Constructing demand-driven Wikidata Subsets*

Daniel Henselmann[1][0000−0001−6701−0287] and
Andreas Harth[1,2][0000−0002−0702−510X]

[1] Fraunhofer Institute for Integrated Circuits IIS, Nürnberg, Germany
name.surname@iis.fraunhofer.de
[2] Friedrich–Alexander University Erlangen-Nürnberg, Nürnberg, Germany

Abstract. A Wikidata subset is a selected subset from the set of all triples included in Wikidata. We approach the task to create a subset with a demand-based subset that satisfies a present use case. The introduced algorithm constructs triples in multiple steps starting from a seed of URIs. Different input options for the seed, the sequence of construction steps, and a filter enable adaptations to the use case. A formal description and matching SPARQL queries complement the algorithm. Hospital data provides a running example.

Keywords: Wikidata · Subset · Subgraph · Neighbourhood

1 Introduction

Wikidata itself describes a Wikidata subset (Q96051494) as "part of the data contained in Wikidata"[3]. It is a selected subset from the set of all Resource Description Framework (RDF)[4] statements (triples) included in Wikidata. This paper proposes a way to construct a Wikidata subset.

Within our work in research and industry projects, we came upon three use cases that we considered for this paper: (1) data on life science institutions, (2) data on executives of listed corporations, (3) data on microelectronics and their supply chains. In all three cases, we want to extract fitting domain data from Wikidata according to our demands in the project.

Wikidata is one of the first addresses to link own RDF data to open data on the web or simply receive additional information to enhance own data. Wikidata is constantly growing in data volume and thus brings limitations to the practical interactions with the whole dataset. Reducing the Wikidata content

Supported by the German Federal Ministry of Education and Research (FKZ 16ME0224).

[3] See https://www.wikidata.org/wiki/Q96051494 (Last accessed 20 Mai 2021).
[4] See https://www.w3.org/TR/rdf11-concepts/ (Last accessed 4 August 2021).

to a smaller demand-driven subset brings multiple advantages. These include faster results (answers) to queries (questions) of machines (humans), an easier overview of available information, and the possibility of copies on regular-sized machines. *Beghaeiraveri et al.* [2] and *Mimouni et al.* [8] list extended advantages of subsets/subgraphs.

Challenges in building a subset include the volume of Wikidata's data and the resulting possibilities for subsets. A total number of triples N results in 2^N possible subsets, as any number up to N triples can be included in the subset in any variation. The task is to find a subset out of these based on a use case. As it is not feasible to create all subsets and choose afterwards, the task changes to find a starting point (the seed) in the data and from there draw a line separating included and excluded data. The seed thus would be any set of Wikidata items, including none and all.

The subset must fit the demands of a use case. The subset mustn't exclude relevant data to not miss any information which may lead to wrong conclusions. The subset also mustn't include unnecessary information to reduce computation costs when working with the subset and to increase the clarity of its content. In summary, we look for a minimal subset that still covers our demand (a minimal full-covering subset).

Another challenge is how to describe the need of the use case for the subset. We need enough detail to fulfill the criteria of the last paragraph while offering a feasible expression. Furthermore, the algorithm that creates the subset has to process the expression in a way that the subset fulfills it.

Currently available subsets of Wikidata don't contain domain data our use cases demand. Existing approaches to subsetting don't offer the depth of subsets we need for our use cases. Therefore, a new solution for demand-driven Wikidata subsets is required.

The main contribution of this paper is an algorithm to construct a demand-driven Wikidata subset including formal descriptions (see section 6.1). The necessary steps are explained using a running example of hospital data (see section 3). Additionally, we show how the SPARQL query language[5] can be used to implement the steps (see section 6).

2 Related Work

Wikidata has a webpage[6] on subsetting that collects efforts and ideas on the subject. The content focuses on stability and reusability of subsets, a focus we don't share (see section 4).

[5] See https://www.w3.org/TR/sparql11-query/ (Last accessed 29 July 2021).

[6] Available at https://www.wikidata.org/wiki/Wikidata:WikiProject_Schemas/Subsetting (Last accessed 29 July 2021).

Downloadable subsets from Wikidata itself exist in the form of dumps[7] but aren't domain-specific. Instead, Wikidata refers to WDumper[8] for custom dumps that can be domain-specific.

WDumper is a service that provides a subset of Wikidata according to various filters. However, it doesn't support variables as SPARQL queries do for example. Therefore, we can only query the direct neighbourhood of known Wikidata items selected by filters. This doesn't comply the demand of our use cases as we also look for data we are not yet aware of and thus can't select with filters. An evaluation of WDumper by *Beghaeiraveri et al.* confirms that the service can be used in some use cases but also has limitations [2].

DBpedia is similar to Wikidata and offers various download options[9] but neither of them is domain-specific.

The Concise Bounded Description (CBD) of a resource "is a subgraph consisting of those statements which together constitute a focused body of knowledge about the resource" [10]. We pursue the concept of a focused body of knowledge, but not regarding a single resource but rather a set of resources of a domain. This includes triples in a flexible depth from the starting node.

Beghaeiraveri et al. introduce the Topical Subset, "a set of entities related to a particular topic and the relationships between them" [2]. Similarly to the subsets we want to create, a Topical Subset includes data in the neighbourhood of a selected seed with the focus on a domain. However, a Topical Subset doesn't include triples in a flexible depth around the seed and thus doesn't construct a subset around the seed but rather gathers data about the seed. Therefore, the subsets our algorithm creates includes Topical Subsets but not vice versa.

Mimouni et al. present an algorithm to build a Context Graph [8]. Their algorithm has the same basic structure as our algorithm shown in section 6.1, nevertheless differences exist. The Context Graph algorithm has the neighborhood depth as an input parameter while we propose a more flexible construction tuple. Furthermore, their algorithm filters entities provided in the input while our approach includes a more general filter function that may exclude entities, properties, and classes that are discovered by the algorithm at runtime. These differences enable our algorithm to be adjustable to the demands of a use case.

A research area with similarities to our approach to Wikidata subsets is focused Web crawlers. *Olston and Najork* describe the basic algorithm: "Given a set of seed Uniform Resource Locators (URLs), a crawler downloads all the Web pages addressed by the URLs, extracts the hyperlinks contained in the pages, and iteratively downloads the Web pages addressed by these hyperlinks." [9, p. 178]. Our approach starts with a set of seed Uniform Resource Identifiers (URIs), downloads all Wikidata triples containing the URIs, extracts newly found URIs, and iteratively downloads the triples containing these URIs. Focused Web crawlers look to minimize the number of accessed Web pages and

[7] Available at https://www.wikidata.org/wiki/Wikidata:Database_download (Last accessed 29 July 2021).
[8] Available at https://wdumps.toolforge.org/ (Last accessed 29 July 2021).
[9] Available at https://www.dbpedia.org/resources/ (Last accessed 29 July 2021).

maximize the percentage of relevant pages among those [9, 1]. We share these goals for subsets as we want minimal but relevant data. To accomplish this, focused Web crawlers filter and rate every URL using various metrics and then add them to a set of yet to be downloaded URLs called frontier F. Each iteration, the top-rated URL in F is downloaded and processed which again adds new URLs to F [9, 4]. We use filters to exclude URIs we found. However, we don't iterate over single URLs in F but rather the complete set F_i which results in a completely new set of URIs F_{i+1} for the next iteration. Therefore, we also don't rate the URIs.

Another related research area is link traversal query execution. The main idea is to follow URIs found when querying distributed RDF data while executing said query [6, 7]. Similar to focused Web crawlers and our approach, a frontier of to be processed URIs exists.

3 Running Example

We use the graph in figure 1 as an example of a Wikidata subset throughout this paper. It shows simplified data from Wikidata on the hospital Charité. The rounded vertices represent Wikidata items (e.g. Q162684) that we present by their (sometimes shortened) labels (e.g. Charité) instead of their URIs (e.g. https://www.wikidata.org/wiki/Q162684). We use labels to increase readability, the real identifier of an item remains its URI which the original triples contain. The angular vertices represent literals that are concrete data values (e.g. 1958). *Charité* is the seed of the example subset.

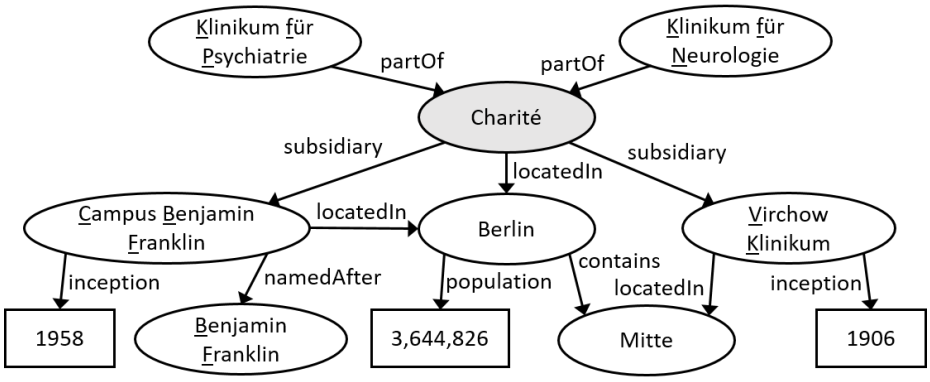

Fig. 1: Example of simplified data from Wikidata on the hospital *Charité*

The displayed subset shows some concepts of this paper: From the seed, inbound and outbound triples in varying distances contain data relevant for the use case and are thus included in the subset. When we iterate to construct the subset, we have to keep an eye on already visited URIs because multiple paths

may exist from the seed to an URI (see *Berlin* in fig. 1). Not all data in the adjacency of the seed is relevant to the use case (see (*CBF namedAfter BF*) in fig. 1). Therefore, we filter the triples before we add them to the subset.

4 Approaches

Two general approaches come to mind when thinking about how and why to create a Wikidata subset: (1) stable defined subsets, (2) subsets by acute demand. We shortly discuss both.

(1) An organization or group creates **stable defined subsets** and provides them to the public, maybe on the Wikidata website. Advantages include: The user of the subset – whoever that might be – does not necessarily need to know how to build a subset or be familiar with Wikidata content, as the creation is done and the download easy. The subset can also be used to normalize the contained Wikidata classes by analyzing instances and developing a recommendation for minimal properties. Disadvantages include: The available subsets don't comply the demand of individual use cases. Open questions include: Must all data belong to a subset? Must data belong to only one subset? What is a fair number of subsets considering relevance and minimalism? Effort: Large onetime effort to create all subsets (ignoring updates).

(2) An algorithm that creates a **subset by acute demand** is available. Every time a subset is required, it is created as needed. Advantages include: With the demand known, the subset can be minimal. Additionally, the subsets are always relevant because they only exist if there is an acute demand. Disadvantages include: The user has to create a subset himself. Therefore they needs time as well as knowledge of their demand, the domain of the subset, and Wikidata. Open questions include: How to express the demand of a use case? How to fit the subset to the demand? Effort: Pay as you use (after a tool is available).

When considering the use cases from our project work, the second approach is better fitting. In this way, we receive useful (minimal and full-covering) results in a quicker fashion. Reusability of created subsets is thus not the focus of our work.

5 Preliminaries

Let G be a set of RDF triples T, U be the set of URIs, B be the set of blank nodes, and L be the set of literals. Each triple $T = (s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ consists of a subject s, predicate p and object o according to the RDF specification[10]. The subject $s \in (U \cup B)$ may be an URI or blank node. The predicate $p \in U$ is always an URI. The object $o \in (U \cup B \cup L)$ may be an URI, a blank node, or a literal. G also represents a labeled directed multigraph, but we continue the view as a triple set in this text.

[10] See https://www.w3.org/TR/rdf11-concepts/ (Last accessed 30 July 2021).

Let $W \subset G$ be the set of all triples in Wikidata. Let $S \subseteq W$ be a Wikidata subset. Let $Q \subset U$ be the set of all Wikidata items. Let $F \subseteq Q$ be the set of URIs that are part of the frontier in an algorithm.

6 Solution Concept

In RDF, data is described through its relations. Therefore, all triples containing a certain URI describe the semantics of that URI. Additionally, data of interest for one domain is usually connected. Consequently, selected major URIs of interest and their surroundings build domain data. This fits our goal of a demand-driven subset as the demand usually covers one domain. The surroundings of a seed of URIs can be constructed through multiple iterations over adjacent URIs (neighbours) with increasing distance to the seed. In summary, a seed of selected URIs and data connected to it in varying distance in the form of triples build a subset (see fig. 1). Different options regarding the seed, the neighbourhood, and filters enable us to adjust the subset to the demand of a use case as shown in the following sections.

6.1 Basic Algorithm

We define the function that constructs a Wikidata subset S as

$$S = subset(W, C, seed(), filter()) \tag{1}$$

with the set of all triples in Wikidata W, the construction tuple $C = (c_0, ..., c_n)$, the *seed* function, and the $filter$ function as inputs. The basic algorithm is shown in the following.

1. Select a set of URIs $F_0 \subseteq Q$ as the seed ($F_0 = seed(W)$).
2. For each construction step $c_i \in C$,
   (a) get all triples $S_i \subseteq W$ that contain an URI $f \in F_i$ of the frontier F_i according to the construction step c_i ($S_i = construct(W, F_i, c_i)$).
   (b) filter the triples from set S_i to a set $S'_i \subseteq S_i$ ($S'_i = filter(S_i)$).
   (c) add S'_i to the desired subset $S \subseteq W$ ($S = S \cup S'_i$).
   (d) add F_i to the set of visited URIs V ($V = V \cup F_i$).
   (e) if c_{i+1} exists: get the set of URIs $F_{i+1} \subseteq Q$ that are adjacent to the URIs $F_i \subseteq Q$ according to the construction step c_i ($F_{i+1} = frontier(S'_i, c_i, F_i, V)$).
3. Export the complete subset S.

How these steps look in detail and combine is the focus of the following sections.

6.2 Seed Considerations

The seed acts as a starting point for the subset construction. It consists of a set of URIs $F_0 \subseteq Q$. F_0 will be the frontier of the first construction step that gets

triples from Wikidata that contain an URI $u \in F_0$. The function $F_0 = seed(W)$ is provided as an input to the algorithm to enable maximal flexibility.

One possibility to express *seed*() is a SPARQL SELECT query that runs on the Wikidata set W and returns F_0. A SPARQL query provides suitable possibilities to define the seed. In the example of figure 1, the SPARQL query for German hospitals shown in listing 1.1 returns $Charité \in F_0$ as one URI of the seed.

Listing 1.1: SPARQL query for German hospitals that returns the seed F_0

```
SELECT ?hospital
FROM W
WHERE {
?hospital wdt:P31 wd:Q16917 ; #instanceOf hospital
          wdt:P17 wd:Q183 . #country Germany
}
```

A basic possibility would be to provide F_0 in *seed*() by hand. This is a feasible option if only a few known URIs are relevant for the use case. In our example, the domain could not include all German hospitals but rather only the Charité, so $F_0 = \{Charité\}$.

6.3 Construction/Triple Considerations

In terms of RDF triples, we define the neighbourhood of a URI u in a set of triples G as the subset $N_G^{RDF}(u) \subseteq G$ containing all triples $T(u, p, o) \in G$ and $T(s, p, u) \in G$ that have u as subject or object. In these triples, all URIs adjacent to the URI u in G are included. The neighbourhood $N_G^{RDF}(U)$ of a set of URIs U defines the union of the neighbourhood of each URI $u \in U$. An example from figure 1 is provided at the end of this section.

Because triples build a directed graph, we can further distinguish adjacent URIs of an URI u between predecessors and successors. The triples that contain u as object and a predecessor are inbound triples. They provide "secondary knowledge" about u [10]. The triples that contain u as subject and a successor are outbound triples. They provide "primary knowledge" about u [10]. We define the subset of inbound triples of a URI u from a set of triples G as the subset $N_G^{-RDF}(u) \subseteq G$ containing all triples $T(s, p, u) \in G$ that have u as object. We define the subset of outbound triples of a URI u from a set of triples G as the subset $N_G^{+RDF}(u) \subseteq G$ containing all triples $T(u, p, o) \in G$ that have u as subject. The sets of inbound triples $N_G^{-RDF}(U) \subseteq G$ and outbound triples $N_G^{+RDF}(U) \subseteq G$ of a set of URIs U define as the union of the appropriate sets of each URI $u \in U$.

Over multiple iterations, we look at the neighbourhood of the given frontier F_i to construct the subset S. The input of the subset must express a sequence of construction steps adequately. Let $C = (c_0, c_1, ..., c_n)$ be a tuple with $c_i \in \{\leftrightarrow, \leftarrow, \rightarrow\}$ that represents the sequence of construction steps. For every iteration i, the expansion c_i is executed on the frontier F_i resulting in the partial subset $S_i \subseteq W$

that adds to the complete subset S. The construction step c_i may construct the whole neighbourhood ($\leftrightarrow$), the set of inbound triples ($\leftarrow$), or the set of outbound triples ($\rightarrow$). The size n of the tuple C expresses the number of iterations and the maximum depth of triples around the seed.

The dedicated function $S_i = construct(W, F_i, c_i)$ specifies to:

$$S_i = \begin{cases} N_W^{RDF}(F_i) & \text{if } c_i = \leftrightarrow \\ N_W^{+RDF}(F_i) & \text{if } c_i = \rightarrow \\ N_W^{-RDF}(F_i) & \text{if } c_i = \leftarrow \end{cases} \tag{2}$$

Regarding an implementation, several options to access and query Wikidata exist[11]. Available are per-item access with dereferenceable URIs, a SPARQL endpoint, and an endpoint for Linked Data Fragments. We won't evaluate these options as the implementation is not a focus of this paper. However, to give another expression of the constructed triples, the listings 1.2, and 1.3 show the SPARQL CONSTRUCT queries to construct the inbound triples ($c_i = \leftarrow$) and the outbound triples ($c_i = \rightarrow$) respectively. Both queries together construct the neighbourhood ($c_i = \leftrightarrow$).

Listing 1.2: SPARQL query that constructs the inbound triples of an URI u

```
CONSTRUCT
WHERE { GRAPH W { ?s ?p $u } . }
```

Listing 1.3: SPARQL query that constructs the outbound triples of an URI u

```
CONSTRUCT
WHERE { GRAPH W { $u ?p ?o } . }
```

In the example of figure 1, the first construction step $c_0 = \leftrightarrow$ returned the inbound and outbound triples of the seed $F_0 = \{Charité\}$. Therefore, the first partial subset is $S_0 = N_W^{RDF}(Charité)$ and returns the triples:

```
{(KfP partOf Charité), (KfN partOf Charité),
(Charité subsidiary CBF), (Charité locatedIn Berlin),
(Charité subsidiary VK)}
```

6.4 Filter Considerations

When we construct the neighbourhood of a URI, we receive all triples connected with that URI. This data quickly accumulates to a sizable subset. However, one of our goals for the subset is a smaller size. Additionally, we look for a minimal and full-covering subset. Thus, the algorithm has a filter function that can exclude triples that don't contain relevant information. It reduces a subset S_i to a filtered subset $S_i' \subseteq S_i$.

[11] See https://www.wikidata.org/wiki/Wikidata:Data_access (Last accessed 2 August 2021).

Because there are plenty of options to filter triples and the focus of filters varies between different use cases, the filter function $S'_i = filter(S_i)$ is provided as an input to the algorithm to enable maximal flexibility.

Some basic filter options to exclude triples are language-tagged literals for selected languages and triples using wikibase:Statements as they model duplicate information. Further possibilities are filters by Wikidata's property hierarchy or a filter for designated classes.

In this paper, however, we propose the idea of a filter for rarely used properties. We argue that triples with rarely used properties provide relatively little value for later analysis (like the *namedAfter* property in fig. 1). If we analyse hundreds or thousands of instances of one class, we can exclude properties that are only used by one or two instances. This may not sound much, but a few properties filtered for many URIs add up. Additionally, the filter function also reduces data volume in future iterations because every removed triple also means one less adjacent URI for the next construction, so the frontier F_{i+1} shrinks (see section 6.5).

Let $A^+_{px} \subseteq S_i$ be the set of triples $A^+_{px} = \{T = (u, p_x, o) \mid T \in S_i \text{ and } u \in F_i\}$ with the URIs $u \in F_i$ and a property p_x. Let $A^-_{px} \subseteq S_i$ be the set of triples $A^-_{px} = \{T = (s, p_x, u) \mid T \in S_i \text{ and } u \in F_i\}$ with the URIs $u \in F_i$ and a property p_x. Let $\alpha(p_x) = \frac{|A_{px}|}{|F|}$ be the fraction of URIs with the property p_x from all URIs in the frontier F. Let α_0 be the threshold value any property p_x must reach ($\alpha(p_x) \geq \alpha_0$) so that A_{px} is not removed from S_i in order to build S'_i. For the construction of inbound triples ($c_i = \leftarrow$), only A^-_{px} is relevant, as $|A^+_{px}| = 0$. For the construction of outbound triples ($c_i = \rightarrow$), only A^+_{px} is relevant, as $|A^-_{px}| = 0$. For the construction of the neighbourhood ($c_i = \leftrightarrow$), A^+_{px} and A^-_{px} are relevant and calculated separately.

The seed F_0 of a subset probably contains Wikidata items (URIs) of only a few classes. URIs of the same class likely have similar properties, so $\alpha(p_x)$ tends to be relatively large for most properties. However, when we iteratively construct the neighbourhood, the classes of the URIs $u \in F_i$ will quickly differentiate and $\alpha(p_x)$ decrease. Therefore, a constant value for α_0 might not be the best solution. Instead, there are arguments to both decrease and increase it with each construction step. We might decrease α_i to not exclude too many triples that could provide relevant information. We might also increase α_i to remove even more triples that could provide irrelevant information. Both can be done in a additive ($\alpha_{i+1} = \alpha_i \pm \beta$) or multiplicative ($\alpha_{i+1} = \alpha_i * \beta$) way. The decision to increase or decrease α_i is in its consequence a decision for a more dense, maybe minimal subset or a full-covering subset that doesn't want to miss out relevant data. The values α_0 and β are set by input parameters of the filter function. It is also possible to select $\alpha_0 = \beta = 0$ to disable the property filter.

With the proposed filter for rare properties, the function $S'_i = filter(S_i)$ specifies to:

$$S'_i = S_i \setminus \{T = (s, p_x, o) \mid \alpha(p_x) < \alpha_i(\alpha_0, \beta)\} \tag{3}$$

6.5 Frontier Considerations

With the data that describes the URIs of the frontier F_i collected and filtered (subset S'_i), the next construction step is due. Therefore, we need to select a new set of URIs F_{i+1}. Because we build a subset from a seed through the construction of the surroundings of the seed, the new frontier F_{i+1} consists of URIs found in the last construction step. These new URIs are adjacent to the last frontier F_i and can therefore be described using the set of adjacent URIs $N_{Si'}(F_i)$ from graph theory [5, 3].

However, not all adjacent URIs are proper for F_{i+1}. Let's say we expand through the complete neighbourhood. In the example of figure 1, we reach the URI *Berlin* with the first construction step. Therefore, the URI is part of the following frontier ($Berlin \in F_{i+1}$). The URI *CBF* is also included in that frontier ($CBF \in F_{i+1}$). After another construction step, we reach *Berlin* again in the outbound triple from *CBF*. This would mean that we process the neighbourhood of *Berlin* twice. Thus, we must only select unvisited URIs for the new set of URIs F_{i+1}. The set of visited URIs V collects all newly visited URIs after every construction step. V can then be excluded from F_{i+1}. In consequence, the sets F_i and F_{i+1} are disjoint: $F_i \cap F_{i+1} = \emptyset$.

We can also omit literals from the set F_{i+1} because they are never the subject of a triple and we are not interested in URIs with identical literal values as object (e.g. the year 1958 in fig. 1). Additionally, we can omit blank nodes because they can't be queried.

The example in figure 1 shows another scenario we have to consider for F_{i+1}. The displayed graph represents the subset we would like to receive as a result of our algorithm. The URI *Charité* serves as seed ($F_0 = \{Charité\}$). Initially, we are interested in all data related to the seed, so we construct the complete neighbourhood in the first step ($c_0 = \leftrightarrow$). However, out of all the URIs we discover with this construction, we know we only want to continue the construction in the next step with the successors, because the outbound relations of the seed are more important for our use case. In this case, the frontier must only consider the URIs $N^+_{Si'}(F_i)$ succeeding the set of URIs F_i. The preceding URIs $N^-_{Si'}(F_i)$ are the opposite option. To include this information in the input, we add two options for the sequence of expansion steps C: Only select predecessors ($\leftrightarrow^-$) or successors ($\leftrightarrow^+$) after constructing the complete neighbourhood. Therefore, the expansion options are $c_i \in \{\leftrightarrow, \leftrightarrow^-, \leftrightarrow^+, \leftarrow, \rightarrow\}$.

For a construction of inbound ($c_i = \leftarrow$) or outbound ($c_i = \rightarrow$) triples, the frontier also defines with $N^-_{Si'}(F_i)$ or $N^+_{Si'}(F_i)$ respectively because only those adjacent URIs exist.

In conclusion, the function $F_{i+1} = frontier(S'_i, c_i, F_i, V)$ specifies to:

$$F_{i+1} = \begin{cases} (N_{Si'}(F_i) \cap U) \setminus V & \text{if } c_i \in \{\leftrightarrow\} \\ (N^-_{Si'}(F_i) \cap U) \setminus V & \text{if } c_i \in \{\leftrightarrow^-, \leftarrow\} \\ (N^+_{Si'}(F_i) \cap U) \setminus V & \text{if } c_i \in \{\leftrightarrow^+, \rightarrow\} \end{cases} \tag{4}$$

The listings 1.4, 1.5, and 1.6 show SPARQL SELECT queries that select all URIs u that are adjacent ($c_i \in \{\leftrightarrow\}$) / predecessors ($c_i \in \{\leftrightarrow^-, \leftarrow\}$) / successors

($c_i \in \{\leftrightarrow^+, \rightarrow\}$) to an URI $f \in F_i$ respectively. The SPARQL queries don't exclude visited URIs $u \in V$.

Listing 1.4: SPARQL query that selects URIs u adjacent to $f \in F_i$ for the frontier F_{i+1} from S'_i

```
SELECT ?u
FROM Si'
WHERE { {$f ?p ?u .} UNION {?u ?p $f .}
FILTER (isURI(?u)) }
```

Listing 1.5: SPARQL query that selects URIs u that are predecessors of $f \in F_i$ for the frontier F_{i+1} from S'_i

```
SELECT ?u
FROM Si'
WHERE { ?u ?p $f .
FILTER (isURI(?u)) }
```

Listing 1.6: SPARQL query that selects URIs u that are successors of $f \in F_i$ for the frontier F_{i+1} from S'_i

```
SELECT ?u
FROM Si'
WHERE { $f ?p ?u .
FILTER (isURI(?u)) }
```

In the example of figure 1, the first construction step $c_0 = \leftrightarrow^+$ results in the frontier $F_1 = \{CBF, Berlin, VK\}$ (see section 6.3 for S_0). The following step only constructs outbound triples ($c_1 = \rightarrow$). The corresponding subset $S_1 = N_W^{+RDF}(F_1)$ returns the triples:

```
{(CBF locatedIn Berlin), (CBF inception 1958),
(CBF namedAfter BF), (Berlin population 3644826),
(Berlin contains Mitte), (VK locatedIn Mitte),
(VK inception 1906)}
```

The resulting frontier would be $F_2 = \{BF, Mitte\}$, however, the subset from figure 1 is already completed with $C = (\leftrightarrow^+, \rightarrow)$.

7 Conclusion

7.1 Summary

The paper introduces an algorithm $S = subset(W, C, seed(), filter())$ to create a demand-driven Wikidata subset. Therefore, we construct triples in multiple steps starting from a seed. Available options $c_i \in \{\leftrightarrow, \leftrightarrow^-, \leftrightarrow^+, \leftarrow, \rightarrow\}$ that sequence the construction offer flexibility. A filter reduces the number of triples to exclude irrelevant data.

7.2 Future Work

We are currently in the process of implementing the introduced algorithm. With a working implementation, we plan to construct a Wikidata subset of the more challenging domain of microelectronics and their supply chains.

Once we created multiple subsets, we need an evaluation to compare them. The clustering coefficient from graph theory seems like one possible parameter.

One advantage of Wikidata is its links to other data sources on the web. With techniques from link traversal query execution, we could follow the links and include data we find in the subset.

References

1. Batsakis, S., Petrakis, E.G., Milios, E.: Improving the performance of focused web crawlers. Data & Knowledge Engineering **68**(10), 1001–1013 (Oct 2009). https://doi.org/10.1016/j.datak.2009.04.002
2. Beghaeiraveri, S.A.H., Gray, A.J.G., McNeill, F.J.: Experiences of Using WDumper to Create Topical Subsets from Wikidata. In: Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021). p. 15 (Jun 2021)
3. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. Elsevier, New York, 5 edn. (1982)
4. Chakrabarti, S., van den Berg, M., Dom, B.: Focused crawling: a new approach to topic-specific Web resource discovery. Computer Networks **31**(11-16), 1623–1640 (May 1999). https://doi.org/10.1016/S1389-1286(99)00052-3
5. Dundar, P., Aytac, A., Kilic, E.: The common-neighbourhood of a graph. Boletim da Sociedade Paranaense de Matemática **23-32**(1), 23 (2017). https://doi.org/10.5269/bspm.v35i1.22464
6. Hartig, O., Bizer, C., Freytag, J.C.: Executing SPARQL Queries over the Web of Linked Data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2009. Lecture Notes in Computer Science, vol. 5823, pp. 293–309. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04930-9_19
7. Hartig, O., Freytag, J.C.: Foundations of Traversal Based Query Execution over Linked Data (Extended Version). In: HT '12: Proceedings of the 23rd ACM conference on Hypertext and social media. pp. 43–52 (Jun 2012). https://doi.org/10.1145/2309996.2310005
8. Mimouni, N., Moissinac, J.C., Vu, A.T.: Domain Specific Knowledge Graph Embedding for Analogical Link Discovery. International Journal On Advances in Intelligent Systems **13**(1&2), 140–150 (Jun 2020), https://hal-cnrs.archives-ouvertes.fr/hal-03052226
9. Olston, C., Najork, M.: Web Crawling. Foundations and Trends in Information Retrieval **4**(3), 175–246 (2010). https://doi.org/10.1561/1500000017
10. Stickler, P.: CBD - Concise Bounded Description (Jun 2005), https://www.w3.org/Submission/CBD/