees of accessibility. The Wikidata API and SPARQL are well documented but can be challenging for users without prior experience. Pywikibot, although detailed in its documentation, requires advanced programming knowledge. OpenRefine, with its graphical interface, is more accessible for cleaning and reconciliation tasks, but its Wikidata plugin may require additional learning.

The learning curve of these tools is diverse. The Wikidata API is relatively easy to use for those with experience in RESTful APIs, while SPARQL requires a deep understanding of its syntax and the data structure in Wikidata. Pywikibot, being based on Python, requires considerable programming understanding, and OpenRefine, while generally accessible, can be complex to master when used in conjunction with the Wikidata plugin for advanced reconciliation tasks. In terms of performance, the Wikidata API and SPARQL offer fast responses for simple queries, although they may face limitations under heavy loads or in complex queries. Pywikibot may experience performance issues in large-scale operations, its efficiency being dependent on code optimization. OpenRefine is very efficient for data cleaning and reconciliation, but its performance can decrease when dealing with very large data sets.

Regarding interoperability and flexibility, SPARQL and Pywikibot stand out for their high compatibility with other Linked Open Data (LOD) systems and standards. The Wikidata API, although more limited, integrates easily 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.

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. This positions OpenRefine as an intermediate option between the API and SPARQL (7 hours), and the Wikidata API as a more efficient option in terms of development, similar to the time of the standard API. This chart visualizes how the tools compare in terms of performance and development time, which can help you decide which is best for your specific needs.

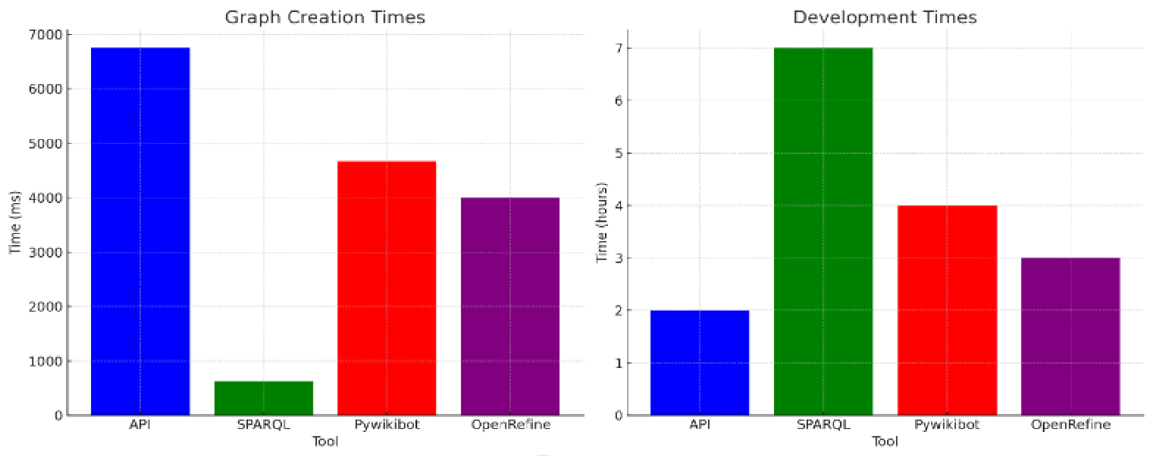


Figure 6: Comparison of Graph Creation and Development Times Across Different Tools.

4. Conclusions and future work

In conclusion, this analysis highlights the importance of selecting the appropriate tool for data management in Wikidata, considering the capabilities and limitations of each one based on the objectives and available resources. The strategic combination of these tools can maximize efficiency and effectiveness in projects that involve the manipulation and analysis of data in Wikidata. As for recommendations, choosing the right tool should be based on the specific needs of the project and the level of technical expertise of the team. This study has revealed significant differences in performance and development time between the evaluated tools for creating graphs using linked open data. SPARQL has been shown to be the most efficient tool for creating graphs, although at the cost of longer development time. On the other hand, OpenRefine and the Wikidata API offer an attractive balance between speed of development and efficiency of execution, making them viable options for resource-constrained projects. This study contributes to the field by providing a detailed comparative analysis, facilitating informed decision making about which tool to use for linked open data integration in specific contexts.

As for future lines of research, it would be valuable to explore optimizing development time in SPARQL to make it more accessible to users with less technical experience. Additionally, integrating artificial intelligence or machine learning into these tools could be investigated to improve both accuracy and efficiency in data reconciliation. The evaluation of these tools in broader and more diverse contexts, such as real-time data integration, is also suggested to expand their applicability and robustness in the field of linked open data.

5. References

[1] World Wide Web Consortium (W3C). (2024). W3C. Available in: https://www.w3.org/

[2] Ávila Barrientos, E. (2020). *Linked data and its use in libraries*. Universidad Nacional Autónoma de México. Available in: http://ru.iibi.unam.mx/jspui/handle/IIBI_UNAM/56

[3] World Wide Web Consortium (W3C). (2010). *Linked Open Data (LOD)*. W3C eGov Interest Group Wiki. Available in: https://www.w3.org/egov/wiki/Linked_Open_Data

[4] Sierra, Á. O. (2022). Insertion of metadata from Spanish libraries in Wikidata: a linked open data model. *Revista Española de Documentación Científica*, 45(3), a330-a330.

[5] Hyland, B., Atemezing, G., & Villazon-Terrazas, B. (2014). Best Practices for Publishing Linked Data-W3C Working Group Note 09 January 2014. 2014-09-04]. Available in: http://www.w3.org/TR/ld-bp

[6] Berners-Lee, T., Hendler, J., & Lassila, O. (2023). The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. In *Linking the World's Information: Essays on Tim Berners-Lee's Invention of the World Wide Web* (pp. 91-103).

[7] Wikimedia Foundation. (2023). *Wikidata: Main page*. Available in: https://www.wikidata.org/wiki/Wikidata:Main_Page

[8] Wikimedia Foundation. (2021). *Wikidata: Linked open data work-flow*. Available in: https://www.wikidata.org/w/index.php?title=Wikidata:Linked_open_data_work-flow&oldid=1394378153

[9] Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., & Vrandečić, D. (2014). 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 (pp. 50-65). Springer International Publishing.

[10] Vrandečić, D., & Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10), 78-85.

[11] Nishanbaev, I., Champion, E., & McMeekin, D. A. (2021). A web GIS-based integration of 3D digital models with linked open data for cultural heritage exploration. *ISPRS international journal of geo-information*, 10(10), 684.

[12] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A. et al. (2015). Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2), 167-195.

[13] Chessa, A., Fenu, G., Motta, E., Osborne, F. et al. (2023). *Data-Driven Methodology for Knowledge Graph Generation Within the Tourism Domain*. IEEE Access, 11, 67567–67599. Available in: https://doi.org/10.1109/ACCESS.2023.3292153

[14] Brandt, M. B., Borsetti Gregorio Vidotti, S. et al. (2018, May 1). Legislative linked open data: A proposal of linked open data modeling for legislative information. *Informacao e Sociedade*. Universidade Federal de Campina Grande.

[15] Diefenbach, D., Lully, V., Migliatti, P. et al. (2019). QAnswer: A question answering prototype bridging the gap between a considerable part of the LOD cloud and end-users. In *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019* (pp. 3507–3510). Association for Computing Machinery, Inc. Available in: https://doi.org/10.1145/3308558.3314124

[16] Uzun, A. (2019). Semantic Enrichment of Mobile and WiFi Network Data. *In T-Labs Series in Telecommunication Services* (pp. 59–96). Springer Science and Business Media B.V. Available in: https://doi.org/10.1007/978-3-319-90769-7_4

[17] Martínez, M. R. (2022). *Quality Metrics to Validate Government Public Open Data Sets* (Doctoral dissertation, Universidad Nacional de La Plata).

[18] Thornton, K., Seals-Nutt, K., Van Remoortel, M. et al. (2022). Linking women editors of periodicals to the Wikidata knowledge graph. *Semantic Web*, 14(2), 443–455. Available in: https://doi.org/10.3233/SW-222845

[19] Chalupsky, H., Szekely, P., Ilievski, F. et al. (2021). Creating and viewing custom versions of Wikidata on a laptop. En *Actas del taller CEUR* (Vol. 2982). CEUR-WS.

[20] Freshour, S., Kiwala, S., Cotto, K. et al. (2021). Integration of the drug-gene interaction database (DGIdb 4.0) with open collaboration initiatives. *Nucleic Acids Research*, 49 (D1), D1144–D1151. Available in: https://doi.org/10.1093/nar/gkaa1084

[21] Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., & Vrandečić, D. (2014). 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* (pp. 50-65). Springer International Publishing.

[22] García, A. M. F., García, M. I. M., Gallinas, R. B., Sánchez, M. M., & Gutiérrez, M. E. B. (2022). 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*(1), 1-19.

[23] Vinoth, A., Thangasamy, S., Nithya, R. et al. (2023, December). The Impact of Tamil Python Programming in Wikisource. In *International Conference on Speech and Language Technologies for Low-resource Languages* (pp. 79-90). Cham: Springer Nature Switzerland.

[24] Deng, S., Heng, G., Xu, A., Zhu, L., & Li, X. (2022). Enhance the Discovery and Interoperability of Culturally Rich Information: the Chinese Women Poets WikiProject.

[25] Schmidt, S. C., Thiery, F., & Trognitz, M. (2022). Practices of linked open data in archaeology and their realisation in Wikidata. *Digital*, 2(3), 333-364.

[26] Azcarraga, L. A. (2023). Radical openness and free knowledge. Repository of open access Mexican academic journals through Wikidata. *In SciELO Preprints*. Available in: https://doi.org/10.1590/SciELOPreprints.5607.

[27] Wikidata (2024). Wikidata:REST API. Available in: https://www.wikidata.org/wiki/Wikidata:REST_API

[28] Wikidata (2024). Wikidata:Stable Interface Policy. Available in: https://www.wikidata.org/wiki/Wikidata:Stable_Interface_Policy

[29] Wikidata (2024). Wikidata:SPARQL query service/queries. Available in: https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/queries

[30] Wikidata (2024). Wikidata:SPARQL Query Service/Help. Available in: https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/Wikidata_Query_Help/es

[31] Wikidata (2024). Wikidata:SPARQL tutorial. Available in: https://www.wikidata.org/wiki/Wikidata:SPARQL_tutorial/es

[32] Wikidata (2024). Wikidata:Pywikibot - Python 3 Tutorial. Available in: https://www.wikidata.org/wiki/Wikidata:Pywikibot_-_Python_3_Tutorial

[33] MediaWiki (2024). Manual:Pywikibot/Wikidata. Available in: https://www.mediawiki.org/wiki/Manual:Pywikibot/Wikidata

[34] Wikidata (2024). Wikidata:Tools/OpenRefine/Editing. Available in: https://www.wikidata.org/wiki/Wikidata:Tools/OpenRefine/Editing

[35] OpenRefine (2024). Overview of Wikibase support. Available in: https://openrefine.org/docs/manual/wikibase/overview