WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

EFMD
EQUIS
ACCREDITED

Master Thesis

Towards analysing the evolution of community-driven (sub-)schemas within Wikidata

Nicola Pascal Krenn

Date of Birth: 02.11.1998
Student ID: 11828689

Subject Area: Information Business

Program: Digital Economy

Program Code: 066/960

Supervisor: Univ.-Prof. Dr. Axel Polleres and
Assoc.-Prof. Dr. Johannes Wachs

Date of Submission: 01.08.2023

Department of Information Systems and Operations, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

DEPARTMENT FÜR INFORMATIONS-
VERARBEITUNG UND PROZESS-
MANAGEMENT DEPARTMENT
OF INFORMATION SYSTEMS AND
OPERATIONS

Contents

List of Figures

List of Tables

Abstract

Since its inception, Wikidata has grown into a vast Knowledge Graph, housing extensive information across diverse topics. The Wikimedia Foundation, as the driving force behind Wikidata, aims at providing knowledge in a collaborative, community-driven manner, specifically through Wikidata as a central place for structured, easily attainable and semantically re-usable information. In this thesis, we analyse Wikidata specifically in terms of its community aspect, with respect to vocabularies used by different communities on different domains. We describe the datasets extracted for this purpose and provide re-usable scripts with adaptable parameters to (re-)create them. Our approach should enable researchers to carry out detailed analyses of Wikidata's communities and possibly their evolution over time, surpassing the capabilities of currently existing solutions. The primary outcome of our study comprises a preliminary analysis of two critical networks for community analysis, one focusing on the domains of entities user communities are contributing to, and the other focusing on vocabulary similarities. In summary, our preliminary analysis paints a picture of several highly distinct communities editing on special domains, therefore being almost exclusively responsible for its contents. The vocabulary similarities still require some work. To this point in time, the results are inconclusive. Looking ahead, we suggest enhancing the two networks with advanced filtering- and weighting algorithms. Additionally, we see immense value in further developing a "pipeline-approach", allowing easy parameter adaptations to identify optimal combinations of parameters, filters, and weighting algorithms. By exploring the intricacies of Wikidata's community dynamics and vocabulary utilization, this thesis contributes to a deeper understanding of collaborative knowledge creation...

1 Introduction

Wikidata has spiked in accumulative knowledge stored in the past years. It is now the most comprehensive community-driven aggregation of semantically stored information, holding data on more than 100 million concepts individually featuring the edits of over 560.000 editors [Vrandečić et al., 2023]. Wikidata is the collaborative Knowledge Graph (KG) brought to life by the Wikimedia Foundation in 2012. The Wikimedia Foundation is also the creator of the online collaborative encyclopedia Wikipedia. The main distinction of Wikipedia to Wikidata is that it stores semantic information, rather than just information.

Knowledge Graphs are in general a technology that is used to add context to data, originally developed by Google. They are mostly used to describe real-world entities using attributes and relationships [Cimiano and Paulheim, 2017]. Figure 1 illustrates a basic example of a Knowledge Graph focusing on the earth and the moon.

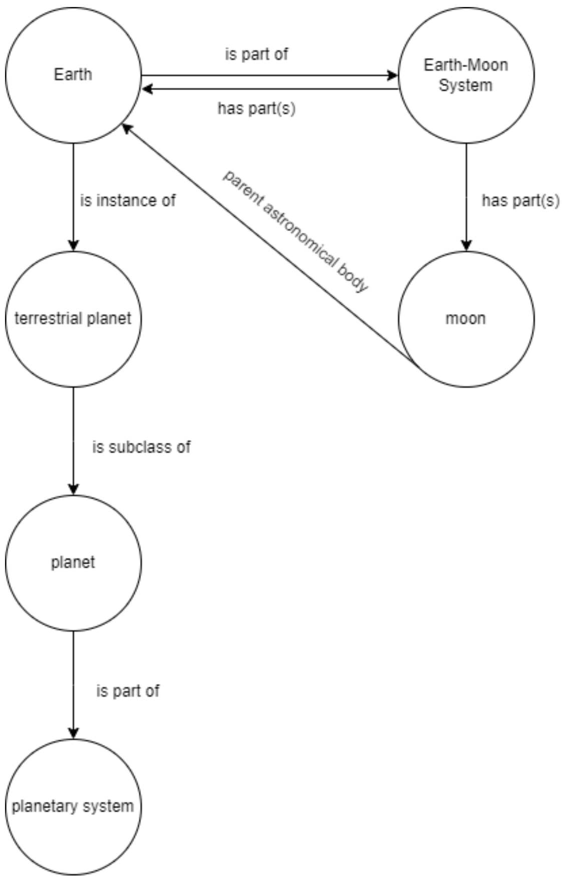
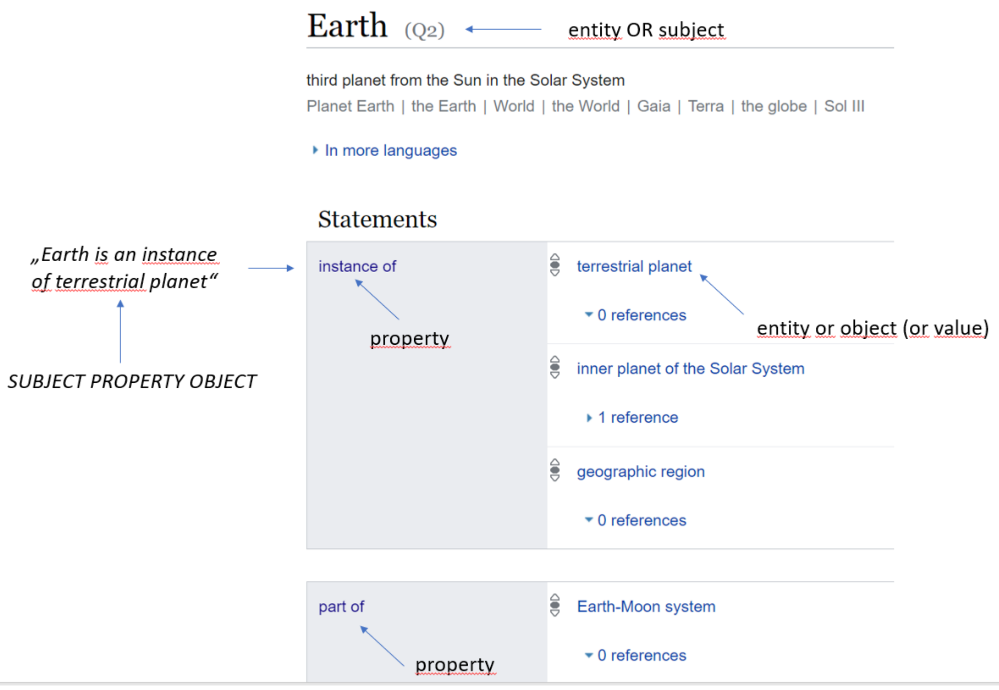


Figure 1: Basic Knowledge Graph example

The *entities* are the big circles describing real-world entities and the arrows illustrate the relationships, further referenced as *properties*. Note that the arrows are directed, always having a clear starting entity, the *subject*, and an ending point, the *object*.

To illustrate these terms further in a Wikidata example, Figure 2 shows the Wikidata page of one of the first real-world entities added to Wikidata, the planet we all live on[1].


Earth (Q2)

third planet from the Sun in the Solar System

Planet Earth | the Earth | World | the World | Gaia | Terra | the globe | Sol III

In more languages

Statements

instance of	terrestrial planet 0 references
	inner planet of the Solar System 1 reference
	geographic region 0 references
part of	Earth-Moon system 0 references

entity OR subject
„Earth is an instance of terrestrial planet"
SUBJECT PROPERTY OBJECT
property
entity or object (or value)
property

Figure 2: Wikidata Page for entity Earth (Retrieved on 3rd July 2023 from https://www.wikidata.org/wiki/Q2)

Augmented in the picture is the page title "Earth" with a "(Q2)" behind it. Q2 would be the identifier for the entity earth. Below that there are the statements, which add information and relation. The first statement shown can be translated to the sentence "Earth is an instance of terrestrial planet", where the sentence follows the order: subject, property, and object. The properties are used to provide context and relation to the object, whereas the object might again be an entity or just a value (e.g. simple text). Every entity has a unique identifier starting with a Q, and every property has a unique identifier starting with a P. The highlighted "instance of" property has the identifier P31 for example.

[1]https://www.wikidata.org/wiki/Q2

The base idea of Wikidata is that users can contribute entities and then use properties to relate information or other items to them.

Without going into more detail on how Wikidata works or how the knowledge is structured, we will first quickly point out two points concerning Wikidata that are important to understand the goals of the thesis:

Wikidata has no fixed ontology. This might be explained more simply as: there is no fixed structure. It relies completely on its users' contributions to ensure some level of structure and completeness. Every piece of knowledge might be structured in completely different ways, even though expressing the same thing [Farda-Sarbas and Müller-Birn, 2019]. However, community guidelines and community projects try to establish consensus on how to do certain things (and how not to).

Everybody can edit (almost) everything in Wikidata. Wikidata allows anonymous edits, as well as bot edits, and seldomly restricts edits. Again, a wrongful/harmful contribution or a contribution in violation of community guidelines might still be performed. To change/delete that, other users have to fix it.

Summed up, both points illustrate that Wikidata is really fully community-driven, with all the pros and cons that arise with this decision. It is reliant on the continuous improvement of its contents, which on a time scale, can be called "evolution".

The (sub-)schema mentioned in the title of the thesis relates to a concept incepted by [Baroncini et al., 2022] derived from [Piscopo and Simperl, 2018]. It can be defined as "the set of properties and the set of items that are used as classes, i.e. those that are the object of instance of (P31), or subject/object of subclass of (P279)" [Baroncini et al., 2022]. Leaving out the edge cases where items (= entities) are used as classes, the schema can be naively defined as the set of properties used. The schema is therefore very important as it can be used to identify which information is stored and which relations are used. We coined another term - *vocabulary*, whereas the set of all vocabularies used is the schema. The difference between property and vocabulary is that it may include additional information on the set of entities that are used as classes, e.g. the vocabulary for property P50 might just be defined as P50, but for P31 it includes also the object e.g. Q500. This gives us the freedom to define our own vocabulary definition and avoids misunderstandings. Figure 3 illustrates this.

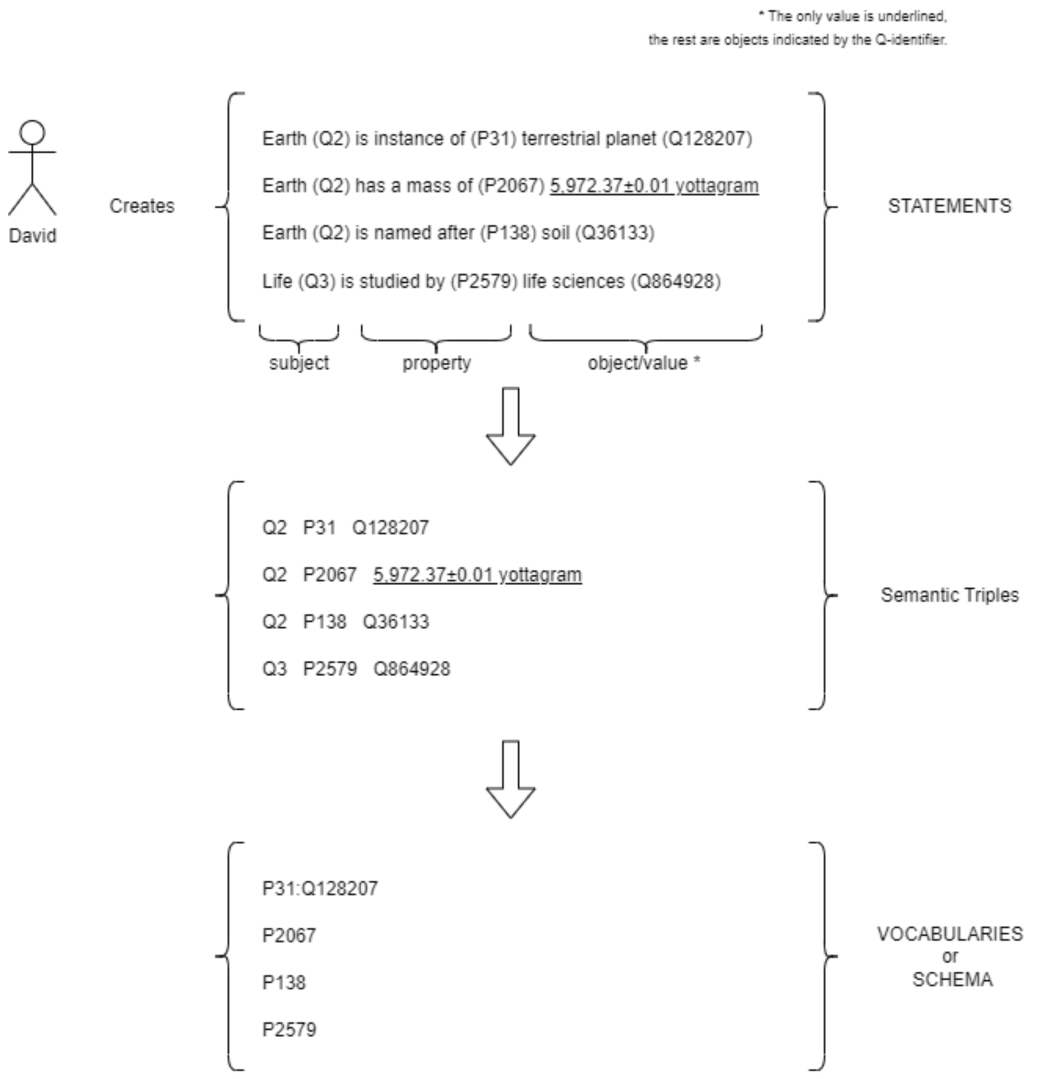

* The only value is underlined,
the rest are objects indicated by the Q-identifier.
David
Creates
Earth (Q2) is instance of (P31) terrestrial planet (Q128207)
Earth (Q2) has a mass of (P2067) 5,972.37±0.01 yottagram
Earth (Q2) is named after (P138) soil (Q36133)
Life (Q3) is studied by (P2579) life sciences (Q864928)
STATEMENTS
subject
property
object/value *
Q2 P31 Q128207
Q2 P2067 5,972.37±0.01 yottagram
Q2 P138 Q36133
Q3 P2579 Q864928
Semantic Triples
P31:Q128207
P2067
P138
P2579
VOCABULARIES or SCHEMA

Figure 3: Schema and Vocabulary definition

1.1 Research Questions and Motivation

Moving on from the introduction, we know that Wikidata's entities and properties are the core information structure (excluding values or other data types). The *schema* can then be characterized as the set of relations used to model some information, whereas a single element of the *schema* is called *vocabulary*. The users are not technically prohibited from using any *vocabulary* other than by some community guidelines; everybody can edit everything. This leads to the fact that Wikidata relies on users enforcing its guidelines to get to a useful *ontology*.

[Baroncini et al., 2022] proposed an approach to analyse the evolution of community-driven (sub-)schemas within Wikidata, focusing on one of the more interesting and yet mostly (to the best of our knowledge) untouched questions: "How did Wikidatas (sub-)schema emerge and who is responsible for it?". It outlines a conceptional approach to do exactly that, by proposing a structured approach with the key steps proposed being:

1. Identify the domain
2. Identify the community
3. Identify the community-driven (sub-)schema
4. Analysing the schema and the community (additionally over time) and then comparing the schemas among communities

The *domain* can be described as a topic, for example, Computer Science or Physics. In the paper, the domain definition is proposed to be executed manually by experts.

This direction of research tackles many dimensions of Wikidata's evolution, the community aspect as well as the ontology/schema/vocabulary aspect, even when skipping the evolution aspect. It could lead to resourceful insights into Wikidata and its evolution, possibly uncovering cues to make Wikidata even better. It could also uncover implicit differences in the way people contribute to Wikidata, specifically in the region's domain and usage of vocabulary.

Contributing to the way along the use case, this thesis answers the overarching research question: "Can we show a topicwise modularisation of Wikidata by clustering users by their used vocabulary?". The question also holds an implicit hypothesis, namely that there should be topicwise modularisation, mainly because different topics/domains describe different entities and relations. For example, there would be no value in using a property "date of birth" for buildings. However, comparable domains like for example Italian museums and Greek museums should, in theory, almost use the same vocabularies and not have significant differences. Differences in comparable domains would lead to a reduced value of the information stored, as it would reduce the expressiveness and comparability.

A more thorough example is: two users might use different properties to describe the cost or price of an object. Wikidata features two properties:

- cost (P2130)
- price (P2284)

Both could potentially be used as synonyms[2]. A person wanting to compare prices for public buildings in Tokyo versus New York might just query for one of the two properties and therefore only get results for one of the two properties. This is per se no problem, but if the experts on buildings in Tokio use a different property/vocabulary for the information than the experts on buildings in New York, the retrieved information might be not as valuable as it could be. In a research example, this might be no problem, since the researcher will probably react to this quickly if it happens at a large scale (if he notices it). However, in a (semi-)automated setting where Wikidata is used as a data source for e.g. an unsupervised AI like ChatGPT, this leads to problems, as the relation used is not the same. Therefore, the similarity of vocabulary is an important measure, especially within certain comparable domains. Additionally, the focus on the users in terms of a community structure might bring additional results, since different people might use different terms to express the same thing. And this might uncover possible community-wise modulation.

A quite similar phenomenon was studied by [Krabina and Polleres, 2021]. They focused on certain financial properties within Wikidata and found that “Many items exist, some with very ambiguous or unclear meaning.”. Their work demonstrated a lack of uniformity and ongoing discussions about how to represent certain aspects in Wikidata.

To sum up, two quite separate streams are examined, illustrated by two maybe less formal research directions:

1. domain similarity of users
2. vocabulary/schema similarity of users

[2]Search result on properties for “cost”: `https://www.wikidata.org/w/index.php?search=cost&title=Special:Search&profile=advanced&fulltext=1&ns120=1`

Direction 1 holds enormous value for the use case described by [Baroncini et al., 2022], since they rely on manual expert definitions of the domain. The proposed clustering might eliminate this need and might enable research to use the unsupervised clustering for various applications. It also uncovers whether users really have some particular focus on some domains, which might also be seen as a starting hypothesis.

Direction 2 is then more focused on the use case of looking into the schemas and comparing them, on a per-user basis. The results might uncover vast differences in the vocabulary used by different communities, which could possibly be a bad thing for Wikidata as a whole. It can lead to cues whether vocabulary is somewhat standardized among users as well.

The specifics regarding the similarity aspect and measure are introduced in section 5, after getting to know more about the data we are looking at and the network analysis approach. However, the similarity measure might just be simply viewed as the weight of users A and Bs jointly used vocabulary or entity, where the weight is high if they use the same vocabulary/entity a lot. The entity here can be viewed as part of a certain domain. In general, this approach of omitting massive amounts of data and looking at the whole edit history, focusing on the simplified results, is novel (to the best of our knowledge). This has multiple reasons, one is the fact that general knowledge graph-focused research does not like omitting data in the way Network Science usually does. However, sometimes we need to just simplify data to be able to look into complex systems like editor communities.

Overall, both directions could then be combined in order to answer the overarching question: "Can we show a topicwise modularisation of Wikidata by clustering users by their used vocabulary?", adding valuable information to the current Wikidata-focused research. Summed up, the research model might be formulated as follows:

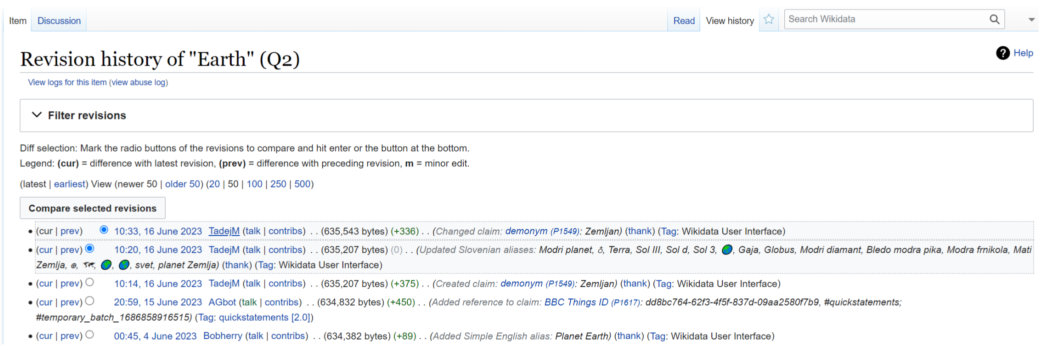
- Overarching Research Question: "Can we show a topicwise modularisation of Wikidata by clustering users by their used vocabulary?"
  - Direction 1 - domain similarity of users
    - RQ: Can we find a community-clustered structure that maps users to domains?

    * Hypothesis: Users have a very distinct focus on one or several domains

  - Direction 2 - vocabulary/schema similarity of users
    * RQ: Can we find a community-clustered structure that maps users according to vocabulary similarity?
    * Hypothesis: Users have a very distinct vocabulary, but it can be put into a community structure

1.2 Approach

For tackling the proposed research question, there is an inherent need for data on what each of the users contributed to Wikidata. In the web application, each user can look at the revision history, which is quite synonymous to edit history. Figure 4 shows the revision history of entity earth. In each revision, a user might add/change/delete multiple *statements* and other data.



Item Discussion Read View history Search Wikidata

Revision history of "Earth" (Q2) Help

View logs for this item (view abuse log)

Filter revisions

Diff selection: Mark the radio buttons of the revisions to compare and hit enter or the button at the bottom.
Legend: **(cur)** = difference with latest revision, **(prev)** = difference with preceding revision, **m** = minor edit.

(latest | earliest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

Compare selected revisions

- (cur | prev) 10:33, 16 June 2023 TadejM (talk | contribs) . . (635,543 bytes) (+336) . . *(Changed claim: demonym (P1549): Zemljan)* (thank) (Tag: Wikidata User Interface)
- (cur | prev) 10:20, 16 June 2023 TadejM (talk | contribs) . . (635,207 bytes) (0) . . *(Updated Slovenian aliases: Modri planet, ♁, Terra, Sol III, Sol d, Sol 3, 🌍, Gaja, Globus, Modri diamant, Bledo modra pika, Modra frnikola, Mati Zemlja, ⊕, 🗺, 🌍, 🌍, svet, planet Zemlja)* (thank) (Tag: Wikidata User Interface)
- (cur | prev) 10:14, 16 June 2023 TadejM (talk | contribs) . . (635,207 bytes) (+375) . . *(Created claim: demonym (P1549): Zemljan)* (thank) (Tag: Wikidata User Interface)
- (cur | prev) 20:59, 15 June 2023 AGbot (talk | contribs) . . (634,832 bytes) (+450) . . *(Added reference to claim: BBC Things ID (P1617): dd8bc764-62f3-4f5f-837d-09aa2580f7b9, #quickstatements; #temporary_batch_1686858916515)* (Tag: quickstatements [2.0])
- (cur | prev) 00:45, 4 June 2023 Bobherry (talk | contribs) . . (634,382 bytes) (+89) . . *(Added Simple English alias: Planet Earth)* (thank) (Tag: Wikidata User Interface)

Figure 4: Wikidata Revision History for entity Earth (Retrieved on 16th June 2023 from `https://www.wikidata.org/w/index.php?title=Q2&action=history`)

For the research question, we essentially need information on the user and the statements that he/she added, along with a potentially useful timestamp of the edits. This data can altogether be found in the shown revision history.

In general, Wikidata offers a lot of APIs and ways to query Wikidata. For full images, the so-called *dumps*[3] are the most preferred choice, as they contain all the data currently stored in Wikidata. There are multiple dump formats specifically designed for different use cases, each being created and uploaded periodically. For our use case, the "pages-meta-history" branch of

[3] `https://dumps.wikimedia.org/`

the dumps is most interesting, as it contains the full revision history in JSON BLOB format. With this, we are able to calculate the added/changed/deleted statements. The dump in use contains all the revisions up until the First of January 2023.

Moreover, a general analysis of the gathered and transformed data is in order, also adding potentially interesting statistics to the ecosystem, before moving on to the main analysis.

For the main analysis, network analysis techniques in combination with community detection techniques are used in order to get to two networks. Specifically, one network maps the domain similarity of users, and one maps the vocabulary similarity of users. Note that each of the networks is operated in a "community-first" way, where domain similarity and vocabulary similarity are heuristically defined by comparing the users' contributions and the then applied community detection algorithm clusters the communities, also using the defined similarity measure.

Other ways to do this like classifying the domains first and then applying a community detection algorithm would also be a viable option. However, developing such a way to classify them in a "community-last" manner would be much harder to do and the "community first" approach offers potential additional interesting (sub-)domains that could emerge. These then potentially give clues to further points of analysis on user behaviour in Wikidata. Specifically, looking at the emerging clusters from a logical reasoning standpoint might uncover additional measurements to include for the similarity measure definition (e.g. add additional information to some vocabulary like the mentioned object in the case of P31 property, see Figure 3).

1.3 Structure of the Thesis and Contributions

In chapter 2, we describe the preliminaries on Wikidata and the current research on Wikidata in multiple directions with a focus on the use case.

In chapter 3, we describe the steps taken in order to get to the data needed for the analysis:

1. Eliciting which data we need
2. Parsing of the Wikidata dump
3. Creating the framework

The corresponding data processing framework and the data it creates are among the main contributions of this thesis, as both may be used for other use cases or more thorough analyses of the same use case.

In chapter 4, we analyse said dataset from various angles in order to get some additional interesting information, which adds to the big picture.

In chapter 5, we conduct the network analysis and explain the used methods including background-knowledge to add information regarding the choices we took and conclude the analysis.

2 Preliminaries

2.1 Wikidata

This chapter provides additional information on Wikidata even exceeding the base definitions in the Introduction, in order to clearly define them.

Since its creation around 7 years ago, Wikidata has grown in importance to many applications in many fields. Being a community-driven project to put data in a resourceful web interface, we expect many more applications to emerge as time progresses. The recent spike of Artificial Intelligence (AI) projects, now enabling also non-expert users to use them, also gives new light on the efforts taken at Wikidata. Specifically, the data and information stored in Wikidata might be used to automatically fact-check the outputs of Large Language Models like ChatGPT, leading to a generally more fact-based output. An early example of this would be [Mountantonakis and Tzitzikas, 2023], showing a way of fact-checking ChatGPTs answers in an automated setting.

In general, Wikidata can provide huge amounts of data and knowledge, being verified and up-to-date by community experts watching over the published information. Wikidata itself states on the 11th of May 2023, 1.893.680.660 edits have been made and that there are currently 24.253 active users[4].

One particular reason for the usefulness of Wikidata is that it is possible to extract huge amounts of data in a semantic way, by querying the underlying knowledge graph. This is also the main distinction compared to Wikipedia, namely that the knowledge is stored semantically. Since many

[4]https://www.wikidata.org/wiki/Wikidata:Statistics retrieved on the 11th of May 2023

KGs are focusing on different types of information or different types of contributions to the KG, the semantic web community defined multiple standards. These were set in order to e.g. ensure interoperability and drive forward to the overall goal of a semantic web. The main standards contributing to this direction are:

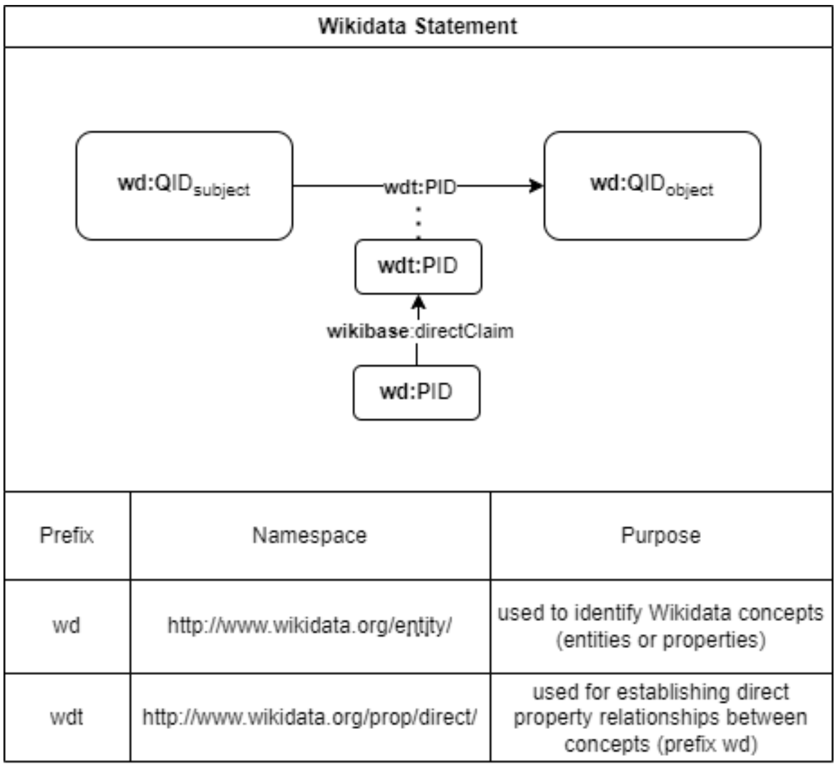
- RDF - the Resource Description Framework
- SPARQL - **S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage (SPARQL)
- RDFS - RDF Schema and the corresponding Web Ontology Language (OWL)

Each of the following paragraphs will quickly sum up the main points of each of these standards and the implications for Wikidata.

RDF - the Resource Description Framework
The RDF standard has been published in 2014 by a working group of the W3C. At its base, it is a framework for expressing information about resources, whereas resources can be anything from people to abstract concepts and physical objects. The framework's main goal is to ensure interoperability of resources between different sources, such as knowledge graphs like Wikidata, DBPedia, or YAGO. Wikidata implements the RDF standard using different so-called RDF namespaces, in order to represent different aspects of the same property. In Figure 5, you can see the basic way Wikidata incorporated the RDF namespaces in the concrete example of a statement. The prefix *wd* is used to identify Wikidata concepts and the prefix *wdt* is used to directly relate properties to relationships. The illustration is incomplete from an RDF standpoint but includes everything needed for the use case.

Again, the so-called triple-structure is very visible, meaning that a statement and therefore relation between entities is established by using three parts - subject, property, and object. This semantic triple, or also synonymous RDF-triple, lies at the heart of the RDF data model. The namespaces are then on the one hand used as a resource identifier as well as to give additional information to the resources. However, for the use case of the thesis, it is only necessary to understand the basic concept of the triples, as we will only focus on the basic statement/triple edits.

SPARQL - SPARQL Protocol And RDF Query Language (SPARQL)
Even though KGs in general can be considered No-SQL databases, practical



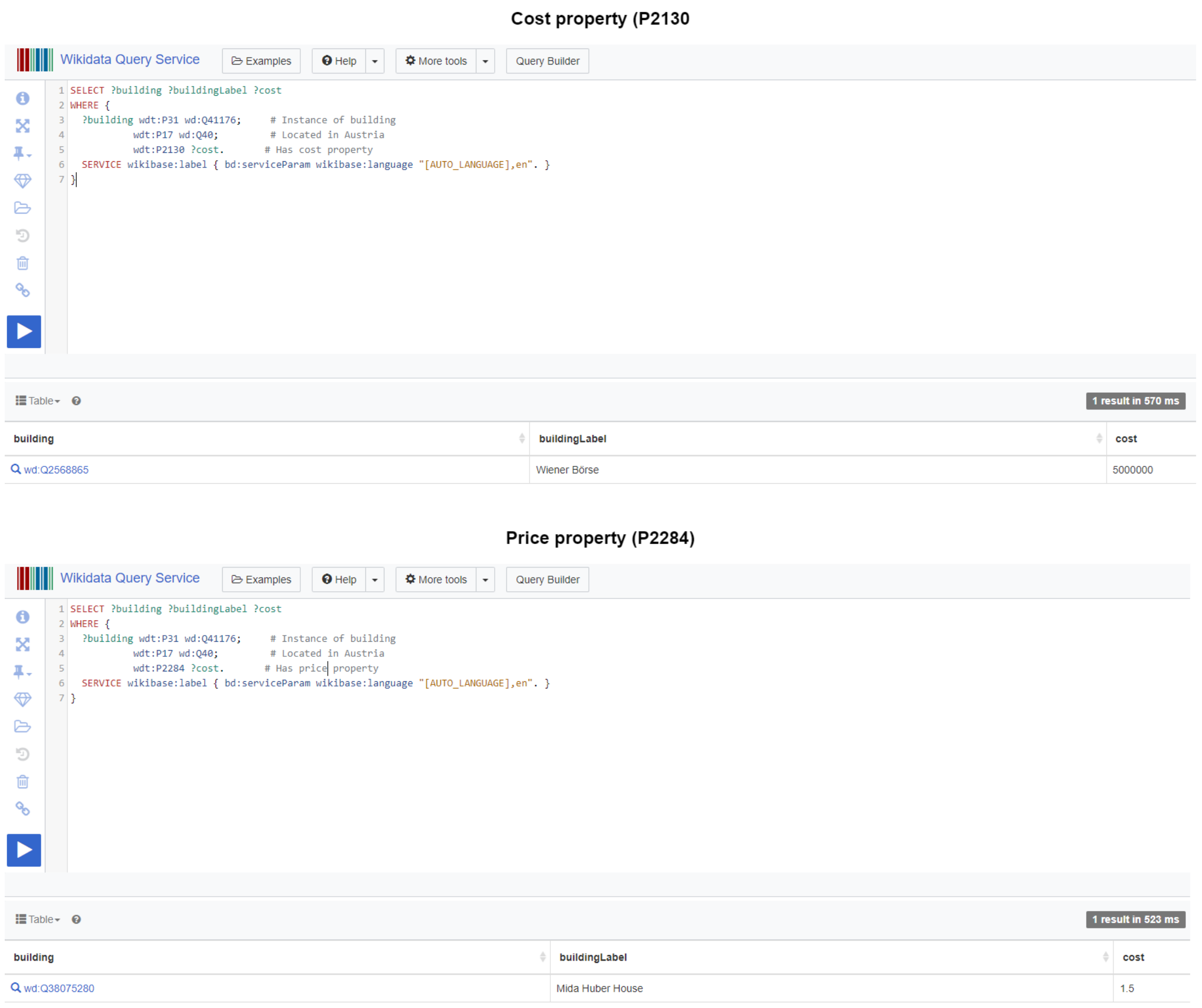
Prefix	Namespace	Purpose
wd	http://www.wikidata.org/entity/	used to identify Wikidata concepts (entities or properties)
wdt	http://www.wikidata.org/prop/direct/	used for establishing direct property relationships between concepts (prefix wd)

...

Figure 5: The Wikidata meta-model regarding statements in the RDF context

languages like SPARQL provide a way to query them [Hogan et al., 2021]. Wikidata also provides an API for queries using SPARQL[5]. SPARQL queries run on Wikidata also include the aforementioned RDF namespaces. Moving on with the running example on prices/cost from the introduction, Figure 6 shows the two queries and highlights the problem even more.

[5]e.g. via Web-API: `https://query.wikidata.org/`



Cost property (P2130

```
SELECT ?building ?buildingLabel ?cost
WHERE {
  ?building wdt:P31 wd:Q41176;     # Instance of building
            wdt:P17 wd:Q40;        # Located in Austria
            wdt:P2130 ?cost.       # Has cost property
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

building	buildingLabel	cost
wd:Q2568865	Wiener Börse	5000000

Price property (P2284)

```
SELECT ?building ?buildingLabel ?cost
WHERE {
  ?building wdt:P31 wd:Q41176;     # Instance of building
            wdt:P17 wd:Q40;        # Located in Austria
            wdt:P2284 ?cost.       # Has price property
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

building	buildingLabel	cost
wd:Q38075280	Mida Huber House	1.5

Figure 6: cost and price property SPARQL query example using query.wikidata.org

On the top of Figure 6, you can see the results for property cost (P2130) and on the bottom for price (P2284). The query is designed for buildings in Austria (Q40) exclusively, to make the queried amount of data smaller and to illustrate this more easily. The results differ, whereas the "correct" vocabulary to use would be the cost (P2130), which returns the "Wiener Börse", meaning the Vienna Stock Exchange. The cost listed there is historic in a not Wikidata-recommended value structure (at the point in time of querying, the 21st of June 2023.). The second property price (P2284) returns only one building, to be exact, a museum in Austria. The price is listed as 1.5. This refers to the entry-ticket price of the museum. One might think about if this can be considered "correct" usage, however, at the time of writing there is no quick way to determine whether this is considered correct by community guidelines and accepted into practice. Because of this, it is arguably a very good example of the problem.

When switching the country Austria (Q40) to the United States of America (Q30), the results get a bit longer than just one building. For example, at the time of querying, the Manhattan Center[6] lists a price property (P2284) of 3 million USD. This may or may not be considered "wrong". The Grand Palace Hotel then lists a cost property (P2130) of 10 million dollars. Note that all prices listed have an additional annotation of a point in time in order to annotate this additional important information. There is also one particularly interesting case in the results, the Hotel Manger, which is labeled as a former Hotel in Boston[7]. The same hotel has both of the named properties, each stating different amounts of USD. This may or may not have a reason, but it illustrates the problem very well.

Now knowing more about the problem of diverse properties in use to describe the same things, we will quickly go into RDFS and OWL, which are essentially trying to standardize this problem.

RDFS and OWL

RDFS and OWL are both data modeling languages for describing RDF data. In exact terms, OWL builds on RDFS and provides more expressivity. Both are ontology languages building up on RDF, being used to model and standardize the RDF structures. The W3C characterizes the RDF Schema (RDFS) as: "RDFS is a vocabulary, in RDF, that explains how nodes of

[6] Retrieved on the 21st of June 2023 from `https://www.wikidata.org/wiki/Q3844593`

[7] Retrieved on the 21st of June 2023 from `https://www.wikidata.org/wiki/Q30644484`

a graph relate"[8]. This relates to the content provided in section 1, as we defined the terms schema and vocabulary ourselves, meaning Wikidata does not use a standardized schema like RDFS or OWL. However, it is important to note, that there are already defined standards, but Wikidata does not use them.

2.1.1 Wikidata Data Model

In order to be able to understand the contents of Wikidata and its data model, the help page `https://www.wikidata.org/wiki/Help:Wikidata_datamodel` offers information on the exact data Wikidata stores and how it relates. In Figure 7, the main data model image as of the 3rd of July is displayed.

The red marks relate to the focus of the thesis, as the claims (also called: statements) are the part of Wikidata we look at. The model begins with an entity, pictured in the left top corner, which then has labels, aliases, descriptions, claims, and sitelinks. The only relevant part here is the claims, one of which can have an id, rank, type, a mainsnak, qualifiers-order, qualifiers, and references. We here again only focus on the mainsnak, which contains the snak and therefore the data that forms the triple statement. Note that there are some edge cases we will explain, i.e. explain what we decided to do with them in the course of the analysis and preprocessing.

As said, the snak is the most interesting part, because it stores the property and the object (or value). In the picture, the snaktype is illustrated to either be a value / somevalue or novalue type. However, the data is depicted by the datavalue, which is illustrated in Figure 8, just as another part of the above image.

In this image, it is depicted, that only one of the possible datavalue types is a Wikidata entity. The rest relates to other values, like for example coordinates. The important part here is to notice that even though the rest has different types, we can view the available datatypes as Wikidata entity reference or just basically text-values, as a coordinate, or also just images can be viewed and differentiated as simple text. This is a simplification of great potential influence on Data Processing because handling all the datatypes would take time and resources.

Summed up, the interesting parts to take away from this are: there are claims that always have a property and the object may have different values or may reference to another Wikidata entity. This is the basic information that one needs in order to understand the use case and the steps taken to

[8]Found here in the W3C wiki: `https://www.w3.org/wiki/RDFS`

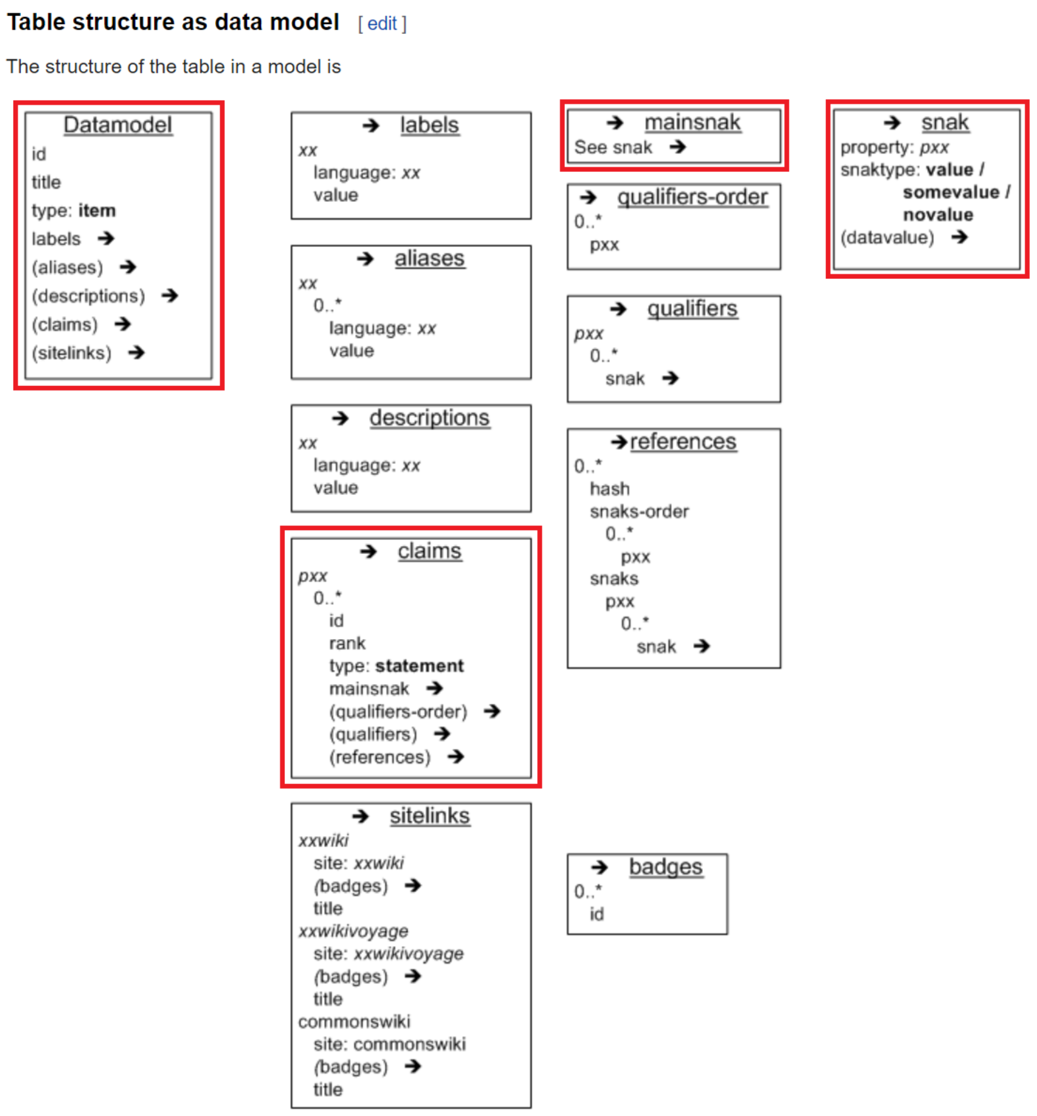


Table structure as data model [edit]

The structure of the table in a model is

```
Datamodel
id
title
type: item
labels ➔
(aliases) ➔
(descriptions) ➔
(claims) ➔
(sitelinks) ➔

➔ labels
xx
  language: xx
  value

➔ aliases
xx
  0..*
    language: xx
    value

➔ descriptions
xx
  language: xx
  value

➔ claims
pxx
  0..*
    id
    rank
    type: statement
    mainsnak ➔
    (qualifiers-order) ➔
    (qualifiers) ➔
    (references) ➔

➔ sitelinks
xxwiki
  site: xxwiki
  (badges) ➔
  title
xxwikivoyage
  site: xxwikivoyage
  (badges) ➔
  title
commonswiki
  site: commonswiki
  (badges) ➔
  title

➔ mainsnak
See snak ➔

➔ qualifiers-order
0..*
  pxx

➔ qualifiers
pxx
  0..*
    snak ➔

➔references
0..*
  hash
  snaks-order
    0..*
      pxx
  snaks
    pxx
      0..*
        snak ➔

➔ badges
0..*
  id

➔ snak
property: pxx
snaktype: value /
          somevalue /
          novalue
(datavalue) ➔
```

Figure 7: Wikidata Datamodel (Retrieved on 3rd July 2023 from https://www.wikidata.org/wiki/Help:Wikidata_datamodel)

gather the data. In the next section, further relevant information on edge cases is depicted.

2.1.2 Relevant Further Information

Concluding the relevant information on Wikidata for the use case, two edge cases are still untouched, which need to be defined beforehand. The claims from Figure 7 list a rank, which in full is called a statement rank. Every

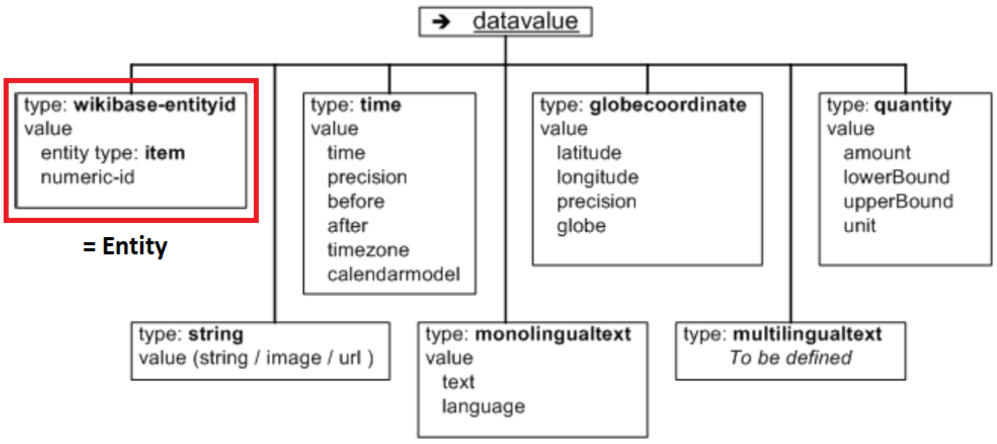


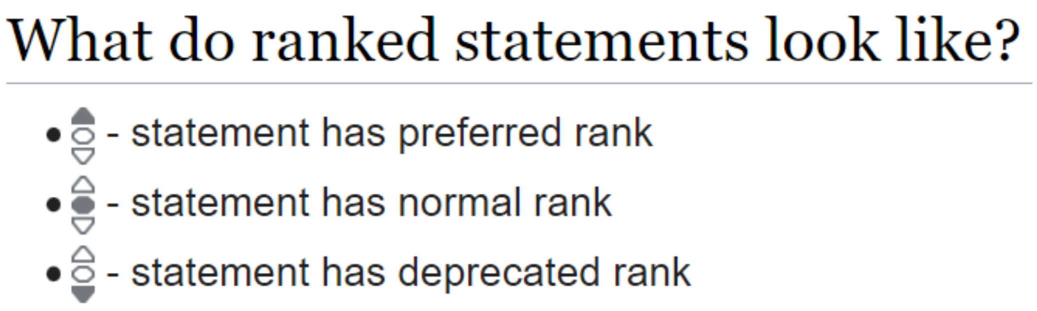
Figure 8: Wikidata Datavalue (Retrieved on 3rd July 2023 from `https://www.wikidata.org/wiki/Help:Wikidata_datamodel`)

statement has a rank and this is explained in the following paragraph. Moreover, not every entity might be able to be edited directly, which refers to another problem of duplicate entities, which Wikidata solved through *Redirect Items*. *Redirect Items* and their implications are explained in the second paragraph below.

Statement Ranks Wikidata features 3 statement ranks in general. *normal statements* are exactly that, normal statements. All statements can be voted by any user, in order to a) identify the most actual and correct statement in case of dispute in a crowd-driven manner and b) in order to deprecate old statements that are no longer true, but not lose the information in comparison to a mere deletion of the statement. Users therefore decide, whether they think a statement is "good" or "bad" in the sense that it is truthful and useful or not, and leave a vote. Wikidata then determines if the votes make the statement a *preferred statement* (a lot of upvotes), or a *deprecated statement* (a lot of downvotes).

Figure 9 shows the three different types of statement ranks in the user interface of Wikidata, where the first statement is a *preferred statement*, the second one is a *normal statement* and the third one is a *deprecated statement*. Note that each property may hold multiple statements, regardless of the type. More on this can be found on the help-page of Wikidata concerning Ranking[9]. In addition, to show the meaning of this a little bit more concretely, Figure 10 shows an example of such statement ranks in action.

[9] `https://www.wikidata.org/wiki/Help:Ranking` retrieved on the 15th of May, 2023


What do ranked statements look like?

- statement has preferred rank
- statement has normal rank
- statement has deprecated rank

Figure 9: Wikidata Statement Ranks (Retrieved on 18th July 2023 from https://www.wikidata.org/wiki/Help:Ranking)

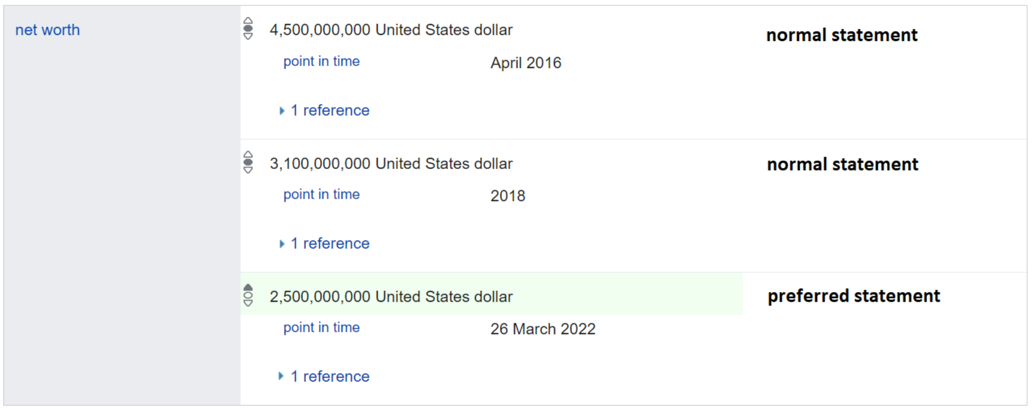

net worth	4,500,000,000 United States dollar point in time April 2016 1 reference	normal statement
	3,100,000,000 United States dollar point in time 2018 1 reference	normal statement
	2,500,000,000 United States dollar point in time 26 March 2022 1 reference	preferred statement

Figure 10: Wikidata net worth statements Donald Trump (Retrieved on 3rd July 2023 from https://www.wikidata.org/wiki/Q22686)

The picture shows the Wikidata entity page of Donald Trump, the former president of the United States of America. His net worth has been an example of great public dispute, with lots of information going around and high public interest (for example, see https://www.nytimes.com/2022/08/02/nyregion/trump-letitia-james-deposition-lawsuit.html). As the picture shows, there are 3 statements, one being a *preferred statement* and two being *normal statements*. The most recent information is marked as a preferred statement, which makes sense since this is probably what is most important. The other two are past values from given references. Hypothetically, if one of the net worths would have been corrected by whatever publicly available source, one of the *normal statements* could also be turned into a *deprecated statement*, as there is newer information and the old statement becomes incorrect, although being thought of as correct for a long time. This gives Wikidata and its contributors the opportunity, to still depict what happened in reality. Note that a net worth without source or any justification would not be seen as a *deprecated statement*, but should just be deleted in general, as it has no real value or basis of expected truth.

Anticipating the upcoming Data Processing framework explanation and the following decision on how to handle the statement ranks:
The statement ranks hold value of their own, and the XML dumps as well contain this information. However, it is an edge case, and the processing can take care of this in multiple ways. We decided to count *deprecated statements* as deletions, therefore not really valuable for the use case. *Normal statements* and *preferred statements* on the other hand are seen as the same, as the fact that the statement is preferred does not add much value to the analysis. However, since every statement is added as a *normal statement* at the time of creation, the statement rank is neglectable. The only implication is that a statement turning to a *deprecated statement* is viewed as a deletion (which we do not look at in the analysis).

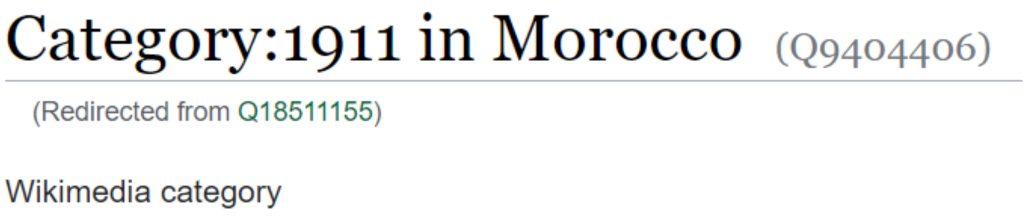
Redirect Items Redirect items handle a main problem of such large-scale knowledge graphs: Duplicates. If we detect that two or more entities describe the same real-world subject, this is problematic from various angles. In order to solve the duplicate issue, keeping all the information that we have on it is the first key. This can be achieved by merging the duplicate items into one. But another problem is that there is a need to decide on one entity that stays and there is a need to handle the statements that now reference to a possibly deleted entity.

Wikidata's solution for this is to make one of the entities that are merged a *Redirect Item*, which merely redirects to the correct item. This ensures that identifiers are stable and otherwise broken references stay intact. More information on this can be found on the Wikidata help page concerning Redirect Items[10]. When a user visits a Redirect Item, it is just directly relayed to the now merged item, the same happens with SPARQL queries accessing *Redirect Items*.

Take for example the two items Q18511155 and Q9404406 from the current Wikidata Redirects Help page[11]. Both refer to a category "1911 in Morocco", so they can be considered direct duplicates. As indicated by the identifiers, they probably were created at very different points in time. Both

[10] https://www.wikidata.org/wiki/Help:Redirects retrieved on the 15th of May, 2023

[11] https://www.wikidata.org/wiki/Help:Redirects


Category:1911 in Morocco (Q9404406)

(Redirected from Q18511155)

Wikimedia category

Figure 11: Redirect Item "1911 in Morocco" (Retrieved on 3rd July 2023 from https://www.wikidata.org/wiki/Q18511155

entities probably had multiple other links to other entities. Instead of taking the contents of one entity and deleting the other one, Wikidata merges the items' contents and makes one item a "Redirect Item", which then is only used as a pointer to the "real" item. You can see that an item is a redirect item as highlighted in the following screenshot in Figure 11.

To again put it in concrete terms, a user visiting the *redirect item* Q18511155 through URL https://www.wikidata.org/wiki/Q18511155 gets redirected to the page with URL https://www.wikidata.org/wiki/Q9404406. A user visiting the URL for the non-redirect-item Q9404406 would not see that Q18511155 is a redirect item to this entity, but will see the contents that were merged to the item at the time.

In our use case, redirect items are not really focus of the work, but they are an edge case. Redirect items are present in the XML dumps, and are indicated. User "Laddo" merged the two mentioned items on November 11th in 2014. So we have a file on the merged item containing its individual contributions until 2014 and a file on the merge target item, where on the same date user "Laddo" added the now merged information.

Anticipating the upcoming Data Processing framework explanation and the following decision on how to handle the redirect items:
It would be an option to handle redirect items directly, however, this raises a lot of questions. Therefore, we decided to cut redirect items out of the analysis process. In the way that the dumps are structured, this means that the information that is merged from one item to another is lost/ignored. Entity Q9404406 does not appear to have the information that was merged from the redirect item.

It is also worth mentioning that the entity in the picture is not a classic entity describing a real-world entity, but a Category. This has no real effect on the use case, as it can be logically identified as an abstract entity in the sense that it still has vocabulary and schema relating it to other entities.

2.2 Relevant Work

Before going into the main sections of determining what is needed in terms of data and what analysis components we designed, giving a little introduction to the previous research done in more relevant directions is in order. Note that section 9 in the end then summarizes some of the other maybe less focused literature.

Data Processing

As already mentioned, one key aspect of the paper is the creation and publication of the tools to create edit history data in a more general setting, and the data we created for our use case. Therefore, we looked at literature describing such frameworks or solutions.

Pellissier Tanon et al. worked on a more sophisticated solution to handle the data. They developed a SPARQL Endpoint and leveraged the "pages-meta-history" dump file to a) provide a way to query Wikidata's different revisions and b) query the edit history [Pellissier Tanon and Suchanek, 2019]. The previously publicly available SPARQL endpoint including GUI has been down for a while, still being down at the time of writing. The lead author did not respond to an ask for updates via e-mail. (And efforts taken in order to host such an endpoint proved rather unsuccessful)

Schmelzeisen et al. published a paper at the ISWC conference outlining "Wikidated 1.0" - "a dataset of Wikidata's full revision history, which encodes changes between Wikidata revisions as sets of deletions and additions of RDF triples" [Schmelzeisen et al., 2021]. They also argue that this is the (at the time in 2021) first published large dataset of an evolving knowledge graph, which is argued to be a recently very popular research subject in the Semantic Web research space. Besides providing limited statistics, they also published their work in a GitHub repository, mentioning that it is open source and working, however, the GitHub repository shows no signs of this as it is labeled "currently refactoring" and there are no recent changes. Also, we were unable to find the created datasets. Therefore, we argue that the efforts taken are well worth the time of examination, but not worthwhile to proceed in this direction. The lead author is also non-responsive via e-mail.

Under the title "Representing Evolving Knowledge Graphs through Incremental Embeddings", a Master Thesis by Lafraie dealt with a completely different topic but developed a script that extracts edit triples from the Wikidata pages-meta-history dumps [Lafraie, 2020]. While not extracting all the information that we needed for our use case, the general approach and code structure were quite appealing. We have been inspired by this work for drafting the early components of the "./preprocessor" folder of our GitHub repository. This is to date the only resource that influenced the engineering work directly.

Analyses

Sarasua et al. looked at a similar dataset as we do, specifically going into quantitative analysis mainly focused on comparing editing behavior over time [Sarasua et al., 2019]. They look at the volume of edits, the lifespan of editors, at trends in editing behavior and moved on to predicting both of those variables. They conclude with other interesting findings, focusing on a Wikidata dump that covers 4 years of edits (end-date 01.07.2016). In light of (to the best of our knowledge) no more recent studies like this at the time, it might also be interesting to give an update to the descriptive statistics in there.

Piscopo et al. pursued multiple streams of research in Wikidatas contents, one of them being "What Makes a Good Collaborative Knowledge Graph: Group Composition and Quality in Wikidata" [Piscopo et al., 2017]. They looked at a sample of 5.000 Wikidata Items, going into quality, effects of tenure, and interest diversity, focusing partly on human users. They concluded that "the interaction between human and algorithmic users is necessary to create high quality items" and that "Contributions from anonymous users are instead detrimental for quality". Additionally, they found that tenure and interest diversity have a positive correlation with quality. A good mix of people from different backgrounds and experience in Wikidata-edits is found to be likely beneficial. In comparison, our approach focuses on using a near full-size sample.

Note that both Analysis Papers here have been used to detail and craft the approach in [Baroncini et al., 2022], which again is the main resource of inspiration for the use case presented.

2.3 Summary

In light of the complex details of the technologies, this chapter tries to recapture the most important and streamlined things necessary to understand the further elaborations.

Wikidata is a Knowledge Graph (KG), containing vast information on different things, illustrated by using so-called claims or statements. Each statement uses a property to reference either a value or an entity again. The *triples* or also *RDF-triples* are the core of the Resource Description Framework (RDF), which mainly fixes the Syntax of data interchange in order to standardize. SPARQL is a query language designed to query knw in RDF-structure. Wikidata has no fixed ontology, and is not applying ontology schemas like RDFS or OWL. The two presented edge cases *statement ranks* and *redirect items* add additional detail to the way the data is stored and there is a need to handle these directly from a data perspective when trying to analyse the statements. Previous Data Processing or Data Gathering research focused on obtaining the Wikidata edit history was found to be unfit for the use case, however, [Lafraie, 2020] and the according GitHub repository[12] was used as an inspiration for the early stages of parsing the data.

The next section will go into detail on the Data Processing steps taken and will also focus on explaining the requirements of data for the use case.

3 Data Processing

3.1 Requirements

Referencing subsection 1.1 Research Questions and Motivation, there is an inherent need for data. In exact terms, data that lists the edits that users have performed. This section will list the requirements that have been put up in order to get to the framework along two central questions: "Who should be included?" in terms of which users and "What should be included?" in terms of which edits are interesting. These two questions are very important for the analysis since we want to look at users that contribute from the mentioned perspective. Therefore, it is essential to know what and who is included, and vice-versa.

Who should be included?

The question on who should be included is a fairly undefined question. It relates to the fact, that almost anyone can contribute to Wikidata, in general, called users. Users can come in four flavours: registered (human) users, registered (bot) users, anonymous (human) users, and anonymous (bot) users.

[12]Found here: `https://github.com/rlafraie/WikidataEvolve`

It is important to mention, that bots in general, meaning programs automatically contributing to Wikidata, play a huge role in the way Wikidata is built. According to [Steiner, 2014], 88% of all edits they looked at stem from "robotic users" or bots. There are many use cases in which bots are actively contributing valuable information to Wikidata or maintaining existing information. Especially in the areas where one wants to import data from external sources, or one wants to enforce certain community guidelines, bots are a valuable part of the Wikidata economy.

Reflecting on the use case, it is relatively easy to rule out anonymous edits as a whole, because there is no way to identify the human or bot behind it and therefore it cannot add to the bigger picture.

For the registered bots, however, it is not that easy. They add valuable information to Wikidata or might enforce certain rules, possibly also switching "wrongful" properties for "right" ones. This, however, is based on predefined rules, not human behaviour. Therefore, we decided to also not look at bots, as they might add noise and hinder us from really understanding the human-made schema.

In order to be able to detect bot users, one can look into the Wikidata guidelines on bots[13]. In theory, all bot users should be flagged as bots by having a "bot" string in their username. However, in "Bot Detection in Wikidata Using Behavioral and Other Informal Cues", there is an evaluation about whether or not all bot users comply with the Wikidata bot rules and the accompanying registration and flagging. They found that at the time of the study in 2018, 3% of edits not flagged as bot edits were bot-made and therefore wrongly flagged when using an existing Wikidata account to make the edits, and 2% of anonymous contributions were bot-made. The accompanying published model scores well in identifying bot user edits and could be used to find the additional percentages of wrongfully flagged bot user edits. However, the paper also states that the possible skewing of the following analysis should be minimal to non-existent, depending on the use case [Hall et al., 2018]. Overall, the paper also states that current methods for bot-detection are either getting the Wikidata-approved table of all current and former bots actively contributing, or using the signaling phrase "bot" in the name, which all bots should, in theory, have.

That is the theory on bot detection as of now and because of the heavy data work involved, we decided to just go for the naive approach and identify bots by the signaling "bot" phrase in the username (for now).

[13]See: `https://www.wikidata.org/wiki/Wikidata:Bots`

What should be included?
The triple statements with subject, property, and object. Additionally, all information on the user (username) and the timestamp of the edit should suffice from a conceptual perspective, to be able to identify and use the information provided. All of this data can be found in the Wikidata edit history, as mentioned earlier.

However, there are some programmatical edge cases like the mentioned statement ranks or redirect items. The handling of these will be detailed when going deeper into the corresponding parts of the framework.

In general, a user can either edit, add or delete statements. Since we are interested in the schema and an edit does in general not edit the property and therefore the overall schema, we decided to neglect the changes. Additions are the main part of interest since these establish a property and therefore a vocabulary added to the schema. Deletions are important, but difficult to handle in terms of their meaning, as one statement might be deleted and another one added as a substitute. Therefore, these are also not important for the use case. In sum, the established requirements can be brought down to the following list:

- All edits performed that added a statement (performed by a human, registered user)
- Including information on the user, the statement added, and (for future purposes) the timestamp

The corresponding data file minus the timestamp has been coined as `store_UVEC.csv`, where UVEC stands for: **U**ser **V**ocabulary **E**ntity and **C**ount. Vocabulary references to the definition proposed in the introduction, where the vocabulary is aggregated by the property and potentially also the object (see Figure 3. The entity in this case references to the subject, where the statement was added. `store_UVEC.csv` is the final delivery object of the Data Processing framework. It is then transformed into more streamlined files for the planned analysis subtasks. section 4 analyses the proposed file `store_UVEC.csv` in basic terms.

3.2 Implementation aspects and computation resources

The Wikidata Statistics page[14] lists that Wikidata holds 102.886.792 items, which have been created and updated by 1.886.991.152 edits (not mentioning

[14]Checked on 02.05.2023, https://www.wikidata.org/wiki/Wikidata:Statistics

whether this is triple-related or general edit session related). The sheer volume makes Data Processing a crucial part of applied computations of scale, leading to increased complexity. Instead of, like some research on Wikidata does, limiting the time-frame of Wikidata to study or just drawing a random sample, this thesis handles the full time-frame and contents because providing such elaborate datasets is crucial for future research, which might then again limit the time-frame if needed.

In exact terms, the Wikidata dump of choice is the dump storing the information up until the 1st of January 2023. It was obtained via the Wikidata dump page https://dumps.wikimedia.org/wikidatawiki/. The dump is now not available anymore, but the same files are available in a more recent dump. Specifically, all the files have a file name like the following: "wikidatawiki-20230401-pages-meta-history1.xml-p1p154.bz2". They are .bz2 compressed, which we chose instead of the standard .7z compression format, because of better performance. The full dump has a size of approximately 1.6 terabytes and took several days to download. The storage in use is an SSD because of better access times.

Multiple scripts were developed to handle this task, dividing one big step into small ones to reduce complexity and ensure adaptability. The computations were run on a virtual machine running a Ubuntu Server version, featuring 48 CPU cores and 128GB of RAM, and a 3TB SSD. The main script language used is Python, with the main handler in bash script. Using several methods and packages in order to leverage the 48 CPUs in a multiprocessing setting was key since Python is natively developed as a single-core script language. This leads to an additional layer of complexity regarding RAM management and creates the threat of out-of-memory errors. We tried to minimize this risk by naively applying batch-handling settings and writing to buffer files in order to free RAM in most cases. The resulting scripts are for sure not optimal in terms of performance, resource utilization, and compatibility with smaller machines having fewer resources. However, they are efficiently developed in order to create the data that is needed for this study. For production use, refactoring the scripts and even more optimizations in order to avoid out-of-memory errors would have to be thought about. Therefore, we released the scripts in an open-source license setting, maybe leading to open-source-style improvements in the future. We invite everyone to contribute or build better versions based on this. All the scripts are to be found on Github under https://github.com/N-Krenn/WIKIEVO. The corresponding data files, which eliminate the need to compute, are to be found under https://github.com/N-Krenn/WIKIEVO.

3.3 The Framework

Before going into the details of the parsing, filtering, and structuring of the data, the general framework is explained.

The scripts are put into three self-contained parts, namely being:

1. preprocessor
2. extractor
3. network preparation

Note that the parts require the input from the previous part in order to work in the way that they are intended to work. The base input starting from the top is the mentioned "pages-meta-history" dump (it does not matter if complete or not).

The following paragraphs will go into some detail on the functions and main deliverables each of the parts bring, without going into too much implementation details.

preprocessor The preprocessor takes the dump as input and transforms its contents into simple comma-separated-values (CSV) and stores this into files per entity (Q-entity). In essence, it parses the XML of the dump, taking the parts we want to see.

extractor The result of the preprocessor is pretty big and contains unneeded information. To streamline it for the use case, the extractor accesses the output of the preprocessor and transforms the information further, leaving out parts not necessary. The result is one file `store.csv`, which contains all the relevant edits row by row, again in CSV-format.

network preparation The folder `network_prep` contains all the files for transforming the result of the extractor even further, mainly aggregating it in the way that the two created networks need it. This is the most ad-hoc part of the scripts, as the analysis is rather preliminary and there is little sense in re-running the same thing over and over. The results are multiple aggregated datasets, `store_UVEC.csv` for example, as well as the two networks mentioned in the introduction.

Each of the parts reduces the size of the data significantly, all the parts of the data are published in an online storage. The data of the in-between-steps and the customization options of the scripts allow for the opportunity to conduct modified use cases using our scripts.

Framework Diagram The overall framework diagram is depicted in Figure 12. The details are to this moment not fully explained but will be in the following section. The main point here is to illustrate again, that all of the three main components contribute to a streamlined approach to get the data in the fit form.

3.4 Implementation details and workflow

This section is entirely focused on the details of the workflow that happens in order to get to the datasets used for the analysis. It can be viewed as a report of implementation. It is structured by the main compartments, also quoting the most important code of each of the steps. The last step is then to explain the Main Handler in the sense that one reading it is able to make the foreseen customizations and run the scripts.

preprocessor

As mentioned, the preprocessor takes the full dump as input, which consists of multiple .bz2 compressed XML files. In essence, this part is responsible for parsing it and getting all the information.

The base file structure of an XML file is depicted in the code snippet below. It focuses on the entity Q15, which is the continent of Africa. We will not go into detail about each of the XML-tags, but the main information is stored in the $< text >$ tag. Especially the *claims* are interesting, the rest of the needed information is contained in either the subject header (and not the revision itself). The example shows two revisions, that could each have totally different outcomes, depending on what the user changed.

```
<page>
  <title>Q15</title>
  <ns>0</ns>
  <id>111</id>
  <revision>
    <id>16</id>
    <timestamp>2012-10-29T17:03:21Z</timestamp>
```

```
      <contributor>
        <username>Denny</username>
        <id>713</id>
      </contributor>
      <comment>Created page with &quot;A continent&quot;</comment>
      <model>wikibase-item</model>
      <format>application/json</format>
      <text bytes="160" xml:space="preserve"> ... TEXT OMITTED ...
          claims&quot;:[] </text>
          <sha1>hmfipnvrbxi8qvtd15gu35iqprqbdnb</sha1>
    </revision>
    <revision>
      <id>20</id>
      <parentid>16</parentid>
      <timestamp>2012-10-29T17:10:49Z</timestamp>
      <contributor>
        <username>Denny</username>
        <id>713</id>
      </contributor>
      <comment>/* wbsetlabel-set:1|de */ Afrika</comment>
      <model>wikibase-item</model>
      <format>application/json</format>
      <text bytes="182" xml:space="preserve"> ... TEXT OMITTED ...
          claims&quot;:[] </text>
      <sha1>a0f3zqelhf49i98e46wgzwoine3x7jp</sha1>
    </revision>
```

The preprocessor with its main-file `preprocessor.py` utilizes the cores that were entered as a parameter to process one XML file per core. It decompresses it and works through the content of the XML file line by line. The following code snippet shows how this is achieved as an example for the timestamp and the username, which are both contained in the $< revision >$ tag:

```
    elif line.startswith("    <timestamp>"):
        timestamp = line[len("
            <timestamp>"):-len("<\timestamp>\n")]
    elif line.startswith("      <username>"):
        username = line[len("
            <username>"):-len(r"<\username>\n")]
```

For each revision, the then collected information is transformed by helper functions stored in `revision_handling.py`. The “truthy claims list” is computed, which in essence is nothing else than all the claims currently made in the revision under study. The name itself hints to the fact, that only “truthy claims” are taken into account, not the already mentioned *deprecated statements*. This is the point, where only *preferred statements* and *normal statements* are seen as statements listed, and the *deprecated statements* are seen as deleted and not valid anymore.

The two sets of truthy claims, one containing all the previous revisions claims and then the current revisions claims are compared. This leads to a list of insert operations (new_set - old_set) and a list of delete operations (old_set - new_set). The beauty of this is, that changes are viewed as a deletion and an insertion. If comparing the sets of claims from the previous to the actual revision, the statement that has been changed is not there anymore. Therefore it is viewed as a deletion. However, when comparing it the other way around, there now is a new statement. In this way, changes are depicted as both, a deletion and an insertion. This simplification is highly effective, since computing the changes as real changes would lead to another comparison and more complexity of computation.

Note that somebody wanting to adapt our code might as well just take the output of the preprocessor and then find the “changes” there, since there should be a deletion first and then an addition of the same property with different object values. However, this is only possible for “normal” changes, where the object value is modified. In general, through the web interface, users may only edit the object value or add additional information to it, but not change the property value of some already existing statement directly as of now. However, changes made by first deleting the statement and then adding a new one with a different property are inherently harder to find and classify.

The then gathered information is stored in a CSV-formatted file, where one line represents one addition/deletion. As an example, you can see an excerpt of the resulting file for Q9316 below. It only shows the “earliest” 10 edits.

```
timestamp, type (addition/deletion), subject-ID, property-ID,
    object (either ID or value), username, user-ID
2013-04-01T09:24:47Z,+,9316,373,"Sikhism",BotMultichill,123349
```

```
2013-05-16T21:05:37Z,+,9316,508,"36397",SamoaBot,116859
2013-07-01T01:44:20Z,+,9316,279,9174,Emw,5584
2013-10-05T16:34:55Z,+,9316,910,8744743,Esquilo,70719
2013-10-16T06:55:40Z,+,9316,349,"00570974",F705i,113398
2013-12-05T23:43:20Z,+,9316,646,"/m/06yyp",Legobot,18825
2014-12-06T22:54:00Z,+,9316,31,9174,Yamaha5,62081
2015-01-20T13:03:34Z,+,9316,1151,11510351,AdamBMorgan,66207
2016-01-09T07:14:13Z,+,9316,935,"Sikhism",Jeblad (bot),2155874
2016-01-10T09:23:55Z,-,9316,935,"Sikhism",KrBot,150965
```

Entity Q9316 is relating to "Sikhism", a religion. The file is named Q9316.txt and is stored in the directory named like the dump file it was taken out of. In this step, the *redirect items* are computed, but have a prefixed filename "redir_". In the next step with the extractor, this prefix of the filename is used to identify and omit the *redirect items*.

runtime of preprocessor The preprocessor in general is the most resource-heavy part of the scripts we provide. In exact terms, we only let it run two times since the computing times are well above 7 days on our machine. The parsing scales on both, hard-drive speed and the number of CPU-cores, but the CPU-cores are the most determining factor.

Please note that the preprocessor is in essence related to the scripts provided under[15] *by Rashid Lafraie. We adapted the scripts heavily to our use case, but were inspired by the structure and some code parts.*

extractor

The output of the preprocessor holds many files, stored in folders directly related to the dump file they have been taken out of. So the extractor screens through all of those files and again filters and transforms them into the needed structure. This is the essential step of eliminating unneeded data, as deletions are discarded and timestamps are also not looked at. It does that in a similar way to the preprocessor, also utilizing multiple cores. The resulting file is called `store.csv` and is just one huge file for all the edits.

The extractor features two main parameters, that one can edit:

[15] https://github.com/rlafraie/WikidataEvolve

1. **cut entity:** To reduce the overall volume of output data and processing time, the parameter cut entity allows choosing a Wikidata entity (starting with Q). The chosen entity's first claim creation timestamp is the cut-off timestamp. This means, that all following entities will not be looked at, and the preceding entities' edits will only be looked at until the timestamp. In concrete terms: When choosing Q500 as the cutoff entity, only entities Q1 to Q500's first claim edit will be chosen, whereas Q1's edits will only be selected if they were made before the timestamp. This gives the user the great opportunity to limit the time-frame and first test the extractor, without having to accept huge runtimes.

2. **list of common properties:** We already motivated the fact that for some of the most used properties, it is a good idea to also include the object in the vocabulary definition. This is because of various reasons and adds context to the property, which otherwise might be quite meaningless. With this parameter, we give the opportunity to customize the selection. In the thesis example, this list only features P31 (instance of).

After setting these two parameters, the extractor starts its operation according to the parameters. The result of the extractor looks like the following:

```
index,username,userID,vocabulary,entity
0,Whym,3820,P107,8
1,Emw,5584,P279,8
2,Incola,89578,P279,8
3,BotMultichill,123349,P373,8
4,Mattflaschen,6856,P31:331769,8
5,Mattflaschen,6856,P31:9415,8
6,SamoaBot,116859,P508,8
7,Shisma,5295,P18,8
8,BotMultichill,123349,P910,8
```

The object and the timestamp are not in the dataset anymore. This leads to a great reduction of data size, as `store.csv` now only has 23GB. This size of data is pretty easy to handle by different means, but not necessarily trivial.

runtime of extractor In the final example, the extractor took around 16 hours to perform its operations. This is pretty fast, especially when looking at the runtime of the preprocessor.

network preparation

As already mentioned, the tools used to get through the analysis consist of separated scripts, each doing some in-between operation or the final step of constructing the network analysed. We will not go into great detail here, as the scripts should be pretty self-explanatory and the analysis part will explain in detail what each of the algorithms looks at.

However, we already know that `store.csv` needs to be transformed in different directions in order to look at the use case. `store_UVEC.csv`, which we already mentioned, is nothing but an aggregation of `store_voc.csv`, again reducing its size. The aggregation is performed by leveraging in-memory data wrangling library pandas[16] in combination with the package modin[17], which scales pandas utilities to multiple cores.

From this, aggregations for making the networks edge-lists are performed, however, this is explained in section 5.

Main Handler

The main handler is named `main.sh` and is a bash script starting each of the scripts with the correct parameters. By default, it uses the standard parameters used to recreate this study. However, someone trying to recreate a similar use case might want to adapt them. They are extensively explained in the script itself under `https://github.com/N-Krenn/WIKIEVO/blob/main/main.sh`.

Final Overview of the Framework Figure 12 lists all the processing steps and additional augmentations on what is cut at which point. It also shows the further steps for the Network Analysis, which are not explained yet but will be explained in section 5.

[16] `https://pandas.pydata.org/`

[17] `https://modin.readthedocs.io/en/stable/index.html`

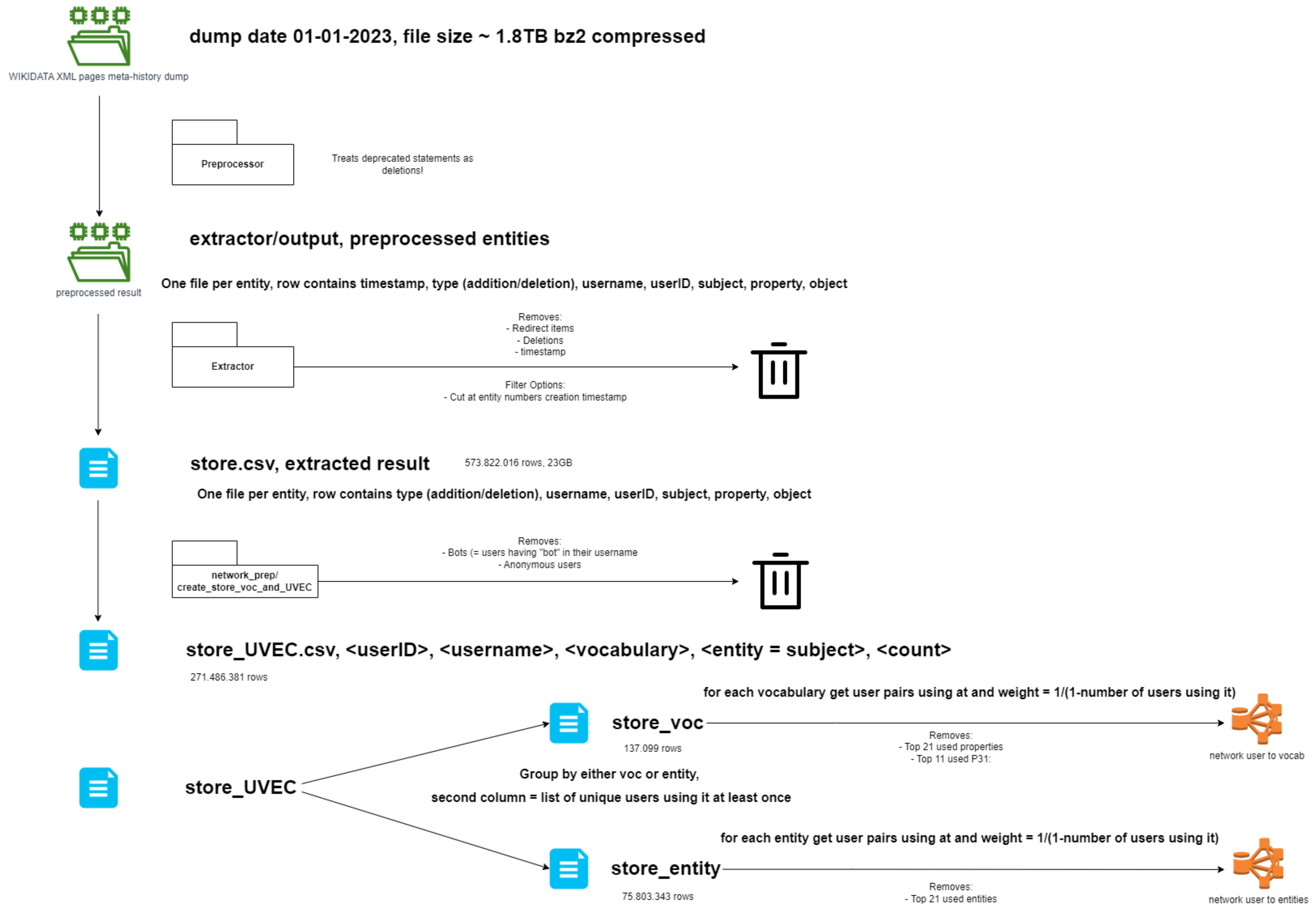

WIKIDATA XML pages meta-history dump
dump date 01-01-2023, file size ~ 1.8TB bz2 compressed
Preprocessor
Treats deprecated statements as deletions!
preprocessed result
extractor/output, preprocessed entities
One file per entity, row contains timestamp, type (addition/deletion), username, userID, subject, property, object
Extractor
Removes:
- Redirect items
- Deletions
- timestamp
Filter Options:
- Cut at entity numbers creation timestamp
store.csv, extracted result
573.822.016 rows, 23GB
One file per entity, row contains type (addition/deletion), username, userID, subject, property, object
network_prep/
create_store_voc_and_UVEC
Removes:
- Bots (= users having "bot" in their username
- Anonymous users
store_UVEC.csv, <userID>, <username>, <vocabulary>, <entity = subject>, <count>
271.486.381 rows
store_UVEC
Group by either voc or entity,
second column = list of unique users using it at least once
store_voc
137.099 rows
for each vocabulary get user pairs using at and weight = 1/(1-number of users using it)
Removes:
- Top 21 used properties
- Top 11 used P31:
network user to vocab
store_entity
75.803.343 rows
for each entity get user pairs using at and weight = 1/(1-number of users using it)
Removes:
- Top 21 used entities
network user to entities

Figure 12: Data Processing Framework Overview

4 Analysis of the processed dataset

As already mentioned, the dataset `store_UVEC.csv` consists of U - username/userID, V - vocabulary, E - entity (subject), and C - count. The originating file `store.csv` as the result of the extractor features 573.822.016 rows (=edits). One interesting thing to point out here is that Wikidata states that around 1.887.755.724 edits have been made in total[18]. `store.csv` contains all additions (including the "changes" where the change is counted as an addition). Therefore, summed up in one naive statement, probably one-third of Wikidatas edits are a) additions and b) not performed on redirect items. Please note that we did not check this statement, it is merely an observation - many exclusions apply.

After eliminating the bots by deleting all edits where the username features a non-case-sensitive "bot", there are 306.316.135 rows left.

Note that the bot removal does not remove whole user-IDs, but single edits. Therefore, if a user renames himself to something containing "bot", the consecutively made edits with this new username will be removed, but the edits before that stay.

Then, after eliminating anonymous users, 302.986.544 edits stay. After aggregating by all characteristics and summing for the count variable, the set has a total length of 271.486.381 rows.

So starting from the extractor result `store.csv`, around 300 million rows are deleted, with the biggest portion removed being the bot edits. From manual inspection and a few clues, there are still some wrongly flagged bots in the dataset, but we ignore them at this point.

Table 1 provides an overview of row counts for each step, the asterisk illustrates that also some header rows of the aggregated datasets were removed in this step.

In Table 2 some basic information on the columns is displayed. Interesting here is that there seem to be few statements that have a very high aggregation in "count", therefore being executed very often, possibly with multiple values for the object.

[18]Checked on 03.05.2023, `https://www.wikidata.org/wiki/Wikidata:Statistics`

	number of rows	**difference**	**difference to store.csv in %**
store.csv	**573 822 016**		
removing bots	306 316 135	- 267 505 881	53%
removing anonymous users*	302 986 544	- 3 329 591	53%
grouping	271 486 381	- 31 500 163	47%
`store_UVEC.csv`	**271 486 381**		**47%**

Table 1: Overview on row counts of data processing

	userID	**entity**	**count**	**username**	**vocabulary**
count	271 486 381	271 486 381	271 486 381	271 486 381	271 486 381
mean	946 925,31	43 034 233,95	1,12		
std	1 307 120,59	35 970 030,99	2,22		
min	1	1	1		
25%	44 949	10 620 042	1		
50%	160 623	33 111 439	1		
75%	1 895 734	72 215 150	1		
max	6 041 217	115 955 563	5 601		
unique values	142 124	75 803 343		142 112	137 099

Table 2: Overview on *store_UVEC.csv*

user-centric evaluation
Since the use case focuses also on the users, the following quickly provides an overview of the user base in the dataset, mainly focusing on a) the individual users and b) the individual edit count. As mentioned in section 9, some papers have looked at this in more detail and from different angles.

The dataset contains 142.124 unique userIDs. Note that the username may be changed because of various reasons and is not unique, but the userID is. It is the exact userID also used in Wikidata, which may also be queried through the APIs provided.

By performing a grouping on the userID, one thing that previous research has already uncovered comes to light: few users are responsible for the most edits. Figure 13 shows this phenomenon, visualizing the distribution of user edits by single userID. The figure features logarithmic scaling in order to make the data more visualizable. It is safe to say that by manual inspection and the extraordinarily high edit counts of some users, there are still some bot users in this list, which have not labeled their username with "bot" as they should according to the community guidelines[19]. This also relates to what one paper regarding bot detection in Wikidata states [Hall et al., 2018].

vocabulary-centric evaluation
Since the vocabulary is also one distinct piece of information that is of huge value, we will quickly look into this as well. Figure 14 shows the distribution of vocabulary usage in general.

Comparing this to the histogram of the user edits, it paints a similar picture. Few vocabularies are used many times and it declines. Note that the definition for vocabulary here again is that it just lists the property used, except for property P31 (instance of), where it annotates the linked object as well.

Table 3 lists the top 50 properties used (ceteris paribus). Annotated in the table is also the summed count for all P31 properties used, which makes it the most used vocabulary in this table. There are also three further P31 properties including the object in the table, therefore individually being part of the top 50 properties used.

This table is novel, as it only features additions and registered edits, while not having a "bot" flag in the username, also excluding redirect items. Wikidata publishes something similar periodically, although focusing on the "quantity of item pages that link to them". Table 4 shows the top 30 of the

[19]Retrieved 04.05.2023 from `https://www.wikidata.org/wiki/Wikidata:Bots/`

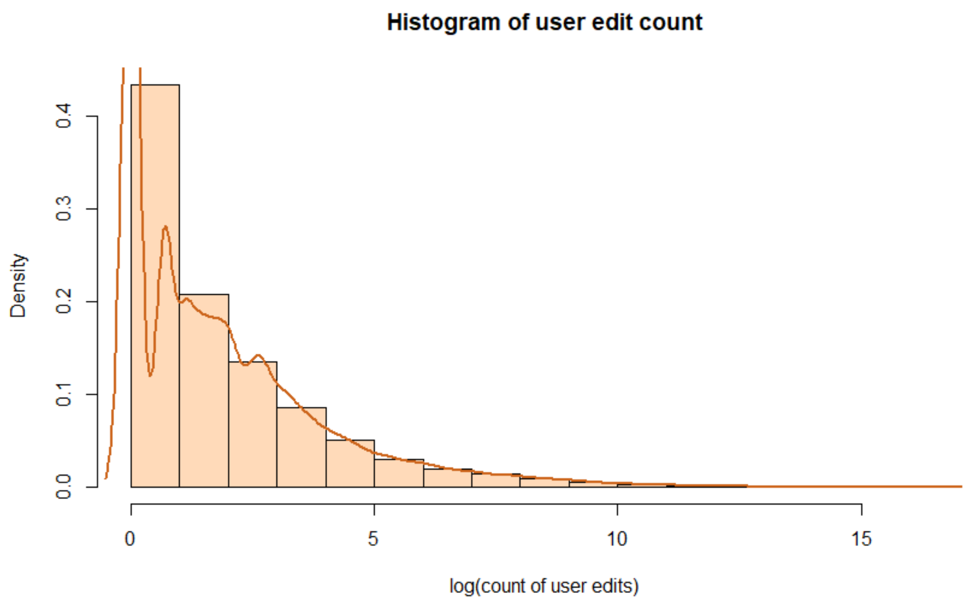


Figure 13: Logarithmic user edit count histogram

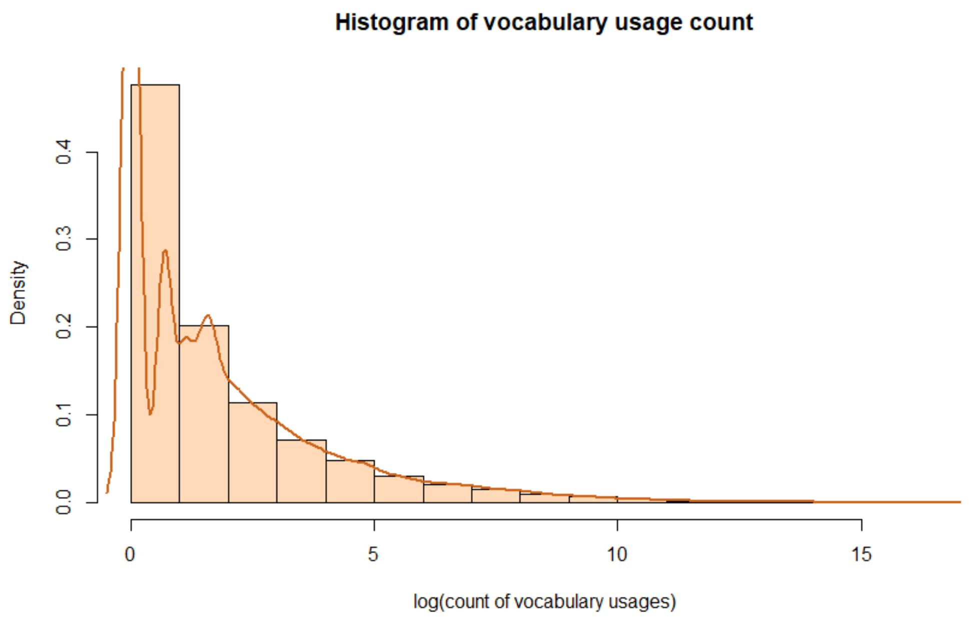


Figure 14: Logarithmic vocabulary usage count histogram

vocabulary	**count**	...		
P921	15.231.938		P31:7318358	2.120.169
P407	11.505.119		P570	2.047.564
P106	7.961.543		P421	1.896.053
P50	7.781.531		P7859	1.878.919
P2860	6.880.546		P214	1.864.096
P17	6.842.307		P31:4167836	1.851.811
P735	6.378.169		P1433	1.795.190
P131	6.078.806		P361	1.749.758
P21	4.940.412		P571	1.704.255
P31:5	4.083.002		P971	1.658.123
P18	4.077.653		P108	1.544.396
P569	4.044.264		P40	1.531.649
P734	3.499.680		P136	1.507.306
P27	3.453.092		P6179	1.473.000
P2093	3.341.690		P856	1.396.745
P646	3.120.430		P166	1.345.933
P625	2.795.039		P641	1.271.282
P2888	2.765.384		P495	1.173.497
P577	2.527.764		P1566	1.152.317
P1476	2.396.107		P356	1.140.560
P373	2.326.930		P3083	1.116.976
P19	2.315.381		P39	1.107.901
P10585	2.300.615		P364	1.106.015
P528	2.256.543			
P2671	2.220.806		**TOTAL**	**160.860.516**
P69	2.161.122			
P1412	2.141.128		P31	28.418.975

Table 3: Top 50 most used vocabularies

property	**Quantity of items listing the property**	. . .		
cites work (P2860)	290.079.030		astronomical filter (P1227)	33.144.982
series ordinal (P1545)	170.920.827		DOI (P356)	31.290.077
author name string (P2093)	136.714.053		author (P50)	29.924.037
instance of (P31)	112.249.115		catalog code (P528)	28.862.240
stated in (P248)	93.476.057		main subject (P921)	27.137.210
retrieved (P813)	93.038.309		catalog (P972)	23.942.753
reference URL (P854)	71.157.358		language of work or name (P407)	22.317.205
PubMed ID (P698)	63.859.886		object named as (P1932)	18.832.977
title (P1476)	48.565.223		country (P17)	16.661.583
publication date (P577)	48.180.275		located in the administrative territorial entity (P131)	12.840.015
published in (P1433)	41.007.111		point in time (P585)	12.834.801
page(s) (P304)	37.783.708		PMCID (P932)	11.699.344
volume (P478)	37.083.220		occupation (P106)	10.808.364
issue (P433)	34.569.417		start time (P580)	10.669.635
apparent magnitude (P1215)	33.145.057		coordinate location (P625)	10.633.150

Table 4: Wikidata's top 30 properties (Retrieved from https://www.wikidata.org/wiki/Wikidata:Database_reports/List_of_properties/Top100 on the 7th of July 2023

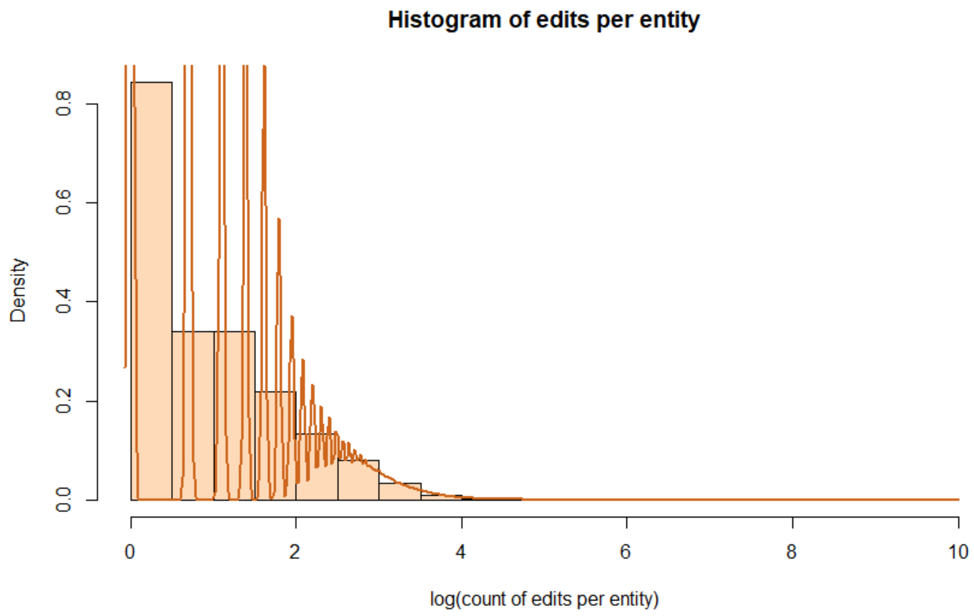


Figure 15: Logarithmic edits per entity histogram

total list containing the top 100 items. The differences compared to Table 3 are very visible, although some properties change in importance. One of the changes visible is that certain properties used to link information from one source to another, for example the property "cites work" (P2860) is not the top property anymore. Moreover, "retrieved" (P813) is completely missing in our list. From our current standpoint, we think that these might not be as important anymore, because they might be heavily used by bots.

entity-centric evaluation

The entities in the set are a distinct piece of information as well, just like the vocabulary. Figure 15 shows the same distribution graph of the entity usage in general.

Comparing this to both previous histograms, it also paints a similar picture. Few entities are used many times and it declines. The entity in this case is only the subject, so the edited item, not the object.

Table 5 lists the top 50 entities used (ceteris paribus), just like in the vocabulary case.

The dataset `store_UVEC.csv` is available for download in a compressed format under the link provided under `https://github.com/N-Krenn/WIKIEVO`.

entity	**count**	...		
Q4115189	20.852		Q17339402	2.058
Q21558717	6.368		Q87119811	2.027
Q861252	5.700		Q15397819	1.818
Q21505108	3.911		Q183	1.787
Q22676705	3.888		Q30	1.682
Q21521425	3.786		Q30279457	1.675
Q86913546	3.764		Q819425	1.630
Q21521423	3.742		Q623	1.615
Q87250860	3.728		Q21996341	1.539
Q13646	3.727		Q408	1.518
Q21481859	3.610		Q142	1.460
Q21521427	3.480		Q56754739	1.443
Q21559757	3.406		Q16943273	1.439
Q21481867	3.383		Q252	1.438
Q48653337	3.315		Q51786082	1.415
Q27349821	3.267		Q56895655	1.411
Q27335792	3.234		Q13406268	1.405
Q29037899	3.176		Q691	1.381
Q27336022	3.043		Q48589857	1.380
Q27336098	2.973		Q27347646	1.356
Q27335409	2.574		Q16553	1.353
Q27336831	2.486		Q87477462	1.328
Q21994528	2.436		Q42119679	1.322
Q27345502	2.389		Q148	1.278
Q42126267	2.317			
Q84055514	2.126		**TOTAL**	**143.439**

Table 5: Top 50 most used entities

5 Network Analysis

The following chapter as the main section of the thesis will outline the efforts taken in order to answer the research question.

First we will explain some preliminaries around Network Analysis, before moving on to explaining the approach more concisely and technically.

Then, in the subsections on Network 1 and Network 2, the detailed networks are analysed and described, along with each of the conclusions on the networks.

Finally, we interpret the results of both networks and begin to outline and motivate future work ideas.

5.1 Preliminaries

A network can be described as a set of nodes and edges, much similar to the described knowledge graph. In fact, both of those terms are related to Graph Theory. Graph Theory is a mathematical field of study, where graphs are mathematical structures used to relate pairwise elements in a modeling way. In fact, Figure 1 used to show a basic example of a knowledge graph is also a graph.

In exact terms, one of the more mathematical and canonical definitions by [West et al., 2001] is:

"A **graph** G is a triple consisting of a **vertex set** $V(G)$, an **edge set** $E(G)$, and a relation that associates with each edge two vertices (not necessarily distinct) called its **endpoints**."

Thus, a network in essence can be modeled as a graph. This is important since Graph Theory methods are used to create another more specific field of study, called Network Analysis. It focuses mainly on the complex connectedness of things, for example, the modern society and people's connections. Numerous studies focused on different aspects of this connectedness were conducted already, leading to new insights on for example how people in social networks are connected (e.g. [Catanese et al., 2011][Boy and Uitermark, 2016]).

Overall, the idea of network science is to model very complicated things as simply as possible. The results mostly turn out to be somewhat complicated, but at least tractable in some sense. One famous early example is the paper by [Watts and Strogatz, 1998] on "Collective dynamics of small-world networks". They put Network Science somewhere between either a completely random or completely regular topology, using algorithms and techniques to simplify them and find the “gist” of the networks. In essence, this is also what we try to do with the Wikidata edit history. The way users contribute to Wikidata is assumed to be very complex, but with enough simplification, we might be able to find some core insights.

However, the area around network analysis and knowledge graphs such as Wikidata is pretty novel in research (to the best of our knowledge). This might have different reasons, one of which is that the research community focusing on knowledge graphs might not like the idea of omitting many aspects and data points in order to get to some simplified conclusion. In general, knowledge graphs and in particular Wikidata are way too big to really be able to look into the way users contribute in detail, without this simplification. Other means include creating metrics and standardized measurements and comparing them in case studies of particular smaller excerpts of data, which is what has been done surrounding Wikidata a lot. However, we think that simplifying the whole Wikidata dump's data to a point where networks and clustering algorithms can help to get insights is a particularly interesting methodology and holds massive potential, coming from different perspectives.

What we add on top of the network approach is the community analysis approach. Community analysis has been a term in network analysis for quite some time now, as early adopters like [Arenas et al., 2004] or more mature projects like [Ferrara, 2012] illustrate. The basis of this is that the nodes in networks can be clustered into so-called communities, where one community consists of a set of users (or nodes in the network sense). There are different methods for different use cases, each one being determined in its effectiveness by the clustered result (as different use cases may want to achieve different things).

The infomap community clustering approach is one very famous example of such a community detection algorithm, aiming to cluster the nodes into different communities. It was developed by Martin Rosvall and Carl T. Bergstrom and is based on probability flows of random walks on a network. For more detailed information on the famous and well-accepted algorithm please consult [Rosvall and Bergstrom, 2008][Rosvall et al., 2009]. This is the main algorithm of choice used for the community detection aspect. The algorithm bases its decisions to label one node as part of one community based on the overall connections to other nodes, as well as an optional weight for each of the connections.

In sum, network analysis and its accompanying community analysis approaches are used to elicit and analyse the overall connectedness of nodes (in our use case: users). The methods used may be of different natures, all having a strong quantitative or heuristic focus.

In the next subsection, we detail the approach in general terms before going into the network creation specifics and analysis parts, revisiting the goals and the research questions.

5.2 Detailed Approach

Note that the starting dataset the analysis is based on is `store_UVEC.csv`. In order to recall the goals and streamline the analysis approach, this subsection gives additional information.

Looking at `store_UVEC.csv`, the data that is contained focuses on:

1. the user
2. the vocabulary
3. the entity that was edited (or: subject)

Naively thinking about the types of networks one could create, two of the elements could be used, one as nodes and one as edges. When putting the similarity aspect in focus, one could think of six networks that are more obvious to spot:

1. nodes: users, edges and their weights: similarity of entities (or: similarity of domain in the bigger picture)
2. nodes: users, edges and their weights: similarity of vocabulary used (or: (sub-)schema similarity in the bigger picture)
3. nodes: entities, edges and their weights: similarity of vocabulary used (or: (sub-)schema similarity in the bigger picture)
4. nodes: entities, edges and their weights: similarity of users (or: similarity of domain experts)
5. nodes: vocabularies, edges and their weights: similarity of users that use it
6. nodes: vocabularies, edges and their weights: similarity of entities it appears in

When looking at the list, we agree that some networks seem quite meta and still need interpretation work. However, especially the first two examples hold, in our opinion, large potential:

- The first would introduce a novel kind of community classification by domain.
- The second would introduce a novel kind of community classification by vocabulary.

These are exactly the two networks matching our originally posed research questions. The other networks might hold a value of their own, potentially increasing their value in combination with the other networks.

The following paragraph will quickly examine the approach of the now named *Network 1* that maps the vocabulary similarity of users. Again, the users here would be considered the nodes and the edges depict the vocabulary similarity.

Defining the vocabulary similarity seems pretty abstract, but in exact terms, one might naively use the number of times two users x and y contributed the same vocabulary as seen in the following extensive example:

Let there be two users, David (userID 1234) and Clemens (userID 1235). David is the user depicted in Figure 3 (revisit this to review the statement to vocabulary definition again).

David (userID 1234) made the following statements:

1. Earth (Q2) is instance of (P31) terrestrial planet (Q128207)
2. Earth (Q2) has a mass of (P2067) 5.972.37 ± 0.01 yottagram
3. Earth (Q2) is named after (P138) soil (Q36133)
4. Life (Q3) is studied by (P2579) life sciences (Q864928)

These edits show up in `store_UVEC.csv` as the following lines:

```
index,username,userID,vocabulary,entity
1, David, 1234, P31:128207, 2
2, David, 1234, P2067, 2
3, David, 1234, P138, 2
4, David, 1234, P2579, 3
```

Note that the appearance of these statements in `store_UVEC.csv` *implies that the subjects Q2 and Q3 are no redirect items and that the edits were additions (or changes, but only the addition part of the change is displayed). The statement rank is not important, since we only look at the additions in the file.*

Clemens (userID 1235) made the following hypothetical statements:

1. Mars (Q111) is instance of (P31) planet (Q634)
2. Mars (Q111) has a mass of (P2067) 641.71 ± 0.01 yottagram
3. Mars (Q111) has use (P366) colonization of Mars (Q838950)

These edits show up in `store_UVEC.csv` as the following lines:

```
index,username,userID,vocabulary,entity
5, Clemens, 1235, P31:634, 111
6, Clemens, 1235, P2067, 111
7, Clemens, 1235, P366, 111
```

The naively defined similarity of their vocabulary might be defined as the sum of overlapping vocabularies, which is 1 (Index Nr. 2 and 6 feature the same vocabulary).

Index numbers 1 and 5 have the same property, but not the same object, as P31 is annotated with the object for additional meaning. In sum, the count of overlapping vocabularies between David and Clemens is 1 and we naively defined this as the weight of their vocabulary similarity. If we would have multiple users editing different things, this weight would show the strength of the two users' vocabulary similarity as indicated.

However, this naive weighting is not optimal in terms of really determining the strength of the similarity of the vocabulary of the two users. For example, it does not account for multiple facts, like if the vocabulary used is frequently used or just a few people use it for special occasions. It also does not account for the fact that users with a generally high edit count might have stronger connections to each other, even though this is not really the case.

For this exact reason, there is a need for a weighting algorithm, that depicts the vocabulary similarity, taking as much as possible additional information into account. For our use case, the hyperbolic weighting approach seemed ideal as a starting point.

Before explaining the hyperbolic weighting approach, the definition of *edge-lists* is in order. The basic example from above with users David and Clemens is used to illustrate this even further, adding another person called Theresa (userID 1236) with the edits:

```
index,username,userID,vocabulary,entity
8, Theresa, 1236, P366, 111
9, Theresa, 1236, P2067, 193
```

So, in total, we have a list of the following edits in the example of `store_UVEC.csv`:

```
index,username,userID,vocabulary,entity
1, David, 1234, P31:128207, 2
2, David, 1234, P2067, 2
3, David, 1234, P138, 2
4, David, 1234, P2579, 3
5, Clemens, 1235, P31:634, 111
6, Clemens, 1235, P2067, 111
7, Clemens, 1235, P366, 111
8, Theresa, 1236, P366, 111
9, Theresa, 1236, P2067, 193
```

The connecting vocabularies are:

- P2067: used by David, Clemens, and Theresa
- P366: used by Clemens and Theresa

The rest of the vocabularies used is distinctly used, these are just used by one user, not multiple. So if crafting a network for the purpose of the finding of community structures of similar vocabulary, the three nodes would be David, Clemens, and Theresa. The existence of an edge would require at least one instance of the same vocabulary used. Looking at the list, there would be the edges:

- David to Clemens (one occurrence of the same vocabulary used)
- David to Theresa (one occurrence of the same vocabulary used)
- Clemens to Theresa (two occurrences of the same vocabulary used)

These edges would need to have an assigned weight, which is determined by the weighting algorithm. The basis of the way we weigh the similarity is by the usage of the specific vocabulary. Therefore, when looking at the way how to produce this list of edges computationally, the most straightforward option is to take one vocabulary and compute all the edges, then assign a weight and sum with the rest of the individual vocabulary weights.

In pseudo instructions, this would look like the following:

```
for vocabulary in store_UVEC.csv:
    Find all combinations of two users --> edges
    Apply computed weight for vocabulary to each edge in edges
    Append to edge_list for all vocabularies, if the edge
        already exists sum the weights
```

Computationally, the first part of finding all combinations of users for one vocabulary is the most resource-heavy part. The mathematic binomial coefficient can help here. The basic mathematic principle, where C(n,k) is the number of possible combinations if drawing k items (in this case k=2) out of n items, is:

$$C(n,k) = \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

With this formula, one can determine the number of edges that a vocabulary used by n users would produce. This number can get pretty high. For example, the top 21 vocabularies of network 1 would lead to 5.749.636.702 edges. Each of the networks will have a section explaining the filters in order to reduce this heavy computation to a viable runtime (as the methods differ).

Moving on with the running example of David, Clemens, and Theresa, we now know that there should be 3 edges in total. The weight assigned to these edges is determined by the sum of all edges between two users created by looking at each vocabulary individually (see 5.2). The weight is still undefined, however, a hyperbolic weighting approach came to use. Revisiting the goal of this network and accompanying clustering, we want to get to the similarity of vocabulary between users.

The weighting happens by looking at one vocabulary which has been used by n individual users. The number of users that used the vocabulary at least once illustrates the general popularity of the vocabulary, i.e. how commonly it is used by different users. Taking the number of n users using a specific vocabulary, the weight is defined by:

$$w(n) = \frac{1}{n-1} \tag{2}$$

In the running example, all three users used vocabulary P2067 individually at least one time. This means n would be 3, leading to a weight of 0.5 for all three edges created by it. Moving on to P366, used by Clemens and Theresa, n would be 2. This leads to a weight of 1. Note that a high weight here means that the similarity is high. Then, the two edge-sets for P2067 and P366 are combined. Since Clemens and Theresa already have an edge for P2067, the 0.5 weight is added with the weight of 1, leading to a similarity weight of 1.5 for their connection. On the other hand, both, Theresa and Clemens have a connection to David with a weight of 0.5. Looking at the values, this illustrates that the vocabulary similarity between Theresa and Clemens is higher than the similarity of David with both of them.

The hyperbolic weighting approach is frequently used in both fields, network analysis as well as the sub-field community detection. Newman created one of the more canonical papers focusing on its effectiveness and mechanics. It is a well-cited resource for different kinds of analysis [Newman, 2001].

To sum up the weighting algorithm a bit more formally instead of relying on the two main equations, the following is the more general formula for a weight w_{ij}, describing the weight of the connection between two users i and j. The original algorithm (although not yet named hyperbolic weighting) looks at physics papers' co-authors and determines the strength of relationships between the authors. One paper would be k, and the δ is 1 if the respective user collaborated in paper k:

$$w_{ij} = \sum_k \frac{\delta_i^k \delta_j^k}{n_k - 1} \tag{3}$$

When looking at the formula, one thing becomes imminent: it does not care about how many times a user used a certain vocabulary, but in general, if he used a vocabulary at least once. (In the next section, there is a discussion on this topic in more detail focused on the individual network.)

The final list of edges connecting the nodes with an accompanying weight is called the **edge-list**. It is used to formalize a graph in the python package igraph[20]. Then, the mentioned infomap community clustering approach is used to compute the clustering. The algorithm is also included in the package igraph and in its basis, it works with probability flows of random walks on a network. For more detailed information on the well-accepted and much-used algorithm consult [Rosvall and Bergstrom, 2008][Rosvall et al., 2009].

Finally, after having created the network and computing the clustering, the modularity is an important measure. It measures the strength of the division of a network into modules (e.g. clusters), is computed. This measure can be seen as a main indicator for strong clustering, ranging from 0 (very weak clustering indication) to 1 (perfectly separated). Using the modularity as an indication, plotting (sub-)communities, and visualizing it as well as individual inspection led to the final results.

5.3 Network 1 - mapping similarity of vocabulary used

The network displays the strength of the similarity of vocabulary used between users. Again, the current definition of vocabulary in this analysis is the property used, except in cases where property P31 was used, then it includes

[20] `python.igraph.org`

the object. P31 is the property for "instance of", which is an important part of the general ontology.

This particular decision is made in order to:

1. provide a future-proof parameterized way to include (possibly all) properties including objects
2. give more additional meaning to P31, which is one of the most used properties and in general very important for the ontology
3. be able to show some kind of relation to what P31 tries to express (e.g. P31 referencing to Q5 "human")

The necessary preprocessed data is found in `store_CSV.csv` and an *edge-list* for the network is computed according to the approach detailed in subsection 5.2. Filters were applied in order to make the edge-list feasible to compute. They are listed hereafter in subsection Filtering. The last subsection will focus on the results of the performed clustering.

5.3.1 Filtering

In the framework diagram, Figure 12, one can see that various exclusions are applied for the use case already. The main additional exclusion to the previously detailed exclusions is that the top 21 entities by unique users that used it at least once are cut. Note that 21 can be any arbitrary number and that it was actually aimed for 20, but due to a little coding inaccuracy, the script cut the top 21 vocabularies for both edge lists.

This exclusion is very important performance-wise for this network (in comparison to the next one) since the edge-list creation will form all possible combinations of users using a certain vocabulary. For example, P569 has 39.580 users using it at least once. Mathematically, if one wants to find each combination of two users (not caring about the fact in which order the users are), one would find 783.268.410 possible combinations.

The top 21 properties, in this case, would lead to 5.749.636.702 weights that have to be assigned, whereas moving on from the top 21 to the end, it would only be 2.305.929.094 in total. This little simplification of computation makes it faster, easier and the weights almost lose nothing in terms of meaningfulness, since the weighting is also based on the number of unique users using it (variable n in the formula detailed in subsection 5.2) and the weight would be surprisingly small in comparison to other, less used vocabularies.

5.3.2 Final Overview and preliminary conclusion

Using the full edge-list, subsets are created in order to make it computationally more feasible. The infomap community clustering approach is applied using Python and the library iGraph[21].

For the edge subset 99th quantile, which features 7.1 million rows, it returned a modularity of approx. 0.013. For the 95th quantile, which features 35 million rows, it returned a modularity of approx. 0.006. We then aborted the calculations, since they are computationally heavy, and in general, it is very unlikely that with more edges it will improve.

The modularity and some preliminary analysis on the graph suggest, that there is no naive inherent clustering in the vocabulary similarity. We believe, however, that further and better filtering, in combination with a better filtering algorithm might turn this network into something very useful - and if not, the context provided by Network 2 will probably uncover some correlation (More on this in section 7).

Further concrete steps that are planned regarding this network:

1. **Take the frequency of using a certain vocabulary into account.** As already mentioned, the current version of weighting does not take the general frequency into account. This means that one usage of P500, for example, gives the same relation to this property as a user using it 100 times. In light of the use case, this is rather sub-optimal. Further approaches mainly use different weighting algorithms like the Jaccard Weighting.

2. **Filter low edit count users.** In general, the approach does not really care about users not frequently editing Wikidata in this use case example here. Therefore, it might reduce noise in the data if they are cut before weighting, or the weighting algorithm should take care of this itself by applying lower weights for generally very low edit count users.

3. **Filter very high edit count users (= potential bots).** Analogous to the low edit count users example, very high edit count users using a lot of different vocabularies and/or just using one vocabulary very many times might lead to skewing and might make potential clusters impossible to separate. Also, in this example, either the weighting algorithm or the filters should at least account for this in some way.

[21] `python.igraph.org`

4. **Use a disparity filter, make the data smaller, and provide a pipeline approach to make it easier to test and run.** Points two and three also reference to the point of finding the core structure of the network by omitting data. For example, a disparity filter could solve this problem. The pipeline approach references to the fact to make different parameters easily testable (more on this in section 7).

5.4 Network 2 - mapping similarity of entities used

Network 2 displays the strength of the similarity of entities used between users. As already mentioned, the node here is one user, and the edge between one user and another is the weighted strength of their entity similarity, weighted by the weighting algorithm. subsection 5.2 is detailed for Network 1, however, Network 2 uses the same approach, just switching the vocabulary similarity for the entity similarity. The following subsections go into detail on the mechanics applied and will present a final overview and a preliminary conclusion.

5.4.1 Filtering

In Figure 12, you can see that various exclusions are applied for the use case. The main additional exclusion to the previously detailed exclusions is the fact that the top 21 entities by unique users that used it at least once are removed. This is in order to simplify computation. This should, however, not skew results, since the weighting algorithm takes into account that many users have edited the item, therefore leading to a generally low weight. Therefore, it should not lead to differences in the big picture. Compared to Network 1, this exclusion is not as important, since the edge-list is far easier to compute due to the fact that there are more entities than vocabulary and the average count of users using one entity is by far lower than the count of users using one specific vocabulary.

Further ideas for the general future work on filters would be specific to Network 2. For example, some filters on the entities themselves might be useful, such as removing very low edit count items, which therefore contain very little information, or removing very high edit count items, which might be more general and not very specific to one domain.

5.4.2 Final Overview and preliminary conclusion

Using the full edge-list for Network 2, subsets are created in order to make it computationally more feasible. Then, the infomap community clustering approach is applied using Python and the library iGraph[22].

For the subsets, the 50th quantile version (including 50% of the total edges ranked by weight height), scored the highest modularity of 0.3145. The full edge variant scored a modularity of 0.3116. Because the difference is almost non-existent, the full edge list was chosen to perform the following analysis. This clustering features 133.361 elements and 2.796 clusters, where each element = node = user can only be part of one cluster. This clustering can be considered somewhat successful, therefore the following information on the manual inspection of the clustering is very interesting.

For a quick overview of the clusters and their distribution, see Table 6.

	count users in cluster
total number of clusters	2.796
mean	47,697
standard deviation	2.107,085
minimum	2
25% quantile	2
50% quantile	2
75% quantile	4
maximum	111.397

Table 6: Clustering overview for network 2

Table 6 shows that there are many clusters only containing a few users and one really big cluster. This is the cluster with ID 1, featuring 111.397 users (displayed as “max” in Table 6). From a naive perspective, this could be considered the “core-community”, not especially focusing on a topic. This might also have multiple other reasons, e.g. that it is led by some big user that has a lot of diverse edits or this might be just consisting of very many small edit count users. There will be some more discussion on this later, but first checking the success of the clustering, starting with the top 10 clusters by user size, is more interesting.

One of the most basic evaluation techniques is to get all the entities that led to the clustering and look at them for evidence if the clustering did indeed identify some arbitrary domains. This was not a difficult task for the

[22]`python.igraph.org`

bigger clusters. Quickly querying the Wikidata API for the names of the top 50 most used entities by cluster results in quick insights. The second largest cluster is cluster number 8, featuring 1.534 users. The query for the mentioned 50 top most used entities (by unique users) led to the following result, indicating a strong structure focused on Computer Science. The result of this query is visualized in Table 7.

entity ID	**entity label**	...		
Q698	Mozilla Firefox		Q201904	Qt
Q14579	Linux kernel		Q285741	qBittorrent
Q173136	Blender		Q16639197	GitLab
Q171477	VLC media player		Q41242	Opera
Q59	PHP		Q41540	BIND
Q186055	Git		Q82268	Eclipse
Q10135	LibreOffice		Q170855	Drupal
Q192490	PostgreSQL		Q572226	Kdenlive
Q19841877	Visual Studio Code		Q850	MySQL
Q483604	Mozilla Thunderbird		Q16766305	Atom
Q306144	nginx		Q4046338	Pale Moon
Q756100	Node.js		Q3050461	Elasticsearch
Q13166	WordPress		Q15206305	Docker
Q1165204	MongoDB		Q17363870	KDE Plasma 5
Q48524	Chromium		Q42478	Perl
Q787177	MariaDB		Q286306	curl
Q8038	GIMP		Q965596	Gradle
Q11393	VirtualBox		Q13233410	Android Studio
Q44316	GNOME		Q22906900	Brave
Q14561	Wayland		Q25874683	Nextcloud
Q290284	ownCloud		Q575650	Rust
Q847465	FFmpeg		Q1206660	IntelliJ IDEA
Q1830735	Samba		Q34236	FreeBSD
Q11354	Apache HTTP Server		Q223490	LLVM
Q188558	Wine		Q404293	Falkon

Table 7: Network 2 Cluster 8 entity names visualization

Upon further manual inspection of the cluster, this pattern further increased, while not finding any indicators for even one entity not at least partly belonging to the Computer Science community. The users in this cluster used in total 136.684 entities at least once. Summing all edits they performed shows 338.136 edits in total. However, checking 136.684 items manually for their topic seems rather unrealistic, the samples taken all looked very good.

Moving on to the next smaller cluster, cluster number 2, featuring 1.219 users. Again using the same technique, this cluster seems to be focused on Physics, specifically maybe more advanced physics, with a little focus on seemingly CERN-related topics as well as Higgs Boson and Dark Matter.

Moving on and skipping 3 larger clusters (14, 5, and 13), cluster 32 seems to feature one special kind of domain, having a regional focus. Before computing the clustering, a feeling that some users might have a language or country-specific focus of edits arose, which this cluster seems to give clues to. It almost exclusively features churches, castles, museums, and other points of interest, located in Italy. When querying some of the items for property P17 (= country), it shows that almost all feature P17:Q38, where Q38 is the entity Italy.

At this point, we can conclude that the clustering by only merely looking at the similar entities edited found some meaningful and also big clusters, potentially also uncovering a country-based clustering of interest in users. We therefore argue, that looking into this method of clustering users by entities similarly edited can lead to very distinct identified user communities and may uncover additional classifying attributes like the country example. Moreover, we argue that this clustering is rather sharp in identifying almost exclusively unmixed communities, but since the one huge cluster with number 1 is featuring a very large amount of users, it is probably too sharp.

Many initial points of improvement and potential bias that the clustering may feature arise, but the biggest two concerning this clustering are:

1. Power users might be the “center” of a cluster and the reason why the cluster exists. Take the Italy cluster for example - if a user/bot edits many items in one domain, this might be the reason why the cluster exists, leading to possible bias since there are more clusters, but only the ones featuring a power-user may be identified.

2. The biggest cluster probably features users with a lot of “noise”, e.g. having very diverse topics and entities, as well as very small edit count users. Moving on one could try and reduce this big cluster, as it cannot be used to indicate a domain.

As a first analysis point to get insights into the improvement points, we looked at the average edit count per user and compare it along the clusters. Table 8 shows this and some other metrics that might prove useful. It might be also worthwhile to mention that a previously unknown to us Wikidata Sandbox seems to be the most uniquely edited item in the biggest cluster[23]. The help page lists 4 Wikidata Sandbox items in total, which can be used to try certain edits. This might be a reason for cluster 1 being so big, as they are probably some of the most used items in general. However, because of the weighting algorithm, the weight would be very small if edited by many users individually.

[23]see `https://www.wikidata.org/wiki/Wikidata:Sandbox`

	sum number of edits	count of users	median	max edits per user	Q90 Percentile	average edits per user	indicated main topic for top 50 entities
cluster number							
1	180 678 861	111 397	3	15 222 974	115,0	1 621,9	probably most edited items
8	412 777	1 534	2	84 954	48,7	269,1	Computer Science
2	82 908 577	1 219	8	14 338 645	1 895,0	68 013,6	Physics
14	943 100	797	2	284 970	264,0	1 183,3	Video Games
5	7 187 534	639	8	2 268 423	2 043,4	11 248,1	Botanics
13	1 928 206	497	5	889 471	843,0	3 879,7	Morocco, Egypt, e.g. P17:Q1028
32	655 343	425	8	127 226	731,4	1 542,0	Italian Buildings / P17:Q38
15	529 428	328	7	328 866	236,6	1 614,1	Chemicals / Pharmaceutics
55	1 721 606	306	11	1 140 864	368,0	5 626,2	India P17:Q668
21	100 538	284	14,5	24 716	408,7	354,0	mostly companies, some with German-Region focus
TOTAL	302 885 360	133 361	4	15 222 974	144,0	2 271,2	

Table 8: Top 10 Cluster Comparison for network 2

Further elaborating on Table 8, one can see that cluster number 1, the biggest cluster in terms of users, indeed contains a lot of “noise”, meaning a lot of users with small edit counts. The 90th percentile is 115 edits, so it is probably fair to say that this is the case, especially when compared to the other clusters. One can easily spot that in cluster 1, cluster 2, and cluster 5, there is at least one user with an edit count greater than 2 million edits. These are potentially bots or normal users using scripts for at least some edits. Also, for all the other clusters, one can see that one very big user is present, compared to the 90th percentile.

In light of the results above, concluding for future work, probably removing more users that can be labeled as bots or bot-like activity, cutting the bottom X % of users (since not very interested in non-regular users) and maybe another weighting algorithm like the jaccard weighting approach might be a good idea, possibly increasing modularity and decreasing the size of the biggest cluster number 1.

5.5 General Interpretation of both networks

Network 1, the network mapping similarity of vocabulary used, and therefore the main focus of this thesis, did not reveal significant clustering firsthand as Network 2 did. Further incomplete analysis points and more advanced filtering methods proved insignificant to this point.

Network 2, the network mapping similarity of entities used, revealed a clustered nature, although having a very big cluster that should be reduced and split into sub-components in the future. This suggests that Wikidata Editors can be classified in domains, in the current examples either by focus of interest or possibly geographical reasons.

6 Summary and Conclusion

Wikidata and its accompanying community have matured in the last decade. The information stored in Wikidata is now in use in many applications, from research to other knowledge-driven applications. Since lacking a well-defined ontology definition, it relies on its community to establish consensus on the structure and detail of the information that is stored. Specifically, the underlying schema and the vocabulary used by individual users should be evaluated more concisely in the future. This thesis contributed to this pathway towards analysing the community-driven (sub-)schemas within Wikidata, showing a clear path forward. The possible future outcomes of this path range from analyses over the current state of the knowledge stored in Wikidata and uncovered similarities, but also towards identifying critical evolution steps that lead to a more uniform and "correct" schema. This is essential in ensuring that the world can make the most out of the knowledge stored in Wikidata. As a step towards this goal, we looked at the possible topicwise modulation of Wikidata by clustering users by their used vocabulary along the research question: "Can we show a topicwise modularisation of Wikidata by clustering users by their used vocabulary?".

We processed the Wikidata edit history to a full extent and published said script framework on GitHub under `https://github.com/N-Krenn/WIKIEVO`. The resulting datasets that we identified to be useful for further research are published and linked in the same GitHub repository. The scripts and datasets are licensed under an MIT license, so everyone wanting to adapt or contribute anything is very welcome.

`store.csv` contains all user edits, specifically additions and deletions, excluding redirect items up until the 1^{st} of January 2023. This is, to the best of our knowledge, the largest publicly available research-intended dataset focusing on the Wikidata edit history to this day.

`store_UVEC.csv` features all edits adding content to the Wikidata, given certain limitations. It aggregates individual edits performed by users using a certain vocabulary on an entity. The dataset is also publicly available and more focused on the use case presented in the thesis, already aggregating our definition of a *vocabulary*. We used said dataset to perform the analyses, again transforming it in different ways.

Besides adding more recent descriptive statistics on Wikidata to the research landscape, we also carried out a preliminary network analysis focused on the goal of detecting community-driven (sub-)schemas within Wikidata. For the network analysis, we conclude that mapping the users' entity similarity in a network leads to sharp clusters, potentially providing a way to identify editor communities focusing on specific domains (or topics) within Wikidata. The mapping of the users' vocabulary similarity, however, is to this point still very unclear. Initial methods did not show a clustered structure.

Therefore, the answer to the proposed Overarching Research Question "Can we show a topicwise modularisation of Wikidata by clustering users by their used vocabulary?" is partly considered answered. We outlined topicwise modularisation by directly clustering the edited Wikidata entities and showed some examples of clear focus on particular domains. The vocabulary aspect, however, is still inconclusive to this point.

We propose a future methodological approach in order to sharpen the networks, possibly improving the domain clustered structure or identifying a vocabulary similarity clustering structure. The newly described approach focuses on a weighted Jaccard similarity measure and a following disparity filter approach, which could be applied in a pipeline-structured way. Its preliminary details are outlined hereafter in section 7. After trying to improve the networks individually, we also see an opportunity to transfer the results of the domain clustered structure to the vocabulary similarity clustering in a case study of hand-picked domains.

In sum, we contribute to the state of the art by providing three distinct results: the datasets, the framework and the analysis. The mentioned datasets are the first ones of their kind, enabling future researchers to perform their own analyses on the data, along with the means necessary to craft such datasets in the first place, our framework with its parameters. The preliminary analysis of the datasets towards showing topicwise modularisation of Wikidata by clustering their users by their used vocabulary shows clear signs of users focusing on certain domains. The vocabulary similarities still require some work and are inconclusive, to this point in time.

7 Outlook

The creation of the data needed for the analysis and the general crafting of the networks has taken a lot of work. Now, having finished the groundwork, we think that even further modularizing and streamlining the network clustering approach is the main future goal. This "pipeline" approach gives the opportunity to try different things with different parameters, without having to manually change anything but the parameters.

For example, [May et al., 2019] used such a parameterized approach in order to arrive at the results. This leads to increased productivity and more possibility to try out different things. This would imply that the filtering, weighting, and network creation should be parameterized and easily adaptable.

The paper also features a weighted Jaccard similarity measure, where the new weight is defined by:

$$s(u,v) = \frac{\sum_{t \in T} min(t_u, t_v)}{\sum_{t \in T} max(t_u, t_v)} \tag{4}$$

Where u and v are two users and t is one element out of T elements. Using the language of Network 2, mapping the users' entity similarity, T is the set of all elements, and t_u and t_v are then the absolute count the explicit user used this one single entity. This way of weighting in order to get a similarity measure is accepted practice [Chierichetti et al., 2010]. This new similarity measure leads to two big shifts in logic compared to the previously used hyperbolic weighting approach:

1. the frequency somebody uses an entity/vocabulary is now also taken into account
2. whether the entity/vocabulary is used by many users or few does not change anything in the weight (compared to the previous hyperbolic approach).

The Jaccard Weighting approach opens new discussions and leads to new and different improvements. For example, taking the general popularity of the vocabulary/entity into account by e.g. simple multiplication of the new weight with e.g. a hyperbolic weighting approach could lead to improvement. The weights themselves could also potentially lead to new interesting insights into the way users contribute to Wikidata, outlining a general popularity measure of vocabularies.

After applying this new weighting, there is an opportunity to apply a principled statistical filter to remove low-weight edges in order to improve computational feasibility and in order to reduce noise. The user-to-user networks, may it be Network 1 or Network 2, are very dense, under whatever filtering used. There are lots of low-weight edges because there are entities/vocabularies that most users edit sooner or later. We propose to drop these edges, however, because of the complex distribution of edge weights, a simple threshold, like the quantile approach used in the analysis, is rather unsatisfying. Reviewing what we really want to get out of the networks, we want to see the core structure, the backbone so to say. Serrano et al. created a method called "disparity filter" to filter such weighted networks, extracting the so-called “backbone” of a network. They applied their method in a case study to airport travel networks [Ángeles Serrano et al., 2009].

Without going into further detail on the statistical/mathematical details of this mentioned filter, the main relation deciding if an edge is relevant is:

$$\alpha_{i_j} = 1 - (k-1) \int_0^{p_{i_j}} (1-x)^{k-2} dx < \alpha. \tag{5}$$

Where p_{ij} is the share of a node i's total weight with node j, and k is the number of connections of node i. This gives each weighted edge in the network a “statistical significance” defined locally. This to a large extent has the potential to solve the problem that we have huge heterogeneity in the activity of users (i.e. powerlaws). At the same time, we can define a global filter, possibly even further increasing potential.

This is the main suggestion on the way to go forward and apply more advanced weighting and filtering techniques, however, we also see a lot of potential in improving the base data and manual inspection techniques already mentioned in the sections on the respective network.

The main ideas regarding improvement of the base data are:

- take care of the powerlaws in the data directly (e.g. cut low edit-count users out of the data, as they are not as interesting for the use case)
- cut non-essential items like the Wikidata Sandbox Items[24]
- play around with different settings for the vocabulary definition, possibly annotating more properties with the object (e.g. P17 (country), which already showed some effects regarding domain-clustering)

Furthermore, one direction to get to some clustered structure regarding the vocabulary would be to refine the “domain clustering” (network 2) and then take the findings from there and project them onto the vocabulary similarity clustering (network 1). In exact terms, one could analyse a certain “domain community” regarding their vocabulary and compare and contrast this to other “domain communities”, showing concrete examples of vocabulary differences between domains. This even further enhances the “community-first” approach that the overall thesis follows.

[24]see `https://www.wikidata.org/wiki/Wikidata:Sandbox`

In sum, since network analysis is almost always focused on trying new things in order to get to improvement, we see many paths to be further examined, leading to an improved overall result and possible new findings. The future work presented along with the code details and datasets holds enormous value in understanding Wikidatas (sub-)schema and community structure from a very specific point of view, but also in a general way.

8 Limitations

The analyses performed all have one common weakness: the input data.

As pointed out in the different steps, we took decisions in order to either reduce the sheer size of data because of limited computing resources (e.g. neglecting deletions, cutting top N vocabulary, ...) and decisions that made conceptual analysis easier (neglecting redirect items, neglecting bots, neglecting semi-automated user edits, ...). In a "garbage in, garbage out" fashion, this may lead to slightly skewed conclusions.

Moreover, the bot identification is at risk of producing wrongful inclusions/exclusions. There is always the chance that either bot edits find their way into the analysis set, or vice-versa non-bot edits get flagged as bot edits. This might also lead to possible hurtful implications. However, we are very confident that all exclusions are for the benefit of the analyses, while carefully weighing the possible effects.

Additionally, potential interactions between the various variables in the network analysis might lead to bias in the clustering. This is, however, based on the nature of the heuristic approach to cluster the communities. In the future, we would like to look into the interactions and statistics behind the way users contribute to Wikidata in far more detail.

9 Related Work

There is quite some prior research in many of the surrounding topics to this thesis, also some explicitly targeting the Wikidata edit history. In the following chapter, we will try to sum up some of the more prominent work that is not included in subsection 2.2. The goal is to differentiate the thesis from the papers.

The study “Wikidatians Are Born: Paths to Full Participation in a Collaborative Structured Knowledge Base” investigated how newcomers join the Wikidata community, showing their path to becoming frequent editors. The approach was mainly qualitative, featuring interviews with Wikidata community members [Piscopo et al., 2016].

Comparing to the almost identically named study “Wikidatians are Born [...]” an older paper by Panciera et al. looked at Wikipedia from a power-user perspective, analysing how “Wikipedians” produce most of the value of Wikipedia [Panciera et al., 2009]. The paper gives broad insights into the structure of the contributors to Wikipedia.

The same authors as in “Wikidatians are Born [...]” also looked at Wikidata from the same perspective as this thesis, although focusing on the users contributing and not the (sub-)schema. They found that the Wikidata ontology “has uneven breadth and depth”. Additionally, they identified two user roles: contributors and leaders, where the leaders have a positive effect on ontology depth, as the knowledge graph evolves [Piscopo and Simperl, 2018]. All these findings contribute insights into how Wikidata Users contribute, but are not focused on what they contribute.

However, regarding the Wikidata usage of properties, there is some informative study by Haller et al. under the title “An Analysis of Links in Wikidata”. An important sidenote of the study is that Wikidata has large numbers of entities as well as properties that are not (yet) extensively linked. It also suggests different directions in order to “increase the interconnectedness of Wikidata with other KGs” [Haller et al., 2022]. They also provide differences in labeling between the RDFS/OWL property standard and the way Wikidata uses the properties. This thesis relates to the paper in some sense, since the way these properties are used also very likely has a temporal difference.

In “What we talk about when we talk about Wikidata quality: a literature survey”, the authors gathered information about how the current research addressed quality in Wikidata. They focused on Intrinsic Dimensions (Accuracy, Trustworthiness, and Consistency) and Contextual Dimensions (Relevancy, Completeness, and Timeliness) as well as Representation Dimensions and Accessibility Dimensions. [Piscopo and Simperl, 2019] The paper at hand gives relevant pointers towards important quality metrics, which may be used in the future of the research path of the thesis to also compare the identified communities in other aspects, such as quality.

Cantallops et al. performed a systematic literature review on Wikidata in 2019 [Mora-Cantallops et al., 2019]. They also suggest that communities and their grouping, which could also be called the "social network" behind Wikidata, is an important route to explore further. We add to this perspective by analysing two aspects, the domain, and the vocabulary.

A User Manual of the framework

The following appendix provides the main instructive resources for the scripts produced and described. The first subsection displays the General Instructions on how to run or modify the framework, giving additional insights. The second subsection then displays the code of the main-handler including the parameters offered at the time. Both are part of the GitHub repository listed under `https://github.com/N-Krenn/WIKIEVO`. The provided datasets are also linked through the repository.

A.1 General Instructions

WIKIEVO

This is the project code of a research project published under link coming soon. It was originally developed as a Master Thesis by Nicola Pascal Krenn (nicola.pascal.krenn@s.wu.ac.at) with supervisors Univ.- Prof. Dr. Axel Polleres and Assoc.- Prof. Dr. Johannes Wachs at the Vienna University of Business and Economics.

What does it do?

It provides utilities for analysing the edit history of Wikidata. It also contains the scripts creating the networks used in the corresponding master thesis.

It holds three essential parts:

- preprocessor (parsing the Wikidata "pages-meta-history" dumps)
- extractor (getting the wanted data out of the preprocessed files into memory / file
- network_prep (the edge-creation and network creation / clustering performed for the paper)

What is the workflow?

If you just want to use the already preprocessed and published dataset (link coming soon) as mentioned in the paper, you follow the steps in "Easy Setup with Dump 01-01-2023 If you want to use another dump, you have to process the data using the provided preprocessor and extractor. Follow the steps in "Advanced Setup and Dump processing". Please bear in mind that the Limitations and Problems mentioned in the paper apply to all work built upon this code and it might not run on machines weaker than the mentioned setup.

Configuration

We provide three versions of main.sh for different use-cases if you either want to use our published inbetween-data-points or want to start processing your own dump. Modify the parameters in this file in order to make changes according to your use-case. We provide explanations of the parameters in the file directly. The standard values used in the thesis are always set in the default variant.

Packages

Please install the following packages in your preferred way before running the scripts:

- pandas
- modin (https://modin.readthedocs.io/en/stable/index.html) including the ray engine
- iGraph (https://igraph.org/python/)

Additional packages or a specific version of them might be needed. We did not test the framework extensively on different versions or machines.

Easy Setup with Dump 01-01-2023

We used the wikidata pages-meta-history dump of 01-01-2023 for our research. This is essentially what the main analysis works on. If you want to use this dataset, download it here (link coming soon) and put it into ./extractor/ named store.csv (uncompress it first). Using the main.sh bash file, you should set the options preprocess (line 12) and extract (line 13) to false, since you do not need to run these steps, because you already inputted their results.

Sub-Variant of Easy Setup with Dump 01-01-2023 rerunning the extractor

If you want to modify parameters of the extractor, e.g. the list of vocabulary items that should include the object, you download this dataset LINK and put it into ./preprocessor/output/...subfolders... in uncompressed format. Using the main.sh bash file, you should set the options preprocess (line 12) to false, since you do not need to run the preprocessor, because you already inputted its results.

Advanced Setup and Dump processing (Warning: It takes time and resources)

The preprocessor does nothing else than parsing the extensive amounts of the .bz2 compressed wikidata dump provided. It should be pretty self-explanatory. Beforehand, you have to download one complete "pages-meta-history" dump to a folder of your choice (or preprocessor/dumps as default). This should already take you several days because of limited download-rates. You should have all the files in the folder directly listed.

After downloading the dumps and installing the packages, you may run main.sh as the main handling bash script. Make sure to configure it to your planned setting beforehand. Lines 12 to 22 should, however, stay as they are. Make sure to be familiar with the settings before running.

We used a machine featuring a 3TB SSD, 48 v-Cores and 128GB RAM for this project. The first step is mostly CPU-bound and does not really need much RAM, but with 46 cores in use it still took us around 40 hours of computing.

Being finished with the preprocessing, we need to extract the things that we want to look at into a simpler format for further analysis. We chose package modin, a scalable pandas derivative, for the handling. This is essentially only possible because of the 128GB RAM. Weaker machines might have a harder time / run into OOM errors.

The resulting datasets are saved to ./network_prep and this is where the .ipynb notebook for the analysis gets its starting data from. Voila - you have your own dump configured. (Note that you might want to change the data that extractor extracts to your use-case or to extend the preprocessor in order to fit your analysis goals)

License

Hereby, I declare that I license the contents of this GitHub repository under the MIT license whereas I license all datasets that are created with the aforementioned script under the CC BY-SA 4.0 License.

A.2 Main Handler and Parameters

Configuration/Parameter section:

```
###set the process steps to false if you do not want to do them right now or you already completed them
        successfully!

#$getlatestdump=true future option, for now we do not have anything for this, wget loop should be enough
preprocess=true
extract=true
store_UVEC_and_VOC=true
store_entity=true
network_prep_1=true
network_prep_2_reduced=true

#Try this option if network_prep_reduced does not work because of too much RAM used. Careful, only set one of them
        true or the script
#But be careful, it creates one file for every entity (which in study case is >20Mio), takes very long.
#Instead of this option we suggest adapting the code of user_weights_network_2.py or just running the above option
        with a higher cut_top_N_entities option set
network_prep_2=false



###Detail Configuration Options###
perform_cut="n" #Options y or n, yes if you want to only extract until a certain specified items creation date and
        time, restricting the time-frame (File = extractor.py)
cut_entity="Q1000" #Enter the Wikidata entity qualifier for cutting here. Note that if perform_cut == "n", this does
        not do anything. Used in file: (File = extractor.py)

cut_top_N_vocabs="20" #specifies the amount of top N most used properties/vocabulary you want to drop. Default is
        20, since the most used will lead to hefty amounts of edges and weights to process.
#Used in file: (File = user_weights_reduced.py)
#cut_top_N="-1" #drop nothing
cut_top_M_vocabs="10" #specifies the amount of top M most used properties/vocabulary of syntax P31:OBJECT you
        want to drop. Default is 10, since the most used will lead to hefty amounts of edges and weights.
# Used in file: (File = user_weights_reduced.py)
#cut_top_m="-1" #drop nothing

cut_top_N_entities="0" #specifies the amount of top N most used properties/vocabulary you want to drop. Default is
        0, since entities do not necessarily need this reduction.
#Used in file: (File = user_weights_reduced.py)
#cut_top_N="-1" #drop nothing
cut_top_M_entities="0" #specifies the amount of top M most used properties/vocabulary of syntax P31:OBJECT you
        want to drop. Default is 0, since entities do not necessarily need this reduction.
# Used in file: (File = user_weights_reduced.py)
#cut_top_m="-1" #drop nothing


quantile_list =" [0.99,0.95,0.50,0.00] " #This defines the quantile subsets (computed in exactly this order) that we
        create for the edge weights which go into the graphs.
#Used in file: (File = network_utils.py)

#Additional setting for "vocabulary" definition:
#In our study example, P31 (instanceof) is not handled like the other properties. It gets a suffix e.g. P31:Q1000,
        whereas normal properties are just handled like P32, with no object string attached. See code excerpt below
#excerpt from extractor.py:
#We compute the "vocabulary" variable, which in most cases is only the property, but in some cases when we can
        identify the property as being one of the most used, we will take the object into this IF the object is an entity
#    list_most_common_properties = ["31"] #31 = instanceOf
#    proprty = list_result [3]
#    objct = list_result [4]
#
#    if proprty in list_most_common_properties and objct.isdigit():
#        vocabulary = "P" + proprty + ":" + objct
#    else:
#        vocabulary = "P" + proprty

list_most_common_properties="[31]" #Standard option
#list_most_common_properties="[]" #Disables this option
#list_most_common_properties="[31,17,354]" #Creates PXXX:QObjectNr for every P31, P17 and P354 if object is
        numerical = entity
```

Execution section:

```
### RUN PREPROCESSOR ###
if [ "$preprocess" = true ]; then
    cd preprocessor
    if python3 preprocessor.py; then
        echo "Exit code of 0, PREPROCESSOR reported success"
    else
        echo "Exit code of $?, FAILURE in PREPROCESSOR. Please study console output and adapt
              parameters/files."
        exit
    fi
    cd ..
fi


### RUN EXTRACTOR ###
if [ "$extract" = true ]; then
    cd extractor
    if python3 extractor.py $perform_cut $cut_entity $list_most_common_properties; then
        echo "Exit code of 0, EXTRACTOR reported success"
    else
        echo "Exit code of $?, FAILURE in EXTRACTOR. Please study console output and adapt parameters/files."
        exit
    fi
    cd ..
fi

### CREATE STORE_UVEC AND STORE_VOC (for first network and possible analysis ###
if [ "$store_UVEC_and_VOC" = true ]; then
    cd network_prep
    if python3 create_store_voc_and_UVEC.py; then
        echo "Exit code of 0, CREATE_STORE_VOC_AND_UVEC reported success"
    else
        echo "Exit code of $?, FAILURE in STORE_VOC_AND_UVEC. Please study console output and adapt
              parameters/files."
        exit
    fi
    cd ..
fi


if [ "$store_entity" = true ]; then
    cd network_prep
    if python3 create_store_entity.py; then
        echo "Exit code of 0, CREATE_STORE_ENTITY reported success"
    else
        echo "Exit code of $?, FAILURE in CREATE_STORE_ENTITY. Please study console output and adapt
              parameters/files."
        exit
    fi
    cd ..
fi




#### CREATION OF FIRST NETWORK 1_network_user_to_vocab ####
if [ "$network_prep_1" = true ]; then
    folder_network_to_build="./1_network_user_to_vocab/"
    filename_edges="user_weights.csv"
    network_to_build="1"

### USER WEIGHTS CREATION ###
    cd network_prep
    if python3 user_weights_reduced.py $cut_top_N_vocabs $cut_top_M_vocabs $network_to_build; then
        echo "Exit code of 0, USER_WEIGHTS_REDUCED reported success"
    else
        echo "Exit code of $?, FAILURE in USER_WEIGHTS_REDUCED. Please study console output and adapt
              parameters/files."
        exit
    fi

    #Catting and Sorting Buffer
    cd ./file_buffer
    sort buffer_user_weights_* > buffer_cat_sorted
    rm buffer_cat
    cd ..

### USER WEIGHTS BUFFER COMBINE###
```

```
    if  python3 user_weights_buffer_combine.py $network_to_build; then
        echo "Exit code of  0,  USER_WEIGHTS_BUFFER_COMBINE reported success"
    else
        echo "Exit code of $?,  FAILURE in USER_WEIGHTS_BUFFER_COMBINE. Please study console output and
               adapt parameters/files."
        exit
    fi

###network_utils create basic graph and infomap community clustering, print modularity###
    if  python3 network_utils.py $filename_edges $folder_network_to_build  $quantile_list ; then
        echo "Exit code of  0,  NETWORK_UTILS reported success"
    else
        echo "Exit code of $?,  FAILURE in NETWORK_UTILS. Please study console output and adapt
               parameters/files."
        exit
    fi
    cd ..
    echo "Exit code 0 for  creation  of  Network 1_network_user_to_entities"
fi


#### CREATION OF SECOND NETWORK 2_network_user_to_entities ####
if  [ "$network_prep_2_reduced" = true ]; then
    folder_network_to_build="./2_network_user_to_entities/"
    filename_edges="user_weights.csv"
    network_to_build="2"

    ### USER WEIGHTS CREATION ###
    cd network_prep
    if  python3 user_weights_network_2.py $cut_top_N_entities $cut_top_M_entities  $network_to_build; then
        echo "Exit code of  0,  USER_WEIGHTS_NETWORK_2 on network 2 reported success"
    else
        echo "Exit code of $?,  FAILURE in USER_WEIGHTS_NETWORK_2. Please study console output and adapt
               parameters/files."
        exit
    fi

    ###network_utils create basic graph and infomap community clustering, print modularity###
    if  python3 network_utils.py $filename_edges $folder_network_to_build  $quantile_list ; then
        echo "Exit code of  0,  NETWORK_UTILS on network 2 reported success"
    else
        echo "Exit code of $?,  FAILURE in NETWORK_UTILS on network 2. Please study console output and adapt
               parameters/files."
        exit
    fi
    cd ..
    echo "Exit code 0 for  creation  of  Network 2_network_user_to_entities"
fi



#### CREATION OF SECOND NETWORK 2_network_user_to_entities ALTERNATIVE METHOD! DO NOT USE
     IF NOT NECESSARY!####
if  [ "$network_prep_2" = true ]; then
    folder_network_to_build="./2_network_user_to_entities/"
    filename_edges="user_weights.csv"
    network_to_build="2"

### USER WEIGHTS CREATION ###
    cd network_prep
    if  python3 user_weights_reduced.py $cut_top_N_entities $cut_top_M_entities  $network_to_build; then
        echo "Exit code of  0,  USER_WEIGHTS_REDUCED on network 2 reported success"
    else
        echo "Exit code of $?,  FAILURE in USER_WEIGHTS_REDUCED on network 2. Please study console output
               and adapt parameters/files."
        exit
    fi

    #Catting and Sorting Buffer
    cd ./ file_buffer_2
    sort  buffer_user_weights_* > buffer_cat_sorted
    cd ..

### USER WEIGHTS BUFFER COMBINE###
    if  python3 user_weights_buffer_combine.py $network_to_build; then
        echo "Exit code of  0,  USER_WEIGHTS_BUFFER_COMBINE on network 2 reported success"
    else
        echo "Exit code of $?,  FAILURE in USER_WEIGHTS_BUFFER_COMBINE on network2. Please study console
               output and adapt parameters/files."
        exit
    fi

###network_utils create basic graph and infomap community clustering, print modularity###
```

```
    if python3 network_utils.py $filename_edges $folder_network_to_build  $quantile_list ; then
        echo "Exit code of 0, NETWORK_UTILS on network 2 reported success"
    else
        echo "Exit code of $?, FAILURE in NETWORK_UTILS on network 2. Please study console output and adapt
              parameters/files."
        exit
    fi
    cd ..
    echo "Exit code 0 for creation of Network 2_network_user_to_entities"
fi

echo "Exit code of 0, all requested functions completed successfully."
```

References

[Arenas et al., 2004] Arenas, A., Danon, L., Díaz-Guilera, A., Gleiser, P. M., and Guimerá, R. (2004). Community analysis in social networks. *The European Physical Journal B*, 38(2):373–380.

[Baroncini et al., 2022] Baroncini, S., Martorana, M., Scrocca, M., Śmiech, Z., and Polleres, A. (2022). Analysing the Evolution of Community-Driven (Sub-)Schemas within Wikidata.

[Boy and Uitermark, 2016] Boy, J. D. and Uitermark, J. (2016). How to study the city on instagram. *PLOS ONE*, 11(6):1–16.

[Catanese et al., 2011] Catanese, S. A., De Meo, P., Ferrara, E., Fiumara, G., and Provetti, A. (2011). Crawling facebook for social network analysis purposes. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, WIMS '11, New York, NY, USA. Association for Computing Machinery.

[Chierichetti et al., 2010] Chierichetti, F., Kumar, R., Pandey, S., and Vassilvitskii, S. (2010). Finding the jaccard median. pages 293–311.

[Cimiano and Paulheim, 2017] Cimiano, P. and Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semant. Web*, 8(3):489–508.

[Farda-Sarbas and Müller-Birn, 2019] Farda-Sarbas, M. and Müller-Birn, C. (2019). Wikidata from a research perspective – a systematic mapping study of wikidata.

[Ferrara, 2012] Ferrara, E. (2012). A large-scale community structure analysis in facebook. *EPJ Data Science*, 1(1):9.

[Hall et al., 2018] Hall, A., Terveen, L., and Halfaker, A. (2018). Bot detection in wikidata using behavioral and other informal cues. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW).

[Haller et al., 2022] Haller, A., Polleres, A., Dobriy, D., Ferranti, N., and Rodríguez Méndez, S. J. (2022). An analysis of links in wikidata. In Groth, P., Vidal, M.-E., Suchanek, F., Szekley, P., Kapanipathi, P., Pesquita, C., Skaf-Molli, H., and Tamper, M., editors, *The Semantic Web*, pages 21–38, Cham. Springer International Publishing.

[Hogan et al., 2021] Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., Ngomo, A.-C. N., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., and Zimmermann, A. (2021). Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37.

[Krabina and Polleres, 2021] Krabina, B. and Polleres, A. (2021). Seeding wikidata with municipal finance data. In *Wikidata@ISWC*.

[Lafraie, 2020] Lafraie, R. (2020). Representing evolving knowledge graphs through incremental embeddings masterarbeit.

[May et al., 2019] May, A., Wachs, J., and Hannák, A. (2019). Gender differences in participation and reward on stack overflow. *Empirical Software Engineering*, 24(4):1997–2019.

[Mora-Cantallops et al., 2019] Mora-Cantallops, M., Sánchez-Alonso, S., and Barriocanal, E. (2019). A systematic literature review on wikidata. *Data Technologies and Applications*, ahead-of-print.

[Mountantonakis and Tzitzikas, 2023] Mountantonakis, M. and Tzitzikas, Y. (2023). Using multiple rdf knowledge graphs for enriching chatgpt responses.

[Newman, 2001] Newman, M. E. J. (2001). Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Phys. Rev. E*, 64:016132.

[Panciera et al., 2009] Panciera, K., Halfaker, A., and Terveen, L. (2009). Wikipedians are born, not made: A study of power editors on wikipedia. In *Proceedings of the 2009 ACM International Conference on Supporting Group Work*, GROUP '09, page 51–60, New York, NY, USA. Association for Computing Machinery.

[Pellissier Tanon and Suchanek, 2019] Pellissier Tanon, T. and Suchanek, F. (2019). Querying the edit history of wikidata. In Hitzler, P., Kirrane, S., Hartig, O., de Boer, V., Vidal, M.-E., Maleshkova, M., Schlobach, S., Hammar, K., Lasierra, N., Stadtmüller, S., Hose, K., and Verborgh, R., editors, *The Semantic Web: ESWC 2019 Satellite Events*, pages 161–166, Cham. Springer International Publishing.

[Piscopo et al., 2016] Piscopo, A., Phethean, C., and Simperl, E. (2016). Wikidatians are born: Paths to full participation in a collaborative structured knowledge base.

[Piscopo et al., 2017] Piscopo, A., Phethean, C., and Simperl, E. (2017). What makes a good collaborative knowledge graph: Group composition and quality in wikidata. In Ciampaglia, G. L., Mashhadi, A., and Yasseri, T., editors, *Social Informatics*, pages 305–322, Cham. Springer International Publishing.

[Piscopo and Simperl, 2018] Piscopo, A. and Simperl, E. (2018). Who models the world? collaborative ontology creation and user roles in wikidata. *Proc. ACM Hum.-Comput. Interact.*, 2(CSCW).

[Piscopo and Simperl, 2019] Piscopo, A. and Simperl, E. (2019). What we talk about when we talk about wikidata quality: A literature survey. In *Proceedings of the 15th International Symposium on Open Collaboration*, OpenSym '19, New York, NY, USA. Association for Computing Machinery.

[Rosvall et al., 2009] Rosvall, M., Axelsson, D., and Bergstrom, C. T. (2009). The map equation. *The European Physical Journal Special Topics*, 178(1):13–23.

[Rosvall and Bergstrom, 2008] Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

[Sarasua et al., 2019] Sarasua, C., Checco, A., Demartini, G., Difallah, D., Feldman, M., and Pintscher, L. (2019). The evolution of power and standard wikidata editors: Comparing editing behavior over time to predict lifespan and volume of edits. *Computer Supported Cooperative Work (CSCW)*, 28.

[Schmelzeisen et al., 2021] Schmelzeisen, L., Dima, C., and Staab, S. (2021). Wikidated 1.0: An evolving knowledge graph dataset of wikidata's revision history.

[Steiner, 2014] Steiner, T. (2014). Bots vs. wikipedians, anons vs. logged-ins (redux): A global study of edit activity on wikipedia and wikidata. In *Proceedings of The International Symposium on Open Collaboration*, OpenSym '14, page 1–7, New York, NY, USA. Association for Computing Machinery.

[Vrandečić et al., 2023] Vrandečić, D., Pintscher, L., and Krötzsch, M. (2023). Wikidata: The making of. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, page 615–624, New York, NY, USA. Association for Computing Machinery.

[Watts and Strogatz, 1998] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684):440–442.

[West et al., 2001] West, D. B. et al. (2001). *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River.

[Ángeles Serrano et al., 2009] Ángeles Serrano, M., Boguñá, M., and Vespignani, A. (2009). Extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences*, 106(16):6483–6488.