Exploring Graph-querying approaches in LifeGraph

Luca Rossetto
Department of Informatics
University of Zurich
rossetto@ifi.uzh.ch

Matthias Baumgartner
Department of Informatics
University of Zurich
baumgartner@ifi.uzh.ch

Ralph Gasser
Department of Mathematics and Computer Science
University of Basel
ralph.gasser@unibas.ch

Lucien Heitz
Department of Informatics
Digital Society Initiative
University of Zurich
heitz@ifi.uzh.ch

Ruijie Wang
Department of Informatics
University Research Priority Program
"Dynamics of Healthy Aging"
University of Zurich
ruijie@ifi.uzh.ch

Abraham Bernstein
Department of Informatics
University of Zurich
bernstein@ifi.uzh.ch

ABSTRACT

The multi-modal and interrelated nature of lifelog data makes it well suited for graph-based representations. In this paper, we present the second iteration of LifeGraph, a Knowledge Graph for Lifelog Data, initially introduced during the 3rd Lifelog Search Challenge in 2020. This second iteration incorporates several lessons learned from the previous version. While the actual graph has undergone only small changes, the mechanisms by which it is traversed during querying as well as the underlying storage system which performs the traversal have been changed. The means for query formulation have also been slightly extended in capability and made more efficient and intuitive. All these changes have the aim of improving result quality and reducing query time.

CCS CONCEPTS

• **Information systems** → *Ontologies*; *Users and interactive retrieval*; *Specialized information retrieval*; Content analysis and feature selection.

KEYWORDS

Lifelogging, Lifelog Search Challenge, Knowledge Graphs, Graph-based Retrieval, Multi-modal Retrieval

ACM Reference Format:
Luca Rossetto, Matthias Baumgartner, Ralph Gasser, Lucien Heitz, Ruijie Wang, and Abraham Bernstein. 2021. Exploring Graph-querying approaches in LifeGraph. In *Proceedings of the 4th Annual Lifelog Search Challenge (LSC '21), August 21–24, 2021, Taipei, Taiwan.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3463948.3469068

1 INTRODUCTION

Any representation that aims at capturing the relevant aspects of a human's daily experiences with any degree of accuracy, which is the ultimate aim of the *lifelogging* process, will necessarily be diverse, multi-modal, highly interconnected and self-referencing. A typical lifelog consists of recordings made by multiple physical, as well as virtual sensors, including wearable cameras, bio-feedback devices, location trackers, and activity tracking software. All of these mechanisms capture different, sometimes overlapping aspects of a lifelogger's daily experience, which makes their resulting recordings strongly interrelated. It is therefore reasonable to represent such lifelog data as a graph structure, for a graph is not only able to represent a multitude of *log entries* and their relations to each other, but also allows for putting them into a larger context relative to general and non-personal information about the world.

In this paper, we present an improved version of *LifeGraph*, a Knowledge Graph for Lifelog Data, which was first introduced [13, 14] at the 3rd Lifelog Search Challenge [8] in 2020. The idea behind LifeGraph is to combine individual lifelog entries and their direct annotations with larger external knowledge bases in order to enrich the context of the log and enable semantically more complex queries, using concepts for which no direct annotations exist. While in the previous iteration, the focus was primarily on the construction of the LifeGraph, in this iteration of the challenge, we focus primarily on how the graph can be queried more effectively. The retrieval system to be evaluated will use largely the same graph with only minor modifications, which is possible do to the reuse of the previous challenge dataset with only minor modification [9].

The following sections will discuss the previous approach as well as some relevant limitations in Section 2, before introducing the the changes made in this second iteration of LifeGraph in Section 3. Section 4 then closes with some concluding remarks.

2 LIMITATIONS OF THE FIRST LIFEGRAPH SYSTEM

The first version of the LifeGraph system consisted of three primary system components: 1) a triple store that held the persistent representation of the graph, 2) a query engine that would translate from user input to a graph-queries, and back and 3) a user interface that could be used to specify queries and browse the retrieved results.

For the triple store, we used BlazeGraph,[1] which is an open-source graph database system with a SPARQL [12] query interface. The query engine was a purpose-built application whereas the user interface consisted of a modified version of vitrivr-ng [6], which is the browser-based user interface component used in the open-source multimedia retrieval stack *vitrivr* [18]. Additionally, we used an nginx[2] instance to serve the UI as well as the static media components.

The adoption of vitrivr-ng for the user interface had some clear benefits, among them readily available mechanisms for tag-based query construction and efficient result browsing, as well as it's capabilities for communication with the evaluation system used during the competition [17]. The built-in late filtering mechanism was, however, only useful in some situations, since hard Boolean filtering would sometimes remove too few results in order for it to be useful or too many results, possibly including the query targets. This was due to both too coarse-grained filter options as well as imprecise query tasks, sometimes containing erroneous details, such as queries miss-stating the month or year when the described event took place.

The choice of using a SPARQL-based graph database as a persistent storage layer, while sounding reasonable initially, turned out to be sub-optimal. This was primarily due to the mismatch between our used notion of semantic similarity based on graph paths between a concept and a log entry and the lack of an analogous notion in the SPARQL language. While SPARQL knows the concept of a *property path*,[3] which enables the querying for entities connected by a path with specific properties, only the start- and end-points of these paths are returned as results and the actual paths themselves are discarded. While this information is sufficient for many use cases, it is not in our case, since the similarity score of a query and any result retrieved by the system is inversely dependent on the length of the path through the graph connecting the two nodes. This limitation made it necessary to perform the graph exploration in multiple queries, which would allow for increasingly complicated paths and estimation of the path length between two graph nodes based on which query returned them first. Using such queries introduced a substantial amount of redundancy, both on the query execution as well as on the result post-processing stage, which would lead to an unnecessarily long query execution time. We intend to remedy this by replacing the underlying database by a system which offers functionality more suited to our querying strategies.

3 IMPROVEMENTS

The following provides an overview of the improvements made to the LifeGraph system, based on insights gained from both the previous Lifelog Search Challenge as well as related activities in the context of the Video Browser Showdown [16, 19], for which we investigated the use of Knowledge Graph structures for video retrieval [15].

[1]https://blazegraph.com/
[2]https://nginx.org/
[3]https://www.w3.org/TR/sparql11-query/#propertypaths

3.1 Cottontail DB

Cottontail DB [5] is a database management system for multimedia retrieval and analysis and replaces BlazeGraph as data management layer. It natively supports the efficient evaluation of vector-space and full-text queries as well as traditional relational queries and is also used as the persistent storage layer in the vitrivr multimedia retrieval stack. While not a graph-database as such, Cottontail DB offers many relevant functionalities that we can leverage for graph exploration. Initial tests have shown that searching for graph paths between a set of initial query concepts and a relevant number of log entries can be achieved using breadth-first-search on top of Cottontail DB while still maintaining interactive response times. This rather simple approach has the added benefit of building up the graph paths iteratively, which means that their exact properties are known and no estimation is required (in contrast to the previous method). Additionally, Cottontail DB's native support for vector-space nearest-neighbor queries enables us to use other notions of semantic similarity during query evaluation, such as the distance in a graph-embedding space. The latter possibility is further discussed in Section 3.4.

Technically, Cottontail DB is a column-store with a type system that supports both scalar values as well as vectors as first-class citizens. The query language allows for formulation of both Boolean search as well as retrieval using nearest neighbor search and the vector space model with various distance metrics, as well as a combination of both. This is complemented with various index structures that can optionally be used to speed-up both types of queries, sometimes by sacrificing retrieval accuracy. Both the set of supported indexes as well as the rule-based query planning and execution engine of Cottontail DB can be easily extended and adapted to offer optimal support for the application at hand.

3.2 Graph pruning and additions

Our first version of LifeGraph was built around COEL (Classification of Everyday Living) [2] through which we connect high-level concepts around everyday activities with low-level annotations or detectable objects in images. Participation in the LSC 2020 has shown that we rarely used search terms provided by COEL, and that COEL played an insignificant role in the query expansion. Following these insights, we remove COEL entities from the graph and instead rely on Wikidata's[4] internal class hierarchy. For this, we expand the initial seed entities (detected objects) along a set of manually defined predicates, such as $:instanceOf$ or $:subClassOf$. This reduces the number of concepts in our graph, and provides a clearer path structure between search keywords and log entries. In turn, we include instances of detectable concepts to increase the breadth of the graph. To avoid noise in the graph, we remove all entities that do not meet a certain degree threshold.

3.3 Query formulation

As in the previous version, a query is composed of one or multiple graph nodes, starting from which a traversal is performed until a sufficient number of related results is found. These initial nodes are selected using a simple auto-completing text box. In this version, we make several minor improvements to this process. For the text

[4]https://www.wikidata.org/wiki/Wikidata:Main_Page

auto-complete, we include synonyms and alternative labels for concepts, in order to make them more easily selectable. We also use information about the inherent expected relevance of concepts, such as their relative occurrence or their shortest path to any log entry, to sort the list of suggestions over a previously-used simple sub-string matching, in order to increase the likelihood that the relevant concepts appear higher in the list. Further possibilities to improve the query item selection mechanism are outlined in Section 3.5.

Once concepts are selected, we make use of functionality that was recently added to *vitrivr-ng* and allows a user to add a notion of *priority* to every specified concept. This enables the distinction between concepts that *should* be part of the result set from those that are *required*, or those that are supposed to be *excluded* from the results. We expect especially the latter to be of use, increasing the precision of a query and thereby reduce noise within the retrieved results.

Additionally, we add functionality to query by time and date information, in order to have a fallback mechanism for queries which contain no semantic information that can be directly related to any concept present in the graph. The previous version of the system allowed only for late-filtering based on properties such as the year, month, day or weekday of a log entry, but did not offer any possibility to simply select all entries from a given time frame in order to browse them.

3.4 Graph exploration methods and distance measures

Standard query languages are unmatched in terms of retrieving precise information from Knowledge Graphs, since the evaluation of standard queries is based on rigorous logical inferences. However, this strength could have negative consequences in the case of lifelog search. Firstly, lifelog search systems are developed for common users browsing daily life logs with vague memory fragments in mind. We should not expect that they search the graph using precise concepts (e.g., use more common terms like "car" instead of the Wikidata entry that is "motor car"). Furthermore, the words used in a query can sometimes include two or more concepts that from the viewpoint of a Knowledge Graph are not at all closely related (e.g., "breakfast" and "sunset"), which would inevitably cause a query evaluation failure. Secondly, as introduced in Section 3.3, users can write search queries composed of multiple concepts with the help of auto-completing and sorting functionalities. From a user's perspective, more input concepts should provide a broader search space for the system, and the system should accordingly retrieve more results. However, from the perspective of query evaluation, more concepts mean more constraints, leading to fewer results. Indeed, this issue can be addressed by query operators (i.e., OR) and heuristic rules (e.g., combinations of input concepts). But these solutions lack flexibility, require a lot of manual effort, and can be time-consuming during the query evaluation. Thirdly, as an automatically constructed Knowledge Graph, LifeGraph is not complete and may contain errors. Any piece of missing or erroneous information in LifeGraph can prevent the successful evaluation of a query, even when most constraints can be satisfied.

Due to the above-mentioned issues, we need a graph exploration method which 1) accepts imprecise and even self-contradictory inputs, 2) can efficiently handle a group of input concepts, and 3) is robust against missing and/or erroneous information in LifeGraph. In the literature [10], a similar scenario is to perform approximate querying over incomplete Knowledge Graphs. Distance measures in the vector spaces of Knowledge Graphs have been adapted as a better alternative to standard queries. Recent works [4, 10] have demonstrated the usability and strength of Knowledge Graph embedding [1, 11].

Knowledge Graph embedding models compute a low-dimensional numerical representation (embedding) of each entity in the Knowledge Graph. Their core idea is that related entities should have similar embeddings, i.e., the embedding space preserves the structural properties of the Knowledge Graph [11]. In our scenario, we can use the distance between the embeddings of two entities as an alternative to the hop-based distance. This has the potential to not only reduce query time for query nodes which have only relatively long connecting paths to log entries, but also to partially compensate for missing links in the graph.

3.5 Recommender system additions

Recommender systems (RS) are information filtering systems that allow us to assess and predict the relevance of a given item for a specific user [3]. We want to explore the possibility of deploying a RS on LifeGraph, for the purpose of enriching query formulations and search results. For RS, it is customary to calculate item recommendations for user on the basis of 1) shared properties between users and items (content-based filtering), 2) user-article pairs that express a certain relationship (collaborative filtering) or 3) a combination of both aspects (hybrid models) [7]. Lifelogs present a challenge for deploying a RS, since they fall outside the traditional scope of the previously mentioned approaches; the item side that is constituted of lifelog entries lacks a matching user side. We see two possible ways of how to circumvent this limitations.

First, a RS can be used for ranking concept labels in terms of their relevance for creating a suggestion list used during query formulation. For this purpose, user search queries are matched against lifelog entry descriptions or Knowledge Graph entities. Using complimentary data such as co-occurrence counts of search keywords or query tags, together with term frequencies of entry descriptions, a RS is able to calculate similarity scores between search queries on the one side and lifelog entry descriptions on the other.

Second, embeddings and path distances can be used in combination with a RS for the purpose of calculating similarity scores between lifelog entries. One way of implementing this is to feed preliminary, distance-based search results into the RS, with the goal of getting an enhanced results set that included additional lifelog entries. The similarity between entries would be assessed on the basis of a similarity between the respective lifelog entry descriptions of the preliminary results and all existing lifelog entries.

4 CONCLUSIONS

In this paper, we have introduced a second iteration of LifeGraph, a Knowledge Graph for lifelog data. Based on insights gained from the participation to the 2020 Lifelog Search Challenge with the first LifeGraph prototype, we propose several changes in order to improve both query specification capabilities and result quality, simultaneously reducing query processing time. While the application of Knowledge Graph-based methods to the organization and retrieval of lifelogs is not extensively studied so far, we still see it as a promising avenue which warrants further study.

ACKNOWLEDGMENTS

This work was partially funded by the Digital Society Initiative of the University of Zurich, the University Research Priority Program "Dynamics of Healthy Aging" at the University of Zurich, and the Swiss Re Institute.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 2787–2795. https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html

[2] Paul Bruton, Joss Langford, Matthew Reed, and David Snelling. 2019. Classification of Everyday Living Version 1.0. https://docs.oasis-open.org/coel/COEL/v1.0/os/COEL-v1.0-os.html Last updated 23 January 2019.

[3] C Aggarwal Charu. 2016. *Recommender Systems: The Textbook.*

[4] Daniel Daza and Michael Cochez. 2020. Message Passing for Query Answering over Knowledge Graphs. *CoRR* abs/2002.02406 (2020). arXiv:2002.02406 https://arxiv.org/abs/2002.02406

[5] Ralph Gasser, Luca Rossetto, Silvan Heller, and Heiko Schuldt. 2020. Cottontail DB: An Open Source Database System for Multimedia Retrieval and Analysis. In *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*, Chang Wen Chen, Rita Cucchiara, Xian-Sheng Hua, Guo-Jun Qi, Elisa Ricci, Zhengyou Zhang, and Roger Zimmermann (Eds.). ACM, 4465–4468. https://doi.org/10.1145/3394171.3414538

[6] Ralph Gasser, Luca Rossetto, and Heiko Schuldt. 2019. Towards an All-Purpose Content-Based Multimedia Information Retrieval System. *CoRR* abs/1902.03878 (2019). arXiv:1902.03878 http://arxiv.org/abs/1902.03878

[7] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[8] Cathal Gurrin, Tu-Khiem Le, Van-Tu Ninh, Duc-Tien Dang-Nguyen, Björn Þór Jónsson, Jakub Lokoc, Wolfgang Hürst, Minh-Triet Tran, and Klaus Schöffmann. 2020. Introduction to the Third Annual Lifelog Search Challenge (LSC'20). In *Proceedings of the 2020 on International Conference on Multimedia Retrieval, ICMR 2020, Dublin, Ireland, June 8-11, 2020*, Cathal Gurrin, Björn Þór Jónsson, Noriko Kando, Klaus Schöffmann, Yi-Ping Phoebe Chen, and Noel E. O'Connor (Eds.). ACM, 584–585. https://doi.org/10.1145/3372278.3388043

[9] Cathal Gurrin, Björn Þór Jónsson, Klaus Schöffmann, Duc-Tien Dang-Nguyen, Jakub Lokoč, Minh-Triet Tran, Wolfgang Hürst, Luca Rossetto, and Graham Healy. 2021. Introduction to the Fourth Annual Lifelog Search Challenge, LSC'21. In *Proc. International Conference on Multimedia Retrieval (ICMR'21)*. ACM, Taipei, Taiwan.

[10] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding Logical Queries on Knowledge Graphs. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 2030–2041. https://proceedings.neurips.cc/paper/2018/hash/ef50c335cca9f340bde656363ebd02fd-Abstract.html

[11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *ICML*. Omnipress, 809–816.

[12] Eric Prud'hommeaux and Andy Seaborne. 2008. *SPARQL Query Language for RDF*. W3C Recommendation. W3C. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[13] Luca Rossetto, Matthias Baumgartner, Narges Ashena, Florian Ruosch, Romana Pernischová, and Abraham Bernstein. 2020. A Knowledge Graph-based System for Retrieval of Lifelog Data. In *Proceedings of the ISWC 2020 Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 19th International Semantic Web Conference (ISWC 2020), Globally online, November 1-6, 2020 (UTC) (CEUR Workshop Proceedings, Vol. 2721)*, Kerry L. Taylor, Rafael S. Gonçalves, Freddy Lécué, and Jun Yan (Eds.). CEUR-WS.org, 223–228. http://ceur-ws.org/Vol-2721/paper557.pdf

[14] Luca Rossetto, Matthias Baumgartner, Narges Ashena, Florian Ruosch, Romana Pernischová, and Abraham Bernstein. 2020. LifeGraph: A Knowledge Graph for Lifelogs. In *Proceedings of the Third ACM Workshop on Lifelog Search Challenge, LSC@ICMR 2020, Dublin, Ireland, June 8-11, 2020*, Cathal Gurrin, Klaus Schöffmann, Björn Þór Jónsson, Duc-Tien Dang-Nguyen, Jakub Lokoc, Minh-Triet Tran, and Wolfgang Hürst (Eds.). ACM, 13–17. https://doi.org/10.1145/3379172.3391717

[15] Luca Rossetto, Matthias Baumgartner, Narges Ashena, Florian Ruosch, Romana Pernischová, Lucien Heitz, and Abraham Bernstein. 2021. VideoGraph - Towards Using Knowledge Graphs for Interactive Video Retrieval. In *MultiMedia Modeling - 27th International Conference, MMM 2021, Prague, Czech Republic, June 22-24, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12573)*, Jakub Lokoc, Tomás Skopal, Klaus Schoeffmann, Vasileios Mezaris, Xirong Li, Stefanos Vrochidis, and Ioannis Patras (Eds.). Springer, 417–422. https://doi.org/10.1007/978-3-030-67835-7_38

[16] Luca Rossetto, Ralph Gasser, Jakub Lokoc, Werner Bailer, Klaus Schoeffmann, Bernd Münzer, Tomás Soucek, Phuong Anh Nguyen, Paolo Bolettieri, Andreas Leibetseder, and Stefanos Vrochidis. 2021. Interactive Video Retrieval in the Age of Deep Learning - Detailed Evaluation of VBS 2019. *IEEE Transactions on Multimedia* 23 (2021), 243–256. https://doi.org/10.1109/TMM.2020.2980944

[17] Luca Rossetto, Ralph Gasser, Loris Sauter, Abraham Bernstein, and Heiko Schuldt. 2021. A System for Interactive Multimedia Retrieval Evaluations. In *MultiMedia Modeling - 27th International Conference, MMM 2021, Prague, Czech Republic, June 22-24, 2021, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12573)*, Jakub Lokoc, Tomás Skopal, Klaus Schoeffmann, Vasileios Mezaris, Xirong Li, Stefanos Vrochidis, and Ioannis Patras (Eds.). Springer, 385–390. https://doi.org/10.1007/978-3-030-67835-7_33

[18] Luca Rossetto, Ivan Giangreco, Claudiu Tanase, and Heiko Schuldt. 2016. vitrivr: A Flexible Retrieval Stack Supporting Multiple Query Modes for Searching in Multimedia Collections. In *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM 2016, Amsterdam, The Netherlands, October 15-19, 2016*, Alan Hanjalic, Cees Snoek, Marcel Worring, Dick C. A. Bulterman, Benoit Huet, Aisling Kelliher, Yiannis Kompatsiaris, and Jin Li (Eds.). ACM, 1183–1186. https://doi.org/10.1145/2964284.2973797

[19] Klaus Schoeffmann. 2019. Video Browser Showdown 2012-2019: A Review. In *2019 International Conference on Content-Based Multimedia Indexing, CBMI 2019, Dublin, Ireland, September 4-6, 2019*, Cathal Gurrin, Björn Þór Jónsson, Renaud Péteri, Stevan Rudinac, Stéphane Marchand-Maillet, Georges Quénot, Kevin McGuinness, Gylfi Þór Guðmundsson, Suzanne Little, Marie Katsurai, and Graham Healy (Eds.). IEEE, 1–4. https://doi.org/10.1109/CBMI.2019.8877397