Experiences of Using WDumper to Create Topical Subsets from Wikidata

Citation for published version:
Hosseini Beghaeiraveri, SA, Gray, AJG & McNeill, FJ 2021, 'Experiences of Using WDumper to Create Topical Subsets from Wikidata', *CEUR Workshop Proceedings*, vol. 2873, 13. <http://ceur-ws.org/Vol-2873/paper13.pdf>

Link:
Link to publication record in Heriot-Watt Research Portal

Document Version:
Publisher's PDF, also known as Version of record

Published In:
CEUR Workshop Proceedings

Publisher Rights Statement:
Copyright © 2021 for this paper by its authors.

Experiences of Using WDumper to Create Topical Subsets from Wikidata

Seyed Amir Hosseini Beghaeiraveri[1][0000−0002−9123−5686] sh200@hw.ac.uk,
Alasdair J.G. Gray[1][0000−0002−5711−4872] a.j.g.gray@hw.ac.uk, and
Fiona J. McNeill[2][0000−0001−7873−5187] f.j.mcneill@ed.ac.uk

[1] School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK
[2] School of Informatics, The University of Edinburgh, Edinburgh, UK

Abstract. Wikidata is a general-purpose knowledge graph covering a wide variety of topics with content being crowd-sourced through an open wiki. There are now over 90M interrelated data items in Wikidata which are accessible through a public query endpoint and data dumps. However, execution timeout limits and the size of data dumps make it difficult to use the data. The creation of arbitrary topical subsets of Wikidata, where only the relevant data is kept, would enable reuse of that data with the benefits of cost reduction, ease of access, and flexibility. In this paper, we provide a working definition for topical subsets over the Wikidata Knowledge Graph and evaluate a third-party tool (WDumper) to extract these topical subsets from Wikidata.

Keywords: wikidata · knowledge graph subsetting · topical subset · wdumper

1 Introduction

A Knowledge Graph (KG) is defined as representing real-world entities as nodes in a graph with the relationships between them captured as edges [10]. In recent years, there have been a growing number of publicly available KGs; ranging from focused topic specific ones such as GeoLinkedData [12], EventMedia [11], and UniProt [16], to more general knowledge ones such as Freebase [6], DBpedia [3], and Wikidata [17]. These general purpose KGs cover a variety of topics from sports to geography, literature to life science, with varying degrees of granularity.

Wikidata [17] is a collaborative open KG created by the Wikimedia Foundation. The main purpose of Wikidata is to provide reliable structured data to feed other Wikimedia projects such as Wikipedia, and is actively used in over 800 Wikimedia projects[3]. It contains over 90 million data items covering over

[3] https://www.wikidata.org/wiki/Wikidata:Statistics accessed 4 February 2021

75 thousand topics[4]. Regular dumps of the data are published in JSON and RDF under the Creative Commons CC0 public license. However, the size of the gzipped download files has grown from 3GB in 2015 to 85GB in 2020, and keeps increasing as more data is added. The size of these files make it increasingly difficult and costly for others to download and reuse, particularly if only focused on a particular topic within the data, e.g. life sciences data or politicians. While Wikidata can be queried directly through an open SPARQL endpoint[5], it is subject to usage limits (as are other public endpoints of KGs) which limits the scale and complexity of queries that can be issued [17]. Thus there is a need for topical subsets of large KGs that contain all data on a specific topic to enable complex analysis queries to be conducted in a cost effective and time efficient way.

Our motivation for research on Wikidata subsetting is a combination of research goals, flexibility and ease of use. From the flexibility and ease of use perspective, we are looking for Wikidata subsets that can allow users to run smaller versions of Wikidata on available platforms such as laptops and PCs. Wikidata, as a knowledge graph with an interesting data model, has significant features for inspiration and improvement, but the speed of research and the diversity of researchers will be reduced if any experiment on it requires powerful servers, processing clusters, and hard disk arrays. From the research point of view, our motivation is creating a type of subset we call *Topical Subset* which is a set of entities related to a particular topic and the relationships between them. Having topical subsets of Wikidata for example in the fields of art, life sciences, or sports, not only helps us achieve the first goal (flexibility and ease of use) but also provides a platform for comparing and evaluating Wikidata features in different topics. The subset will also make the research experiments reproducible as the data used can be archived and shared more easily.

You can envision that such subsets can be generated through SPARQL CONSTRUCT queries. While this is straightforward for small subsets focused on a single entity type, e.g. politicians, it does not scale to interrelated topics that make up a larger domain, e.g. the life sciences subset defined by GeneWiki [18]. In this paper, we present our experiences of defining and creating topical subsets over Wikidata using WDumper [1]; a third-party tool provided by Wikidata to create custom RDF dumps of Wikidata. We define topical subsets over a KG (Section 3.2) and evaluate WDumper as a practical tool to extract such subsets (Section 4).

2 Related Work

KG subsetting, particularly in the context of Wikidata [2], has been gaining attention in the last couple of years, with use cases and potential approaches

---

[4] Query to count all data items `https://w.wiki/yVY` accessed 9 February 2021. Note that the query execution timesout if you try to return the count query `SELECT (COUNT(DISTINCT ?O) AS ?numTopics) WHERE { ?s wdt:P31 ?o }`.

[5] `https://query.wikidata.org/sparql` accessed February 2021

being explored [7]. Although there have been several approaches to subsetting proposed, to the best of our knowledge there is no agreed definition for a topical subset nor a unified and evaluated way to create such subsets. The Graph to Graph Mapping Language (G2GML) enables the conversion of RDF graphs into property graphs using an RDF file or SPARQL endpoint and a mapping file [13]. This could be exploited to create a topical subset based on the definition in the mapping file. However the output would be a property graph. Context Graph [14] can be used to generate subsets of an RDF KG, and was developed to create subsets for testing knowledge base completion techniques. The approach captures all nodes and edges within a given radius of a set of seed nodes. While the generated subsets are suitable for testing knowledge base completion techniques, there is no guarantee to the topical cohesion of the subset, and thus do not meet the needs identified for this work. YAGO4 [15] is a collection of Wikidata instances under the type hierarchy of schema.org [9]. It suggests a new logically consistent knowledge base over the Wikidata A-Boxes, however, there is no choice as to what topic be in the final KG.

WDumper [1] is a third-party tool for creating custom and partial RDF dumps of Wikidata suggested at the Wikidata database download page[6]. The WDumper backend uses the Wikidata Toolkit (WDTK) Java library to apply filters on the Wikidata entities and statements, based on a specified configuration that is created by its Python frontend. This tool needs a complete JSON dump of Wikidata and creates an N-Triple file as output based on filters that the config file explains. The tool can be used to create custom subsets of Wikidata. In Section 4.5 we will investigate whether it can generate topical subsets.

Shape Expressions (ShEx) [8] is a structural schema language allowing validation, traversal and transformation of RDF graphs. By caching triples that match the constraints, referred to as "slurping", a subset of the dataset can be created that conforms to the ShEx schema. Therefore to create a topical subset, all that is required is the definition of the ShEx schema. However, this approach is not available at scale yet and thus cannot be applied to Wikidata.

3   Knowledge Graph Subsetting

General purpose KGs like Wikidata are valuable sources of facts on a wide variety of topics. However, their increasing size makes them costly and slow to use locally. Additionally, the large volume of data in Wikidata increases the time required to run complex queries. This often restricts the types of queries that can be posed over the public endpoint since it has a strict 60-second limit on the execution time of queries. Downloading and using a local version of Wikidata is a way of circumventing the timeout limit. However, this is not a cheap option due to the size of the data. A suggested system to have a personal copy of Wikidata includes 16 vCPUs, 128GB memory, and 800GB of raided SSD space[7].

---

[6] `https://www.wikidata.org/wiki/Wikidata:Database_download`
[7] See this post: `https://addshore.com/2019/10/your-own-wikidata-query-service-with-no-limits/`

There are a large number of use case scenarios where users will not need access to all topics in a large general purpose KG. A small and complete enough subset can be more suitable for many purposes. With a small subset inference strategies can be applied to the data and complete in reasonable time. Topical subsets could also be published along with papers, which provides better reproducibility of experiments. Therefore having a topical subset that is smaller but has the required data can enable complex query processing on cheap servers or personal computers — reducing the overall cost — whilst also providing an improvement in query execution times.

3.1 Topical Subset Use Cases

We now define four use cases for topical subsets in Wikidata that we will use to review WDumper, and can also be used in other reviews as a comparison platform. Note that the use cases are defined in terms of English language statements. A subsetting approach, method, or tool would need to formalise these, as appropriate for their configuration, to extract the relevant data.

Politicians: This subset should contain all entities that are an instance of the class *politician*, or any of its subclasses. In the case of Wikidata, this would be the class *Q82955*, while for DBpedia it would be the class *Politician*. The subset should contain all facts pertaining to these entities, i.e. in Wikidata all statements and properties.

General(military) Politicians: The subset should contain all entities that are an instance of the class *politician*(Q82955) or any of its subclasses, who also are a *military officer*(Q189290) and have the rank of *general*(Q83460), i.e. politico-military individuals. The main goal of this use case is to see the effect of having more conditions in the English definition on the run-time and the volume of the output of subset extraction tools.

UK Universities: The subset should contain all instances of the class *university*(Q3918) or any of its subclasses, that are located in the UK. The subset should contain all statements and properties pertaining to these entities. This use case extends the complexity of the subset by having alternative properties and values to satisfy, e.g. the location can be captured in Wikidata with the properties *country(P17)*, *located in territory(P131)*, or *location(P276)*. Likewise the country could be stated as one of the component parts of the UK, e.g. Scotland.

Gene Wiki: This case is based on the class-level diagram of the Wikidata knowledge graph for biomedical entities given in [18]. The class-level diagram specifies 17 different item types from Wikidata mentioned in the Gene Wiki project. The subset should contain all instances of these 17 classes and their subsets.

The selection of these use cases is a combination of research and experimental goals. The Gene Wiki and Politicians use cases have been selected for future research purposes because of their hypothetical richness in references. The other

Listing 1.1. An example of a function R which is a query to return all entities with type city

```
SELECT ?entity WHERE {
  ?entity wdt:P31 wd:Q515 . # instance of(P31) city(Q515)
}
```

two use cases have been chosen to explore the expressiveness of a topical subset definition, and then to explore the runtime execution of these.

3.2 Topical Subset Definition

We now provide a definition for topical subsets based on the Wikidata data model. Wikidata consists of the following collections:

- E: set of Wikidata entities – their ID starts with a Q.
- P: set of Wikidata properties – their ID starts with a P.
- S: set of Wikidata statements.

Now we define the filter function $R : E \to E$ as a black-box that can be applied on E and selects a finite number of its members related to a specific topic. Let $E_R \subseteq E$ be the output of the function R. For entity $e \in E$ let $S_e \subseteq S$ be **all** simple and complex Wikidata statements in which e is the subject. Note that in Wikidata, a simple statement is a regular RDF triple, while a complex statement is a triple that references and/or qualifiers attached to it. Also, let P_e be **all** properties which are used in S_e triples either for the statement itself or qualifiers/references. With these assumptions, we define dump D_R as a topical subset of Wikidata with respect to R:

$$D_R := (E_R, \bigcup_{e \in E_R} P_e, \bigcup_{e \in E_R} S_e)$$

From the definitions of P_e and S_e we can conclude that $\bigcup_{e \in E_R} P_e \subseteq P$ and $\bigcup_{e \in E_R} S_e \subseteq S$ and subsequently D_R is a subset of Wikidata. We consider R as black-box; the input of R is the set of all Wikidata entities and its output is a subset of Wikidata entities related to a specific topic. The function R can be any set of definitions, rules, or filters that describe a related group of entities. The definition of R depends on the topic that is being described. One example of R is a simple `SELECT` query that describes all entities that have type $city$(Q515) (Listing 1.1).

4 WDumper

WDumper is a tool provided by Wikidata for producing custom subsets from Wikidata[8]. The R function can be seen as a filter approach on entities. For each

---

[8] `https://www.wikidata.org/wiki/Wikidata:Database_download#Partial_RDF_dumps` - accessed 9 February 2021
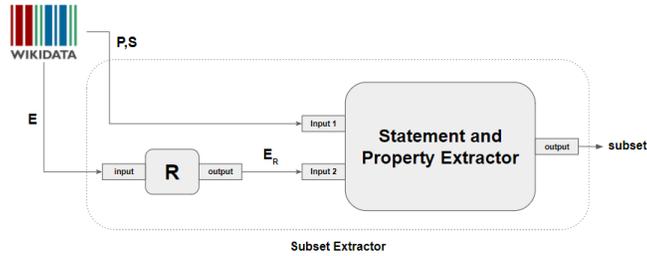
Fig. 1. Component overview of WDumper

topic, the appropriate filters on entities must be defined. Once the filters are defined and the subset E_R is extracted, WDumper extracts all statements with origin e, where $e \in E_R$, along with their qualifiers and references. A component overview of WDumper is given in Figure 1.

WDumper requires two inputs. The first is the complete dump of the Wikidata in JSON. The second is a JSON specification file that contains rules and filters for determining which entities, properties and statements to extract from the full Wikidata dump. This is the definition of the function R. The output of WDumper is an N-Triple (.nt) file that contains the entities and statements specified in the second input. There is also a GUI for creating the input specification file. We review WDumper through the following steps:

1. Writing WDumper specifications for the use cases in Section 3.1.
2. Running WDumper with the above specifications on two complete Wikidata dumps belonging to two different time points and compare the **run-time** and the **volume** of the extracted output.
3. Evaluating the extracted output via performing different queries both on the output and the input full dump.
4. Summarizing results and expressing strengths and weaknesses of WDumper.

4.1 WDumper Specification Files

Section 3.1 introduced use cases to evaluate subset extraction tools. In this section, we describe the corresponding WDumper specification files [5] generated using the GUI depicted in Figure 2. The GUI provides several controls as to what to include in the subset. The first is to select whether you are interested in items or properties. Using properties allows you to extract a subset of the Wikidata ontology, while items returns data instances. Items can be further filtered by giving a property and value. Other options then permit you to select whether all statements are returned ("any") or just the top ranked statements ("best rank")[9]. These filters allow the WDumper to extract the intermediate

---

[9] Note that in Wikidata, each statement can have a rank that can be used to identify the preferred value if the statement has more than one value. Ranks are available in Wikidata RDF data model like qualifiers and references.
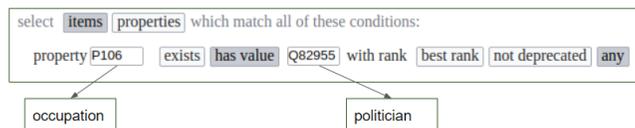
Fig. 2. WDumper Configuration GUI for the politicians use case.

nodes of statements, references, and qualifiers. In all use cases, we created a specification with and without the additional references and qualifiers. This allows us to investigate the effect of statement filters on execution time and output volume. In the specification file, these filters can be seen in the statement sub-array, as "qualifiers", "references", "simple", "full" keys which are false or true respectively.

Politicians: We define a filter on the *occupation property(P106)* of the entities to be a *politician(Q82955)*.

General(military) Politicians: This extends the Politicians definition with two more conditions. The first is the *occupation property(P106)* to be *military officer(Q189290)*. The second is the *military rank property(P410)* to be *general(Q83460)*.

UK Universities: We define a filter on entities with two conditions: the *instance of property(P31)* to be *university(Q189290)*, and the *country property(P17)* to be *United Kingdom(Q145)*.

Gene Wiki: For each item type of the class-diagram in [18], we create a filter on the corresponding entities in WDumper via the *instance of property(P31)* to be *gene(Q7187)*, *protein(Q8054)*, etc. In this case, no filters are defined on the types as we require all statements associated with the types ot be in the subset.

4.2 Experimental Setup

We now give details of our experimental environment in which we will evaluate WDumper.

Input Dumps. We use two full dumps of the Wikidata database. The first full dump is from 27 April 2015[10], and the second is from 13 November 2020[11]. The selected 2020 dump was the latest JSON dump available when conducting our evaluation. The selected 2015 dump is the first archive date for which both JSON and Turtle files are available (we need the JSON file for WDumper running, while the Turtle file is needed to import the full dump in a triplestore and evaluate output of WDumper based on the input). Table 1 provides summary information

---

[10] Downloaded from `https://archive.org/download/wikidata-json-20150427` - accessed 11 November 2020

[11] Downloaded from `https://dumps.wikimedia.org/wikidatawiki/entities/` - accessed 15 November 2020

Table 1. Details of the Wikidata dumps. The Total Items and Total Statements columns are obtained from the Wikidata Stats tool[13].

Release Date	JSON Compressed Size (GB)	Total Items	Total Statements
2015-04-27	4.52	17,632,496	61,427,934
2020-11-13	90.42	90,368,519	1,153,949,899

about these two dumps. The 2015 dump is smaller, can be stored and processed locally even on PCs, and it takes a much shorter time to generate output. For this reason, it is very suitable for initial tests. The 2020 dump, on the other hand, is much richer and can provide insights on how WDumper deals with large datasets of the size that Wikidata now produces.

Experimental Environment. Experiments were performed on a multi-core server with 64 8-core AMD 6380 2.5GHz 64bit CPUs, 512GB of memory and 2.7TB disk space. Java openJDK version 11 (build 11+28) and Gradle 6.7.1 was used to compile and run WDumper.

Experimental Run. The calculated times have been extracted from the elapsed time mentioned in WDumper output log. For each of the execution cases, three independent runs were performed. The average and standard deviation from these times were calculated.

4.3 Evaluating WDumper

WDumper was run with the specification files described in Section 4.1 and the two Wikidata dumps. The generated subsets are included in [5] and the results stated in Table 2. For each use case, we generated subsets of simple statements and with the inclusion of statement nodes, references, and qualifiers (labelled "withRQFS" in the table).

Initial observations from Table 2 show that the run-time on the 2020 dump is significantly longer than the 2015 dump. This can be justified by the larger volume data that must be processed to produce the subset. In all cases, generating the subset with additional statements took longer than generating the simple statements, and produced more volume in the output, indicating the addition of references, qualifiers, and statements nodes in the output. For example, this change is very significant in the case of Gene Wiki in the 2020 dump. The added filters as well as the conditions added to the filters also have a direct effect on the run-time and an inverse effect on the output volume, which is to be expected. Of course within run-times, the amount of data that must be written in the output must also be considered. This is evident in comparisons between UK universities and the military politicians in which the volume of data has a greater impact than the number of conditions. Overall, considering the high volume of data, the time required to extract a topical subset by WDumper seems appropriate.

---
[13] https://wikidata-todo.toolforge.org/stats.php

Table 2. Time elapsed and the size of the output in running WDumper on each use case and full dumps. "withRQFS" label denotes the specification that aims to extract statement nodes, references, and qualifiers. The FoE column denotes the number of filters on entities. The CiF column denotes the number of conditions in each filter. The average of three runs are reported (AVG column) together with the standard deviation (SD column). Sizes belong to compressed nt.gz files that are the direct output of WDumper. Inside parentheses, sizes and times are converted to other units for more readability (h for hours, KB for kilobytes, and MB for megabytes).

Use case	FoE	CiF	2015 Dump			2020 Dump		
			Time (sec)		Size (GB)	Time (sec)		Size (GB)
			AVG	SD		AVG	SD	
Politicians	1	1	1835 (31min)	198	0.07 (70MB)	39779 (11h)	2029	0.37 (370MB)
Politicians withRQFS	1	1	2347 (39min)	28	0.31 (300MB)	44271 (12h)	2811	1.4
UK Universities	1	2	1584 (26min)	38	0.000105 (105KB)	41474 (12h)	3424	0.000255 (255KB)
UK Universities withRQFS	1	2	1774 (30min)	142	0.000175 (175KB)	41890 (12h)	8436	0.000864 (864KB)
General (military) Politicians	1	3	1570 (26min)	59	0.000268 (268KB)	37155 (10h)	917	0.00105 (1MB)
General (military) Politicians withRQFS	1	3	1655 (28min)	48	0.000664 (664KB)	42334 (12h)	7341	0.04 (4MB)
Gene Wiki	17	1	1731 (29min)	95	0.01 (11MB)	40828 (11h)	2284	0.70 (709MB)
Gene Wiki withRQFS	17	1	1844 (31min)	276	0.026 (26MB)	48943 (14h)	4993	4.5

Adding more filters does not have a huge effect on runtime which is dominated by data volume.

4.4 Topical Subset Validation

The previous section considered the runtime performance of the WDumper tool and the size of the generated subsets. We now consider validating the content of the subsets. That is, we consider if the produced output has the information that it was supposed to have according to the definition in Section 3.2. Our assessment will be based on the following conditions:

Condition 1: The number of filtered entities in the output should be equal to the same number of entities in the input dump. For example, in the Politicians use case, the number of persons with the occupation of politician in output should be equal to the number of persons with the occupation of politician in the corresponding input dump. This condition can be tested with COUNT queries on the input and output datasets.

Listing 1.2. Commands used for fixing syntax errors of the 2015 dump.

```
sed -i -E 's/(<.*)}(.*>)/\1\2/' <dump_file>
sed -i -E 's/(<.*)\\n(.*>)/\1\2/' <dump_file>
sed -i -E 's/(<.*)\|(.*>)/\1\2/' <dump_file>
```

Condition 2: For each entity that is supposed to be in output, the number of its related statements must be equal in both input and output datasets. For example, in the Politicians use case, if the main dump has 50 statements about George Washington, we expect to see the same number of statements about this politician in the output too. This condition can be tested using DESCRIBE queries.

Condition 3: WDumper can extract intermediate statement nodes, references, and qualifiers exactly as they are at the input dump. This condition can be tested by querying the qualifiers and references of some given statements.

Testing the conditions requires running different queries on the input and output of WDumper. For the output of WDumper, we use Apache Jena Fuseki version 3.17 to import data as TDB2 RDF datasets and perform queries.

Data Corrections. We encountered problems loading the 2015 turtle dump[14] into Fuseki. Errors arose from the inclusion of bad line endings in more than 100 cases, and the existence of characters such as '\a'. Unacceptable characters such as '\n' and '\\' can also be seen in the WDumper outputs, which reinforces the possibility that this problem occurs due to the conversion of information from the JSON file to RDF format.

For the WDumper outputs and the 2015 dump, these errors were manually fixed using the `sed` commands given in Listing 1.2. The sanitized versions are available in [4]. In the case of 2020 dump, we use Wikidata Query Service (WDQS) due to the time that would be required to fix and load this data into Fuseki. The date of implementating our evaluation queries is approximately two months after the creation date of the 2020 dump (27 November 2020). In this period, new data may have entered Wikidata which are available by WDQS and are not present in 2020 dump (and subsequently are not present in the WDumper output). Because of this, there may be slight differences in the counts of entities and statements between input and output that is not related to WDumper functionality. We tried to use Wikidata history query service[15] to quantify the rate of Wikidata increases in this period, but the history covers a range from the creation of Wikidata to July 1st 2019.

Validation of Condition 1. We use COUNT queries to validate this condition. The purpose of these queries is to count the entities that should be in the output according to the filter(s) of each use case. If WDumper is performing correctly,

---

[14] Downloaded from `https://archive.org/download/wikidata-json-20150427` - accessed 20 December 2020

[15] `https://www.wikidata.org/wiki/Wikidata:History_Query_Service`

Listing 1.3. COUNT queries for evaluating condition 1. Prefixes and most of Gene Wiki's query have been deleted for more readability.

```
############ Politicians ##############################
SELECT (COUNT (DISTINCT ?item) AS ?count) WHERE{
?item wdt:P106 wd:Q82955 . # occupation of politician
}

############ UK Universities ##########################
SELECT (COUNT (DISTINCT ?item) AS ?count) WHERE{
?item wdt:P31 wd:Q3918 ; # instance of university
      wdt:P17 wd:Q145  . # country of United Kingdom
}

############ General(military) Politicians ############
SELECT (COUNT (DISTINCT ?item) AS ?count) WHERE{
?item wdt:P106 wd:Q82955  ; # occupation of politician
      wdt:P106 wd:Q189290 ; # occupation of milit. officer
      wdt:P410 wd:Q83460  . # military rank of general
}

############ Gene Wiki ################################
SELECT (COUNT (DISTINCT ?item) AS ?count) WHERE{
{?item wdt:P31 wd:Q423026 .}   # instance of active site
UNION
{?item wdt:P31 wd:Q4936952 .}  # instance of anat. struct.
UNION
# ...
UNION
{?item wdt:P31 wd:Q50379781 .} # instance of therap. use
}
```

Table 3. Results of performing COUNT queries of each use case (Listing 1.3) on the output of WDumper and input full dump for both 2015 and 2020 dumps. The last column are COUNT results queried against WDQS instead of the 2020 dump itself.

Use case	2015 Dump		2020 Dump	
	Output	Input	Output	Input
Politicians	246,009	246,044	641,387	646,401
General (military) Politicians	165	165	597	602
UK Universities	73	73	183	186
Gene Wiki	19,432	19,432	3,282,560	3,283,471

the result of this count should be the same on both the input and output datasets. These queries will be different for each use case, depending on the definition of that use case. For example, while in the Politicians use case we count the number of people with political jobs, in the case of Gene Wiki we count the union of entities of type disease, genes, proteins, etc. Listing 1.3 shows the queries executed for each use case. These queries run on the each use case's output "withRQFS" since these are included in the input dataset. The results of performing the COUNT queries are shown in Table 3.

Our results show that for the 2015 dump, the number of entities in the output and input is equal except for the Politicians use case. In both 2015 and 2020 dumps, the difference between input and output is less than one percent in the cases of inequality. In the case of 2020 dump, the difference can be attributed to the entry of new data in the interval between our tests and the dump date. This is reasonable especially in the case of Gene Wiki where bots are importing new information into Wikidata every day. In the case of the 2015 dump in the Politicians row, the 35 differences between input and output is unjustifiable. The reason for this difference may be the inability of WDumper to parse the data

Table 4. Results of performing DESCRIBE queries on the selected entity.

Use case	Entity	2015 Dump		2020 Dump	
		Output	Input	Output	Input
Politicians	Q23	408	776	871	921
General (military) Politicians	Q355643	104	150	207	228
UK Universities	Q1094046	64	108	208	224
Gene Wiki	Q30555	12	22	30	37

Table 5. Numbers of predicates in the 2015 dump and WDumper could not fetch, by entity.

Entity	schema:name	skos:prefLabel	Total
Q23	184	184	368
Q355643	23	23	46
Q1094046	22	22	44
Q30555	5	5	10

of these entities in the input dump. WDumper uses the JSON file as input, and to be able to fetch an entity, it must see the specific structure of the Wikidata arrays and sub-arrays in the JSON file. Some entities may not have this complete structure in the JSON file but they do exist in the Turtle file.

Validation of Condition 2. To validate this condition, in each use case we use DESCRIBE queries for an arbitrary entity that is in the WDumper output. DESCRIBE queries list all triples of the given entity. We expect that the result of the DESCRIBE queries should be the same on both the input and output datasets. For each use case, we selected an arbitrary entity (called Tested Entity) which is present in both the input and output dataset. We then run a `DESCRIBE wd:Q...` query and count the extracted triples. Table 4 shows our results.

From Table 4 it is clear that the number of triples in the DESCRIBE queries in both the 2020 and 2015 dumps are not equal. This difference prompted us to explore the differences using the compare module of the RDFlib library. It was found that in the case of the 2015 dump, the input dump contains predicates such as `<http://www.w3.org/2004/02/skos/core#prefLabel>` and `<http://schema.org/name>`, which are not extracted by WDumper. Table 5, shows the details and total numbers of predicates that are in the input dump (2015 dump) for the selected entities and WDumper could not fetch. As we can see, the total column is exactly the difference between the describe queries. In the case of 2020 dump, some of predicates with `<http://www.w3.org/2004/02/skos/core#>` prefix, such as `dateModified`, and all `<http://www.wikidata.org/prop/direct-normalized/>` predicates are not detectable by WDumper. However, in both dumps the statements whose predicate is a property of Wikidata (e.g. P31, P106, etc.), were completely extracted by WDumper.

Table 6. Number of qualifiers and references for the selected property of the selected entity in the output and input of WDumper (2020 dump).

Entity	Property	Qualifiers		References	
		Output	Input	Output	Input
Q23	P26	4	4	2	2
Q355643	P485	1	1	1	1
Q1094046	P355	1	1	1	1
Q17487737	P680	24	24	96	96

Validation of Condition 3. To validate this condition, we selected an arbitrary entity from each use case, and for this entity, we considered one of its statements. We then counted the qualifiers and the references of this statement in 2020 dump (over the WDQS) and in the output of WDumper. Table 6 shows the selected entity, selected property, and the number of qualifiers and references for them. From Table 6, it is clear that WDumper can extract qualifiers and references completely from the input.

4.5 Summarizing Results, Strengths and Weaknesses

The results of our evaluations show that WDumper, as a custom dump production tool, can be used to create some topical subsets. This tool can correctly and completely extract the entities specified by its filters. It also extracts almost all statements related to entities (except it is not designed to extract some prefixes). One of the features we have been looking for is the ability to extract references and qualifiers of Wikidata statements, which WDumper can do. Setting up this tool is not very complicated; the user only needs to select the filters of the entities and statements, run the tool and it extracts all of the information at once. Its GUI is also somewhat helpful, while the JSON structure of its specification files is also simple and understandable.

Limitations and Weaknesses. The most important weakness of WDumper with regard to the topical subsets is the limitation in the definition of entity filters. In WDumper, entities can only be filtered based on the presence of a P_x property or having the value v for a P_x property. Although it is possible to deploy any number of such filters, this is not enough to specify some kinds of use cases. For example, suppose we want to specify the Scottish universities subset. By reviewing some of these universities on the Wikidata website, we can find that their corresponding entity does not have any property that directly indicates they belong to Scotland. Of course, we can define the R function of these subsets through indirect methods (for example, considering the Geo-location of entities in Scottish universities), but these type of filters are not available in WDumper.

The recognition of type hierarchies is another limitation of WDumper. In the case of UK universities, for example, the *University of Edinburgh(Q160302)* is not among the universities extracted by WDumper. The reason for this is

that the *instance of property(P31)* in this university refers to *public university(Q875538)* instead of *university(Q3918)*. In SPARQL queries, such cases are handled by the property paths like `wdt:P31/wdt:P279`*. These property paths are not available in WDumper. The strategy of considering more filters to cover all subtypes needs a comprehensive knowledge of Wikidata ontology. This strategy will fail if the class hierarchy changes.

Another limitation is the inability to communicate between different filters in multi-filter cases. For example, in the Gene Wiki use case, we may want diseases that are somehow related to a gene or protein, while in the WDumper output there are diseases that have nothing to do with genes, proteins, and other Gene Wiki types. The inability to choose another output format other than N-Triples, especially the Wikibase JSON output, which is more suitable for using the subset produced in a Wikibase instance and also has a smaller volume, is another limitation.

The main implications of these limitations is the reduction of flexibility of subset extracting with this tool. With these weaknesses, users have to spend much more time defining the desired subset.

5   Conclusion

In this paper, we reviewed the issue of building topical subsets over Wikidata. Our motivation for topical subsets is to enable efficient evaluation of complex queries over the knowledge graph with lower costs, reproducibility of experiments through archiving datasets, ease of use, and flexibility. We provided example use cases for topical subsets as well as a definition for topical subsets. This definition enables us to evaluate and compare subset creation tools.

In this study we used WDumper for topical subset extraction over Wikidata and tested it by measuring run-time and output volume on four different use cases. We evaluated the correctness of the subsets generated by WDumper by comparing the answers to queries over the subsets and the full knowledge graph. Our experience shows that WDumper **can be used** to generate topical subsets of Wikidata in some use cases but **not for all use cases**. WDumper can extract the entities specified by its filters and extract most statements related to those entities; it also fetches the statement nodes and references/qualifiers. However, WDumper has some weaknesses regarding topical subsets. Its main problem is the way it defines filters on entities that reduces the power of this tool to build topical subsets. The most tangible issue is the **inability to define and fetch subclasses** of a class of entities, which is important in many use cases. Our suggestion for the future works is to explore alternative subsetting approaches such as using SPARQL queries or Shape Expressions. With selectors like SPARQL queries or ShEx schemata, we can increase the expressivity of the subset creation. It will also allow for subsets to be created on Knowledge Graphs other than Wikidata.

Jose Emilio Labra Gayo, Eric Prud'hommaux, Andra Waagmeester, Harold Solbrig, Ammar Ammar, and Guillermo Benjaminsen.

References

1. Wdumper - a tool to create custom wikidata rdf dumps, `https://tools.wmflabs.org/wdumps/`, , GitHub repository: `https://github.com/bennofs/wdumper`
2. Wikidata:WikiProject Schemas/Subsetting - Wikidata, `https://www.wikidata.org/wiki/Wikidata:WikiProject_Schemas/Subsetting`, accessed 2020-12-31
3. Auer, S., Bizer, C., Kobilarov, G., et al.: DBpedia: A Nucleus for a Web of Open Data. In: ISWC (2007). https://doi.org/10.1007/978-3-540-76298-0$_5$2
4. Beghaeiraveri, S.A.H.: Wikidata dump 27-04-2015 fixed syntax errors (Feb 2021). https://doi.org/10.5281/zenodo.4534445
5. Beghaeiraveri, S.A.H.: Wikidata Subsets and Specification Files Created by WDumper (Feb 2021). https://doi.org/10.5281/zenodo.4495855
6. Bollacker, K., Tufts, P., Pierce, T., Cook, R.: A platform for scalable, collaborative, structured information integration. In: IIWeb'07 (2007)
7. Gayo, J.E.L., Ammar, A., Brickley, D., et al.: Knowledge graphs and wikidata subsetting (2021). https://doi.org/10.37044/osf.io/wu9et
8. Gayo, J.E.L., Prud'Hommeaux, E., Boneva, I., Kontokostas, D.: Validating RDF data, vol. 7. Morgan & Claypool Publishers (2017)
9. Guha, R.V., Brickley, D., Macbeth, S.: Schema. org: evolution of structured data on the web. Communications of the ACM **59**(2), 44–51 (2016)
10. Hogan, A., Blomqvist, E., Cochez, M., et al.: Knowledge Graphs. arXiv:2003.02320 [cs] (2020), `http://arxiv.org/abs/2003.02320`
11. Khrouf, H., Troncy, R.: Eventmedia: A lod dataset of events illustrated with media (2012)
12. Lopez-Pellicer, F.J., et al.: Geo linked data. In: DEXA (2010)
13. Matsumoto, S., Yamanaka, R., Chiba, H.: Mapping rdf graphs to property graphs. arXiv preprint arXiv:1812.01801 (2018)
14. Mimouni, N., Moissinac, J.C., Tuan, A.: Domain specific knowledge graph embedding for analogical link discovery. Advances in Intelligent Systems (2020)
15. Tanon, T.P., Weikum, G., Suchanek, F.: Yago 4: A reason-able knowledge base. In: European Semantic Web Conference. pp. 583–596. Springer (2020)
16. UniProt Consortium: Uniprot: a hub for protein information. NAR **43**(D1) (2015)
17. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. CACM **57**(10), 78–85 (2014). https://doi.org/10.1145/2629489
18. Waagmeester, A., et al.: Wikidata as a knowledge graph for the life sciences. eLife **9** (2020). https://doi.org/10.7554/eLife.52614