Strategies for creating knowledge graphs to depict a multi-perspective Queer communities representation

Louann Coste
Ecole Normale Supérieure de Lyon,
15 parvis René Descartes
Lyon
louann.coste@ens-lyon.fr

Flora Helmers
Ecole Normale Supérieure de Lyon,
15 parvis René Descartes
Lyon
flora.helmers@ens-lyon.fr

Hammamache Kheddouci
CNRS, Univ Lyon, INSA Lyon,
UCBL, LIRIS, UMR5205
Villeurbanne
hamamache.kheddouci@
univ-lyon1.fr

Léo Le Nestour
Ecole Normale Supérieure de Lyon,
15 parvis René Descartes
Lyon
leo.le_nestour@ens-lyon.fr

Mahsa Niazi
Ecole Normale Supérieure de Lyon,
15 parvis René Descartes
Lyon
mahsa.niazi@ens-lyon.fr

Genoveva Vargas-Solar
CNRS, Univ Lyon, INSA Lyon,
UCBL, LIRIS, UMR5205
Villeurbanne
genoveva.vargas-solar@cnrs.fr

ABSTRACT

This paper introduces an experimental study for building knowledge graphs about queer communities with different perspectives. The paper describes a set of pipelines implementing several knowledge graphs construction strategies that lead to a complementary understanding of the queer communities. We implemented a library with the knowledge graph construction pipelines and conducted experiments to evaluate execution cost against the comprehensiveness of the content. The library includes visualization tools for observing the knowledge graphs concerning several granularities. Finally, we use storytelling to provide a first interpretation of our results regarding the profiling of queer communities.

1 INTRODUCTION

Vast collections of heterogeneous data containing observations of phenomena can be structured into networks with interconnection rules determined by the variables (i.e., attributes) characterising each observation. The notion of "graph" is a powerful mathematical concept for representing these networks as graphs. The networks can be implemented by efficient data structures and exploited by applying different types of algorithms composed of workflows to solve data science problems. When the graphs become significant and even too large, the algorithms used to process, explore, and analyse them become costly in execution time, even if several cores are used. In this case, given the characteristics of the algorithms, communication is also likely to be expensive. So, workflows that exploit graphs become greedy consumers of computing resources. Intelligent deployment strategies on target architectures must be designed. Just-in-time architectures are a possible solution to fulfil the resources' consumption requirements in an adaptable way [19].

This paper introduces an experimental approach for building knowledge graphs at different scales to reveal several perspectives of Queer History, which has remained hidden in the universal History. Our experiment uses wiki data as the data source. It applies different graph operations/queries and machine learning algorithms to build knowledge graphs of various sizes that contain several queer identities and communities.

Our work models the pipelines used to explore the wiki data knowledge graph and extract subgraphs about queer identities combining different variables like age, artistic production, geographical position, etc. Extracted graphs are studied, classified, and profiled thoroughly to estimate the allocation of resources. The pipelines include performing unions, merge, and community discovery to build several perspectives of Queer History. Developing and modelling these pipelines enables testing, benchmarking, and comparing knowledge graph-building pipelines in different architectural settings.

The remainder of the paper is organised as follows. Section 2 introduces work addressing the construction and exploitation of knowledge graphs. Section 3 introduces our approach based on different pipelines proposing strategies for building the queer history with knowledge graphs that provide different perspectives of this history and considers available resources for deciding about the scope of the graphs. Section 4 reports our experiments where we test different strategies for building knowledge graphs of various sizes and scopes to explore the queer history. Finally, Section 5 concludes the paper and discusses future work.

2 RELATED WORK

The work introduced in this paper is related to techniques and approaches used for building and exploring knowledge graphs. Most works we studied view knowledge graphs as pivot models to integrate data and model knowledge contained in different data collections. Besides the knowledge representation problem, there are construction and exploration cost issues, particularly when the graphs grow big and nodes are highly connected. We rely on knowledge graphs to perform an analytics project about queer history. Our project adheres to existing initiatives that use graphs for modelling the history of queer communities. This section discusses the principles of these projects too.

Knowledge graphs construction and exploration. A knowledge graph is a directed graph G = (N,R) whose nodes N represent entities and literal values (literals), and whose edges R represent relations between these entities. The graph can be divided into two parts: the data and the knowledge. The knowledge can come from several sources that describe entities (i.e., concepts) and from relations among entities that can be explicitly stated or discovered [12]. Depending on the model used to define a

knowledge graph the query language can change. Knowledge graphs defined using the RDF model can be queried using the SPARQL query language. In the web there are lot of examples about Knowledge Graphs like DBPedia a cross-domain knowledge graph with factual information extracted from Wikipedia pages. It was created manually with around 768 classes and 3,000 properties. Wikidata also a cross-domain knowledge graph with the facts created and edited by humans and bots. It has contexts for statements like validity time, and reference. Open Research Knowledge Graph (ORKG) is a description of research papers in a structured manner to make research papers easier to find and compare.

Knowledge graphs are used to integrate bibliographical data and exhibit references and papers co-authorship. Examples of solutions are DIG [18], Knowledge Vault [4], CiteSeerX [14], Pujara et al. [15], the Knowledge Graph Identification [16] and NELL [3]. The authors of [8] propose a framework for incrementally building a knowledge graph of artists used to link the data from different museums. An initial knowledge graph is built from a data source and it is incrementally extended through the use of the MinHash/Locality-sensitive Hashing (LSH) algorithm. These approaches rely on a predefined schema describing the data used to build the graph. Other approaches, adopt the opposite strategy and do not use a schema for building knowledge graphs. For example, Reverb [6], OLLIE [5], and PRISMATIC [7]. Knowledge graphs can also be built by consolidating data from different sources like the Linked Data Integration Framework (LDIF [1]), YAGO and YAGO2 [10, 17] and Freebase [2].

Modelling the Queer History with graphs. Queer history was not recorded in a structured way for a long time. The most prominent project willing to integrate the history of LGBTQ+ communities is the WikiProject LGBT [1] for developing LGBT-themed content in Wikidata and advocating for the community of contributors developing this content. This project provides a forum for central discussion of topics of interest to stakeholders in LGBT+ content. The community develops recommendations for modelling LGBT+ content, presents queries for showcasing LGBT content, and is the center for discussion of challenging issues in LGBT+ data curation. Wikidata contains self-organized archives starting in the 1970s. In Munich the Forum Queeres Archiv, has been collecting, researching, and publishing for 20 years: Shelf meters full of bequests, files, books, journals, objects. The project Queer data [2] transforms analog queer history into linked open data and incorporates it into an open, freely accessible, public data infrastructure. The knowledge graph contains 7415 entities from the data sources of the Forum Queeres Archiv München, with more than 38,000 connections.

Discussion. Two challenges can be associated with knowledge graphs: (i) building them for representing knowledge contained in one or several data collections and connecting entities manually or automatically; (ii) exploring and operating on top of knowledge graphs to retrieve, discover and predict knowledge. The techniques and strategies used for both tasks depend on the context of the data, its characteristics and the underlying knowledge domain. In this context, regarding the Queer history in different knowledge domains (e.g., in the artistic production), aspects like the context in which the queer community has been an under-represented, almost invisible group should determine how a knowledge graph models it. Particular properties must be considered in a knowledge graph modelling task, like non-binary gender variables characterising entities. These variables are essential in determining how queer community members are connected to other members of the same community and those in other ones. The challenge to modelling Queer history is to use different properties to define identities that can be profoundly diverse and relations and provide a multi-perspective model. The union or correlation of these different perspectives can build a more representative queer history knowledge graph. Therefore the different viewpoints must be analysed and explored to identify communities, influence, and trends. The objective and main results of the work introduced by this paper are showing different perspectives of queer communities by building and integrating graphs.

3 BUILDING KNOWLEDGE GRAPHS PIPELINES

The main contribution of our work is the design of three pipelines to build knowledge graphs using different techniques. The purpose is to propose a pipeline that:

- Balances the coverage of the resulting knowledge graph that is its scope concerning the queer communities (i.e., are all potential queer individuals part of the graph, are their connections able to exhibit different communities).
- Reasonably consumes resources (CPU and memory) to explore, analyse and build dense graphs.

The second contribution is the implementation of these pipelines within a library [3] that allows testing them by calibrating specific parameters such as the number of iterations, the number of nodes to visit, etc. We also provide a ready-to-use notebook to run the experiments we have conducted (see Section 4).

3.1 Pipeline 1: Pure SPARQL based knowledge graph

Figure 1 illustrates the pipeline that builds a pure SPARQL-based knowledge graph. We propose the construction of two initial graphs by querying Wikidata to retrieve individuals identified as non-cisgender and non-heterosexual and their connections.

The Wikidata service offers the WikiData Query Service endpoint, which allows users to run SPARQL queries on the WikiData database. SPARQL allows creating a graph with the CONSTRUCT keyword, and the resulting graph can be downloaded as a .csv file with subject, predicate, and object columns. This .csv file can be transformed using the Pandas library into a DataFrame or a networkx Graph using the `networkx.from_pandas_dataframe` function. The interest in producing a `networkx` graph is that the library provides additional methods for graph analysis.

In the example below, a graph is constructed with all triples where the subject is a non-heterosexual or non-cisgender individual. This activity corresponds to the data retrieval phases of the pipeline by interacting with the Wikidata endpoint server (see Figure 1).

```
CONSTRUCT {
  ?person ?predicate ?object.
  }
    WHERE {
    {
        ?person wdt:P31 wd:Q5 . #?personId is a human
         ?person ?predicate ?object.
```

[1] https://www.wikidata.org/wiki/Wikidata:WikiProject_LGBT
[2] https://queerdata.forummuenchen.org/en/
[3] https://github.com/FloraHelmers/QueerHistoryProject

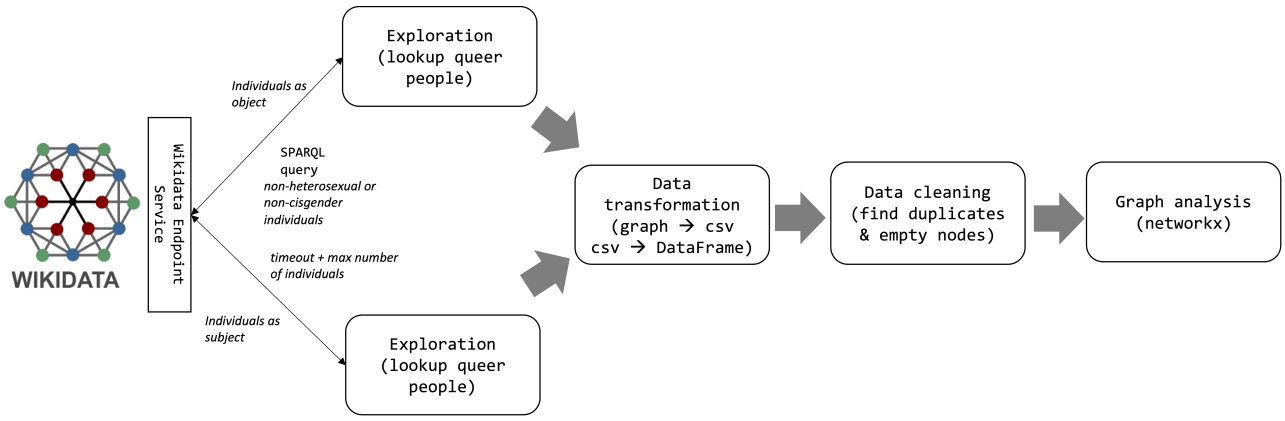

Figure 1: Pure SPARQL pipeline

```
        {
            ?person wdt:P21 ?sexorgender. #?person has ?sexorgender
            #?sexorgender is not male, female, cisgender male, cigender female, or cisgender person
            FILTER(?sexorgender NOT IN (wd:Q6581097, wd:Q6581072, wd:Q15145778, wd:Q15145779, wd:Q1093205)).
        } UNION {
            ?person wdt:P91 ?sexualorientation . #?person has ?sexualorientation
            FILTER(?sexualorientation != wd:Q1035954). #?sexualorientation is not heterosexual
        }
    }
    SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE]". }
}
```

We also create a second graph where the person is not the subject but the object. With these two graph views, we have all the connections to the elements. The exploitation is not necessarily easy because we only have the Internationalized Resource Identifier (IRI) of the selected elements, which may not be self-explanatory. We retrieve the labels of the entity using the service Wikibase. Retrieving the whole graph is costly, so we set a timeout. Note that this query evaluation doubles the time of the first query. So we have to put a limit on the size of the graph.

Observing the graphs, we see that we have to clean the data because they contain duplicates and "empty" nodes. Indeed, some elements have different IRI but have the same meaning: for instance, http://www.wikidata.org/prop/direct/P161 and http://www.wikidata.org/prop/statement/P161 gives access to the same property which is "cast member". However, since they have different terms, they are expressed as if they were different.

We get some empty nodes which do not have labels, such as entities Q19151093 [4] and Q19218452 [5]. Both have different IRI and Wikipedia Identifiers but have the same meaning because their properties are identical. Because we only get access to the elements located at a distance 1 from the people, we cannot correct these elements only using what we have gathered. Besides, we lose the added value of the knowledge graph, which is to automatically deduce information by using the varied distant relationships of the initial node. The principle of pipeline 1 is sound, but the execution cost is high according to the resources we have access to. Therefore we propose alternative pipelines described in the following sections.

3.2 Pipeline 2: Merging Stars

The construction pipeline of the knowledge graph about queer communities from Wikidata using merging stars can be summarised as follows: merge a list of graphs retrieved from Wikidata as `.nt` files that are akin to stars around a central entity node (see Figure 2). As shown in Figure 2, this pipeline starts with a SPARQL query to Wikidata that returns a list of Wikidata item IDs that are related to the queer community (see the corresponding query expression in Appendix A Figure 11).

This list, originally in JSON format, is then used to extract the relevant information necessary to query the graphs associated with each entity on Wikidata (IDs and URLs). We iteratively parse new information by using these graphs, basically creating a merged RDF graph using the RDF data of the nodes from the Wikidata entity URLs in an n-triples format (given in an .nt file format).

The merging of two knowledge graphs in .nt file format is a process in which the data from two separate RDF graphs is combined into a single graph. The process is straightforward, as both graphs are in a standard format that can be easily integrated. The only subtleties that arise when merging two RDF graphs are related to the processing of shared blank nodes.

Shared blank nodes are nodes that appear in both RDF graphs and are used to represent anonymous resources. These shared blank nodes must be merged in an integrated RDF graph to represent the same resource. The rules for merging shared blank nodes are defined in the W3C RDF 1.1 Semantics and Abstract Syntax specification [6]. The merging process must ensure that the integrated graph preserves the intended meaning of the original RDF graphs, including the relationships between the nodes in the graph. Therefore, the process must consider the context in which each shared blank node is used in both graphs and combine them to preserve the RDF data's intended meaning.

Our final integrated graph is then converted to a NetworkX multidigraph that proved to be the most helpful format for our analysis.

Once the graph has been created, it is necessary to prune it in various ways:

[4] http://www.wikidata.org/entity/Q19151093
[5] http://www.wikidata.org/entity/Q19218452

[6] https://www.w3.org/TR/rdf11-mt/#shared-blank-nodes-unions-and-merges

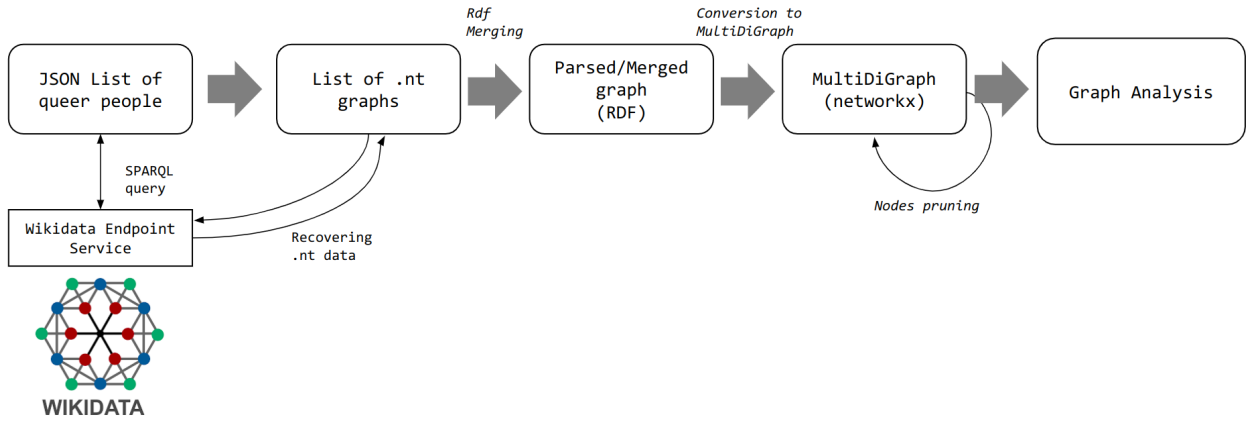

Figure 2: "Merging stars" pipeline

- Removing "dead-end" nodes (i.e., nodes that have an in-degree of less than 2) "Dead-end" nodes are often isolated and do not provide much information about the community.
- Removing duplicates (i.e., nodes that represent the same object literally) and isolated nodes (i.e., nodes that have both in-degree and out-degree of 0). These nodes are not connected to any other nodes in the graph and do not provide information about the community.

Depending on the application, one can also specify more complex pruning processes with rule-based deletion and deletion before each merge.

The pipeline is executed with two parameters: n the size of the list of people used as starting point and a `prune_policy` dictionary describing what process for removing the nodes should be used.

3.3 Pipeline 3: Crawler method

The crawler-based pipeline starts with a small number of nodes. It runs an iterative process to extract critical nodes representing properties of interest (potential common points) and use those to explore and discover more queer people and communities.

Figure 3 illustrates the main activities of the pipeline. The process begins by selecting a subset of nodes using a SPARQL query (see 1 in Figure 3). The query selects distinct people and their labels who have a specific property of interest and do not have other properties that would exclude them from being considered queer. A query result is a limited number of nodes, which is used as the starting point for the iterative process.

Next, the pipeline runs the PageRank algorithm on the initial set of nodes for ranking nodes based on the number and quality of incoming links (see 2 in Figure 3). The algorithm allows the identification of the most critical nodes in the graph, as they are more likely to represent properties of interest. The PageRank algorithm is run multiple times with different parameters, such as the damping factor (alpha) and the number of iterations (see 3 in Figure 3).

After the PageRank algorithm has been executed, the pipeline selects a certain number of the most critical nodes (k_prop) and uses them to explore further (see 4 in Figure 3). Specifically, it runs the same SPARQL query as before, using the selected nodes as the property of interest (see 5 in Figure 3). The result is a new set of nodes connected to the previously selected nodes through the property of interest. These new nodes are added to the original graph, and the process starts over again in n iterations (see 6 in Figure 3).

The crawler-based pipeline continues to run the PageRank algorithm and explore new nodes until a certain number of iterations have been completed (n_iter). The resulting graph is a composite of all the nodes and edges discovered during the process, and it is more densely connected and contains a higher number of queer people than the initial set of nodes.

3.4 Visualising graph structures

Visualising a knowledge graph is a crucial step in understanding the structure and content of the data. However, the standard visualisation tools available are not always enough to effectively display a knowledge graph's complex relationships and interconnections. To better understand the structure of the knowledge graph generated in our study, we implemented custom visualisation techniques, including different graph layouts and edge bundling. These visualisations are essential for showcasing the intricate relationships between the nodes in the graph and allow a more in-depth understanding of the data. The figures presented in this paper have been generated using these custom visualisation techniques, providing a clearer understanding of the graph structure and relationships.

Nodes Layout. The layout of the nodes can heavily influence the visual representation of the graph. We can randomly choose the position of each node, but the results may be a little messy. A popular layout is circular, where all the nodes form a circle (see Figure 4). A more sophisticated layout we mainly used here is the force-directed layout, described in [13], which minimises overlaps in the graph, evenly distributes nodes and links and organises items so that links are of a similar length with as few crossing edges as possible (see Figure 5). This process is done by assigning forces among the set of edges and the set of nodes based on their relative positions and then using these forces to simulate the edges and nodes' motion or minimise their energy. In the case of our pipelines based on .nt "stars" associated with entities,

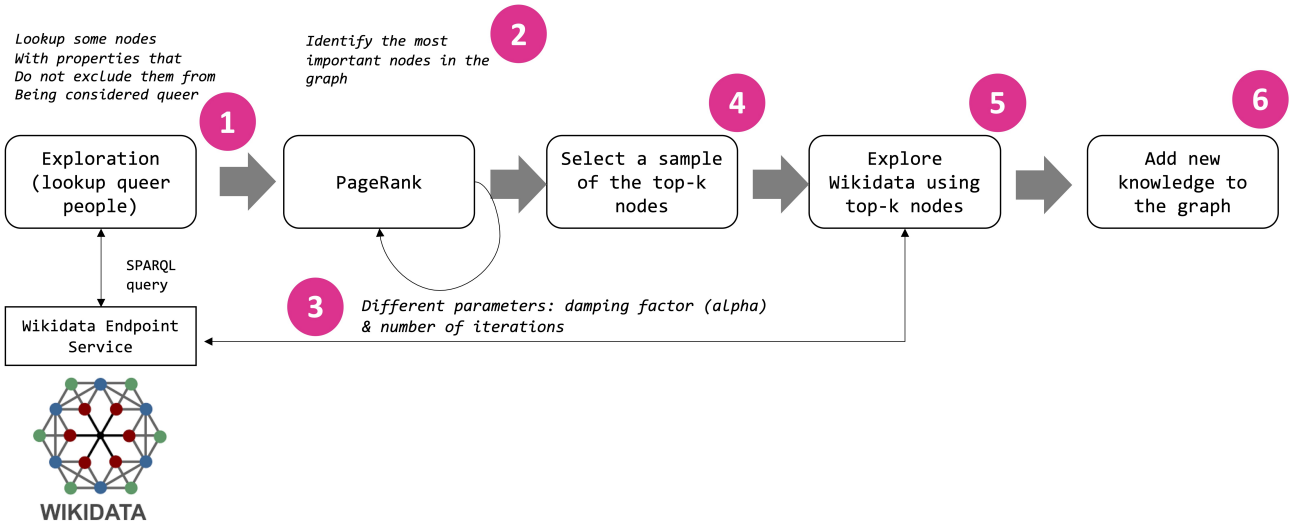

Figure 3: Crawled method pipelines

the force-directed layout is particularly relevant to distinguish the underlying structure obtained.

Edge-Bundling. Even with a sensible layout of the nodes, many edges make it difficult to establish a graph's relations and structure if we draw the edges as straight lines. We can thus use edge bundling, a method that allows edges to curve and then groups nearby ones together to help convey structure (see Figure 6). Rendering direct relations should be quick, even for large graphs, but bundling can be quite computationally intensive. Here we use the function "Datashader.hammer_bundle" (a variant of [11]) for the edge bundling.

4 EXPERIMENTS AND RESULTS

4.1 Implementation

We have developed a library for handling and building knowledge graphs with real-world data for a comfortable experimental setting. This library, written in Python, uses the NetworkX, RDFLib and Datashader libraries and allows us to evaluate the performance of different knowledge graph construction methods easily.

4.2 Experimental Setting

Queer datasets. In the conducted experiments, we use data from Wikidata, as this platform provides structured data that is well-suited for building knowledge graphs [7]). We used "real" data to make our simulation experiments realistic. For example, we plotted the histogram of the number of properties each node from the SPARQL query had and found that the distribution resembled a log-normal distribution. We fitted the data to a log-normal distribution and used the Kolmogorov-Smirnov test to evaluate the fitting. The test showed that the fitting was appropriate, which allowed us to use the log-normal distribution to model the distribution of the number of out edges in our simulation experiments. This strategy allowed us to have a more accurate simulation of the real-world construction of knowledge graphs and to assess the performance of different construction methods more accurately.

Because not all properties constitute common points among people, we only take a fraction of these number of out-edges to have our distribution for the models. By running tests on Wikidata, we found that approximately 1/50 of these out-edges created an entity-linking edge. This gives us the distribution with a probability density function (PDF):

$$\frac{1}{x\sigma\sqrt{2\pi}} \exp -\frac{(ln(x)-\mu)^2}{2\sigma^2} \tag{1}$$

with $\mu \approx 0.495$ and $\sigma \approx 5$.

Understanding the queer concepts. We defined an ontology to organise the critical elements of the Queer Community. We used preexisting standard ontologies such as foaf to describe the people and Geonames to make connections between spaces. Our ontology focuses on the European Queer scene of the 1980s. We added the correlation between gender and sex without having it explicitly said. We based our strategy on the idea of Katharina Brunner in the "Remove NA" project [8]. We used pyramidal relations to describe sexual orientations. Our ontology also focuses on art and collaborations, including the notion of avatars which is essential in the drag culture. The concept of influence is also present, considering elements such as songs played in clubs, characters in films or books, and access to art pieces. This version focuses on the representation of LGBTQ+ people in the culture of the 1980s. Still, it could benefit from exploring the rights of LGBTQ+ individuals from the past from a more global and non-Eurocentric perspective. One possible approach would be to examine the various terms used to define gender across different cultures, such as the third gender in the Bugis society, and to include references to the laws and changes in each country. The first we can get using the Wiki Data Query Service.

Experiments objective. With the experiments we conducted for this study, we aimed to directly compare the performance of the three pipelines in terms of time efficiency, size, structure, and content. To accomplish this, we measured the time each pipeline takes to create the knowledge graph, the number of nodes in the graph and the number of edges in each built graph. Furthermore,

[7] https://ai.stanford.edu/blog/introduction-to-knowledge-graphs/

[8] https://queerdata.forummuenchen.org/en/

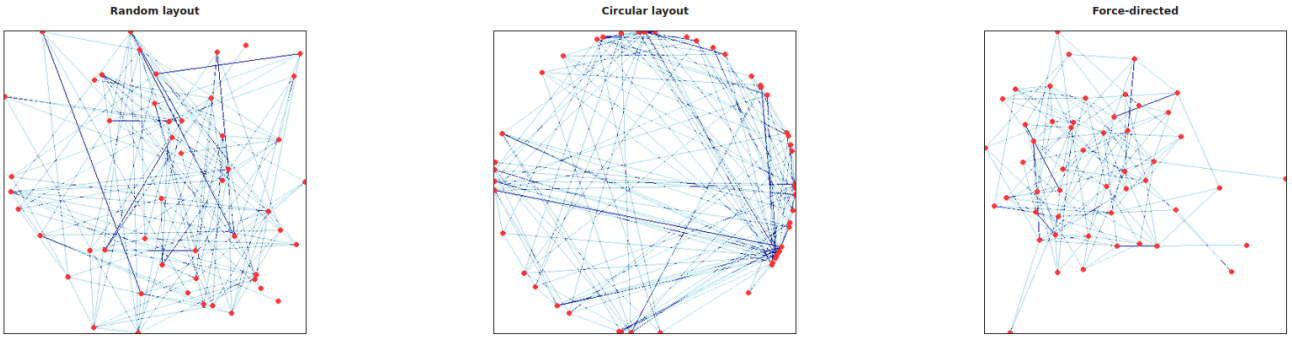

Figure 4: Nodes layout: random, circular, force-directed

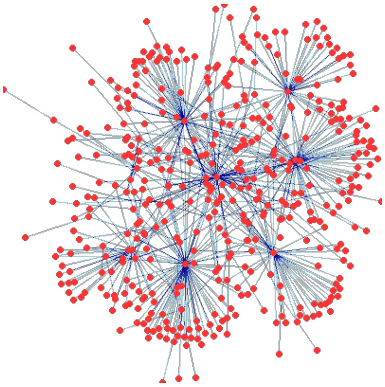

Figure 5: A force-directed layout

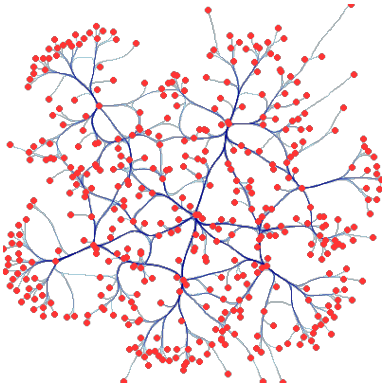

Figure 6: Edge-bundling graph

we analysed the maximal graphs that each method/pipeline can create and compare them.

It is important to note that the experiments were performed under the constraints of the regular account on a free service, meaning that the results may differ if performed under different conditions. However, this experimental setting was chosen to simulate a realistic scenario for a typical user.

As such, we used the Google Colab environment to run our experiments with an allocated virtual machine with CPU (1xsingle core hyperthreaded Xeon Processors @2.3Ghz (i.e., 1 core, 2 threads); 12,7 GB of RAM (with 0.8 GB are already taken); a storage space of 108 GB, of which only 77 GB is available to the user.

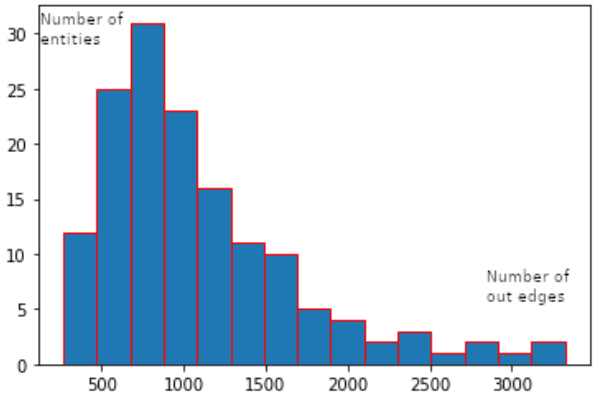

Figure 7: Histogram of the number of out-edges from the .nt files

4.3 Structure and communities.

In our experiments, we found that the structure of the created knowledge graphs varied depending on the construction method used and the distribution used. Some methods resulted in graphs with a high degree of connectivity and a dense network of edges, like the Pure SPARQL pipeline, while others produced sparser graphs with fewer connections between nodes. We also observed that some methods tended to create graphs with a more hierarchical structure, especially in trials with the crawler-based pipeline. In contrast, others generated graphs with a more flat structure.

To better understand the structure of the graphs produced through our pipelines, we used community detection algorithms to identify and analyse the communities within the graphs. In particular, we used the modularity maximisation algorithm, and the Girvan-Newman algorithm [9] to identify communities based on the density of connections within communities and the sparsity of relations between communities. Considering the size of the graphs, the community algorithms were run on either small portions of the graphs or simplified versions where we linked entities according to their properties in common and then removed the redundant nodes. We could then create several dendrograms to be interpreted and used for statistical analysis. Crucially, we found that the identified communities often corresponded to semantically meaningful groups of nodes, such as nodes representing people from the same country or nodes representing concepts in the same domain.

Overall, our experiments suggest that the structure of the created knowledge graphs is influenced by the construction

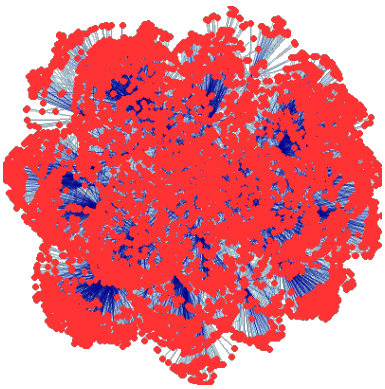

Figure 8: Graph generated by Pure SPARQL, after removing dead ends

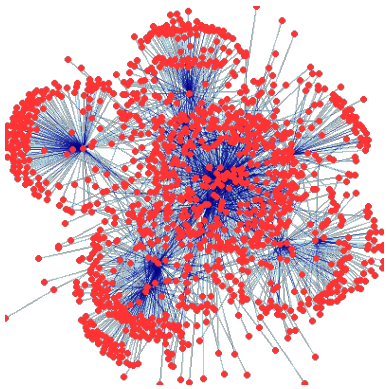

Figure 9: Graph generated by Merging stars

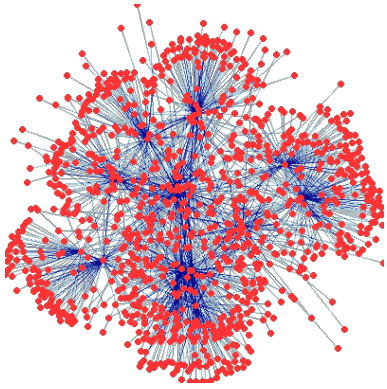

Figure 10: Graph generated by the crawler method pipeline

method used and that different methods may be more suitable for various applications depending on the desired configuration of the resulting graph, with the crawler-based method offering the most freedom in choosing the behaviour of the exploration and allowing to obtain the most salient communities. For example, the behaviour of exploration through common points allowed us to naturally discover key communities, e.g. `[Q160456, Q239533, Q461657, Q2041541, Q3611679, Q6312325, Q104358, Q187814, Q346309, Q93349, Q6897510, Q463319, Q267951, Q2849427, Q7933271, Q259507, Q461758, Q3942181]` corresponding to a group of Black American queer Women. Additionally, it suggests that community detection algorithms can provide valuable insights into the created graphs' structure and organisation and help identify semantically meaningful groups of nodes within the graph.

4.4 Results

The initial "raw" knowledge graph without edges built from Wikidata consists of 8171 references to queer people and 4459 predicates and 32651 objects. The graph with "in" edges, where the LGBTQ+ people are the object of the RDF triples, contains 7281 queer people, that is, 7281 nodes with "in" edges. We get 521 different predicates and 168 923 subjects. In the predicates, when comparing the IRI, there are interesting distinctions.

Structural comparison of the resulting graphs. Visual analysis on dense graphs with an actual number of nodes can be laborious, hence the need for some graphical tools. We observe structural differences on the graph generated, with examples given below in the figures 8, 9 and 10. As stated in Section 3, the pure SPARQL pipeline (pipeline 1) had to include data-cleaning activities to understand the queer communities better. Still, as shown in Figure 8 the raw graph built from exploring Wikidata with SPARQL queries without dead ends is very dense.

Pipeline 2 adopting the merging stars leads to a clean and representative view of the queer communities (in an integrated vision). As described in Section 3 in the merging stars pipeline, we defined a raw graph and then pruned it to eliminate "dead nodes", then duplicates and isolated nodes and finally, a wholly cleaned knowledge graph, thoroughly pruned, as shown in Figure 9.

The graph produced through the crawler method pipeline exploits the notion of centrality to identify the nodes representing influential queer people and then explore their connections to determine whether they belong not to a queer community. The resulting graph is shown in Figure 10. The graph exhibits highly connected nodes and communities of nodes agglutinated around.

Performance evaluation of the pipelines. To evaluate the performance of the three pipelines, we compare the time taken, the number of nodes, and the number of edges in the knowledge graph created by each pipeline for a fixed number of people ($n = 30$). Hereafter, Pipelines 1, 2 and 3 represent, respectively, the Pure SPARQL, Merging stars and Crawler pipelines and Table 1 shows the results.

Pipeline	Time	Nodes	Edges
Pipeline 1	1223 ms	6659	7217
Pipeline 2	36.3 s	8956	19172
Pipeline 3	486 s	10321	20112

Table 1: Comparison of the variables in the knowledge graph created by each pipeline for $n = 50$.

Additionally, we compare the maximal knowledge graph that each pipeline can create, where we do not impose any limit and try to get as many people as possible included. Table 2 shows the results.

The results of the comparison between the three pipelines show that Pipeline 1 is the fastest in creating a knowledge graph. Still, Pipeline 2 has more freedom in choosing which nodes are kept and, as such, could be sped up (for example, by pruning

Pipeline	Size	Time (s)
Pipeline 1	887 048 triples	38 294 ms
Pipeline 2	1 038 197 triples	52 min
Pipeline 3	>300 000 triples	NA

Table 2: Comparison of the maximal knowledge graph that each pipeline can create.

more useless nodes before each merging) while maintaining good quality. In the same way, Pipeline 3 offers maximal freedom in creating the knowledge graph but is, at the same time, the slowest of the three.

These results suggest that while time efficiency is critical, having more precise processing of the nodes in the knowledge graph is also crucial for providing a comprehensive representation of the information that can be visualised and used more efficiently after. Pipeline 3 may be the most suitable for applications requiring a directed and precise information model. In contrast, Pipeline 1 may be ideal for applications with real-time constraints and frequent data updates.

Overall, the results highlight the trade-off between time efficiency and the comprehensiveness of the knowledge graph and suggest that the choice of a pipeline may depend on the specific requirements and constraints of the application.

4.5 Interpretation and storytelling

Queer is often used as an umbrella term to denote sexual identity within a particular community. A queer community may consist of people who identify as lesbian, gay, bisexual, transgender, and so on, and some find queer an easy way to describe such a large community. Labelling people whose sexual identities fall outside of heterosexuality may create solidarity among people based on commonality, which may, in turn, encourage them to identify with one another and build a community in which they find support and organise to initiate a political movement [9].

The concept of "queer community" is a broad term with several definitions. Accordingly, this can result in creating graphs with various densities and branching. Indeed, a queer community may be made up of people who identify as lesbian, gay, bisexual, or transgender, having several groups with dense connections or by adding gender communities beside sexual communities, we get more sparse graphs.

In this context, storytelling is a process of extracting information about the people of LGBTQ+ and creating connections which have not been created or were ignored.

This process can come from a lot of various elements given by the statistical analysis of the graphs obtained; even data encountered when building our pipelines brought interesting perspectives on how the existing state of the data, with the aforementioned log-normal distribution indicating a wealth of data for a selected set of individuals in the long-tail but also a vast underacknowledged majority.

5 CONCLUSIONS AND FUTURE WORK

Through knowledge graphs, this paper introduced our experimental approach to building queer communities' history from different perspectives. Instead of making one unique general knowledge graph, we decided to explore and identify several views from an existing knowledge graph (i.e., Wikidata) to discover new perspectives on queer communities. We propose and exhibit the pipelines used for building the knowledge graphs, including the sets of SPARQL queries used for exploring Wikidata.

We have developed and evaluated efficient methods/models for constructing knowledge graphs. Our simulation experiments have shown that we can use these models to choose the method, reduce the time it takes to build the graph, and improve the quality of the resulting graph. These findings have important implications for the construction and use of knowledge graphs in various applications.

The advantage of materialising the pipelines is that in our future work, we will enhance them, estimating their execution cost and the resources required for building them (data size, data exchanged along the activities, intermediate results size). We will use this meta-data to decide the target architecture necessary for executing the pipelines. We have also shown how knowledge graphs providing different views of the queer community can be explored, combined and analysed to answer specific research questions. There are two future directions regarding these tasks, first is deciding which are the most suitable graph operations and analytics algorithms for answering particular research questions. The second is interpreting results; therefore, we will be willing to apply storytelling techniques to produce dashboards that provide plots and multimedia content to propose an interpretation of the represented knowledge.

[9] https://www.britannica.com/topic/queer-sexual-politics

REFERENCES

[1] Christian Bizer, Christian Becker, Pablo N Mendes, Robert Isele, Andrea Matteini, and Andreas Schultz. 2012. Ldif-a framework for large-scale linked data integration. (2012).

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[3] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*.

[4] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 601–610.

[5] Oren Etzioni, Robert E Bart, Michael D Schmitz, Stephen G Doderland, et al. 2014. Open language learning for information extraction. US Patent App. 14/083,261.

[6] Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 conference on empirical methods in natural language processing*. 1535–1545.

[7] James Fan, David Ferrucci, David Gondek, and Aditya Kalyanpur. 2010. Prismatic: Inducing knowledge from a large scale lexicalized relation resource. In *Proceedings of the NAACL HLT 2010 first international workshop on formalisms and methodology for learning by reading*. Association for Computational Linguistics, 122–127.

[8] Gleb Gawriljuk, Andreas Harth, Craig A Knoblock, and Pedro Szekely. 2016. A scalable approach to incrementally building knowledge graphs. In *International conference on theory and practice of digital libraries*. Springer, 188–199.

[9] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.

[10] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial intelligence* 194 (2013), 28–61.

[11] Christophe Hurter, Ozan Ersoy, and Alexandru Telea. 2012. Graph bundling by kernel density estimation. In *Computer graphics forum*, Vol. 31. Wiley Online Library, 865–874.

[12] Mayank Kejriwal, Craig A Knoblock, and Pedro Szekely. 2021. *Knowledge graphs: Fundamentals, techniques, and applications*. MIT Press.

[13] Stephen G. Kobourov. 2012. Spring Embedders and Force Directed Graph Drawing Algorithms. *CoRR* abs/1201.3011 (2012). arXiv:1201.3011 http://arxiv.org/abs/1201.3011

[14] Alexander G Ororbia II, Jian Wu, and C Lee Giles. [n.d.]. CiteSeerX: Intelligent Information Extraction and Knowledge Creation from Web-Based Data. ([n. d.]).

[15] Jay Pujara and Lise Getoor. 2014. Building dynamic knowledge graphs. In *NIPS Workshop on Automated Knowledge Base Construction*, Vol. 9.
[16] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. 2013. Knowledge graph identification. In *International semantic web conference*. Springer, 542–557.
[17] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. 697–706.
[18] Pedro Szekely, Craig A Knoblock, Jason Slepicka, Andrew Philpot, Amandeep Singh, Chengye Yin, Dipsy Kapoor, Prem Natarajan, Daniel Marcu, Kevin Knight, et al. 2015. Building and using a knowledge graph to combat human trafficking. In *International Semantic Web Conference*. Springer, 205–221.
[19] Genoveva Vargas-Solar, Md Sahil Hassan, and Ali Akoglu. 2022. JITA4DS: Disaggregated Execution of Data Science Pipelines Between the Edge and the Data Centre. *J. Web Eng.* 21, 1 (2022). https://doi.org/10.13052/jwe1540-9589.2111

A APPENDIX

SPARQL query expressions for extracting data from Wikidata for building knowledge graphs. Retrieving nodes related to the queer community for applying merging stars technique (see Figure 11).

```
SELECT DISTINCT ?person
WHERE {
  {
    ?person wdt:P31 wd:Q5 . #?personId is a human

    {
      ?person wdt:P21 ?sexorgender. #?person has ?sexorgender
      #?sexorgender is not male, female, cisgender male, cigender female, or cisgender person
      FILTER(?sexorgender NOT IN (wd:Q6581097, wd:Q6581072, wd:Q15145778, wd:Q15145779, wd:Q1093205)).
    } UNION {
      ?person wdt:P91 ?sexualorientation . #?person has ?sexualorientation
      FILTER(?sexualorientation != wd:Q1035954). #?sexualorientation is not heterosexual
    }
  }
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE]". }

} LIMIT
```

Figure 11: SPARQL query for retrieving the IDs of Wikidata nodes related to the queer community

Zoomed in view of the crawler graph. Do you wonder what the data in Figure 10 really refers to ? We chose some nodes around "Jean Cocteau", the second node of highest degree with its 409 neighbours, in order to present you the various wikidata entities we are dealing with. On this image, the colour of the nodes are mapped to the degrees, with a logarithmic scale. The node "human", in red, has 32 neighbours; and the node "Nobel Prize in Litterature", in dark blue, has 5 neighbours.

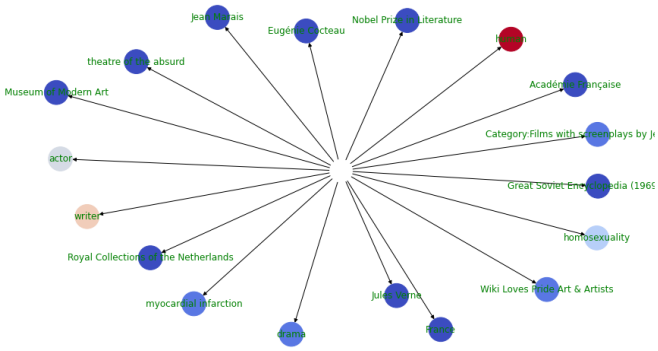

Figure 12: Extract of the graph generated by the crawler method pipeline, centred around Jean Cocteau